AmazonExplorer チュートリアル（Flex Builder 2　beta2 版）

株式会社セカンドファクトリー
デベロップメントセンター
RIA エンジニア

齋藤栄二

AmazonExplorer について

● ダウンロードソースについて

ダウンロードした SampleFiles.zip を解凍します。
解凍後の SampleFiles フォルダには以下のファイル・フォルダが含まれています。

- **AmazonExplorer について.pdf**
- **AmazonExplorer フォルダ**
  - ├AmazonExplorer.mxml
  - └images フォルダ
    - ├no_image_ss.gif（サムネイル画像が取得できない場合の代替画像）
    - └no_image_mm.gif（商品詳細用画像が取得できない場合の代替画像）

現在、beta2 版では日本語環境ではプロジェクトコンパイル時に日本語のリソースバンドルがありません。

FlexBuilder がインストールされているフォルダの¥Flex Framework 2¥frameworks¥locale フォルダの中に「ja_JP」というフォルダを作って、その中に現在ある「en_US」 フォルダ中のものをコピーをしてください。
例 : C:¥Program Files¥Adobe¥Flex Builder 2 Beta 2¥Flex Framework 2¥frameworks¥locale¥ja_JP

その後、FlexBuilder の「project」メニューから「clean」を実行してください。

● ダウンロードソースの使用方法とチュートリアル開始手順

ダウンロードした SampleFiles.zip を解凍し、Flex Builder で新規プロジェクトを作成します。

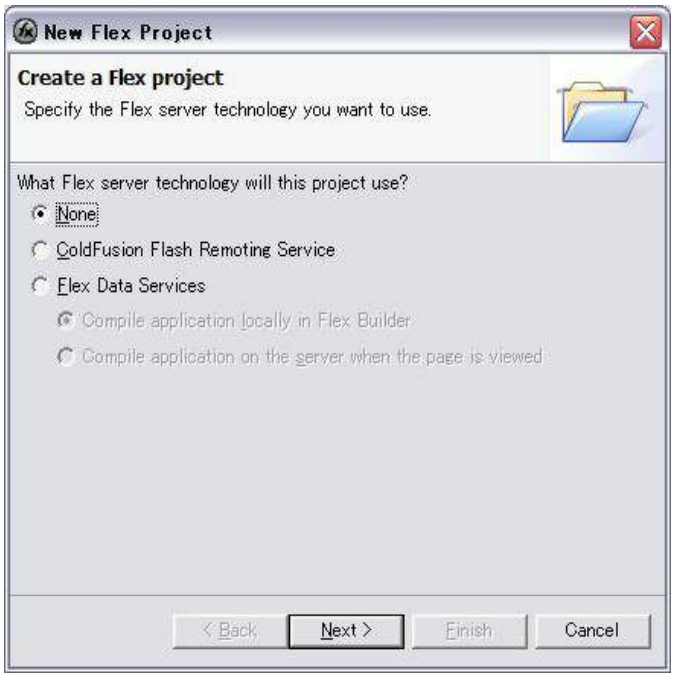
1. Flex Builder の File メニューから New＞Flex Project や Navigator View を右クリックし、同じ様に New＞Flex Project を選択します。

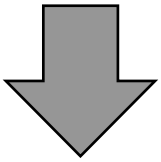
2. New Flex Project ダイアログで、まず Flex Data Services などの使用確認が表示されますが、Flex server technology は使用しないので「None」ラジオボタンが選択されている状態で次へ進みます。

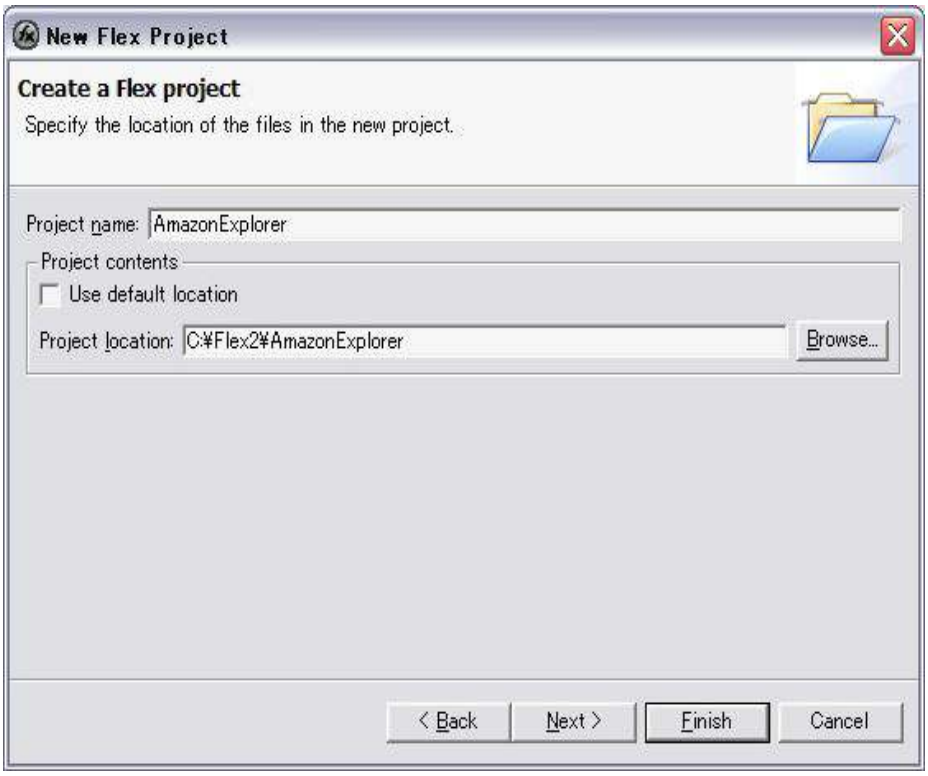
3. Project name に AmazonExplorer など任意の名称を設定し、Project location に解凍して出来た **AmazonExplorer フォルダを指定**し、次へ進みます。

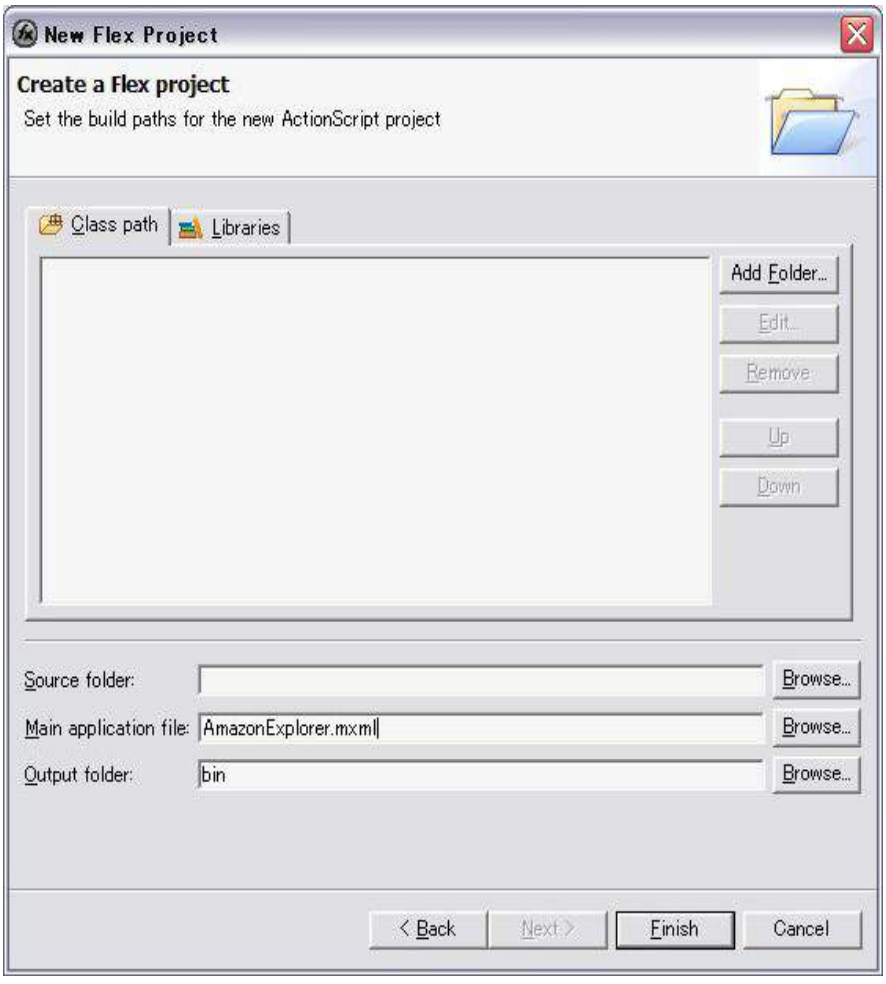
4. 次の画面では Main application file に **AmazonExplorer フォルダ内の AmazonExplorer.mxml** を指定し、「Finish」ボタンを押すと、ダウンロードしたソースが新規プロジェクトとして作成されます。

<u>ダウンロードソースから新規プロジェクトを作成</u>






New Flex Project
Create a Flex project
Specify the location of the files in the new project.
Project name: AmazonExplorer
Project contents
Use default location
Project location: C:¥Flex2¥AmazonExplorer
Browse...
< Back
Next >
Finish
Cancel


New Flex Project
Create a Flex project
Set the build paths for the new ActionScript project
Class path
Libraries
Add Folder...
Edit...
Remove
Up
Down
Source folder:
Main application file: AmazonExplorer.mxml
Output folder: bin
Browse...
< Back
Next >
Finish
Cancel

アプリケーションの右クリック時に表示される「View Source」でダウンロードしたソースではプロジェクトの import が使用できません。Flex Builder でプロジェクトの import をする場合には import するフォルダやアーカイブファイルなどに「.project」ファイルが必要となります。

●Amazon Web サービスを使用するにあたって

このサンプルアプリケーションで、実際に Amazon の商品データを使用するには Amazon Web サービスへの登録が必要です。

1. このサンプルアプリケーションではAmazon Web サービスを利用しています。Amazon Web サービスのAPI 利用時には、送信パラメータで登録IDを指定する必要があります。

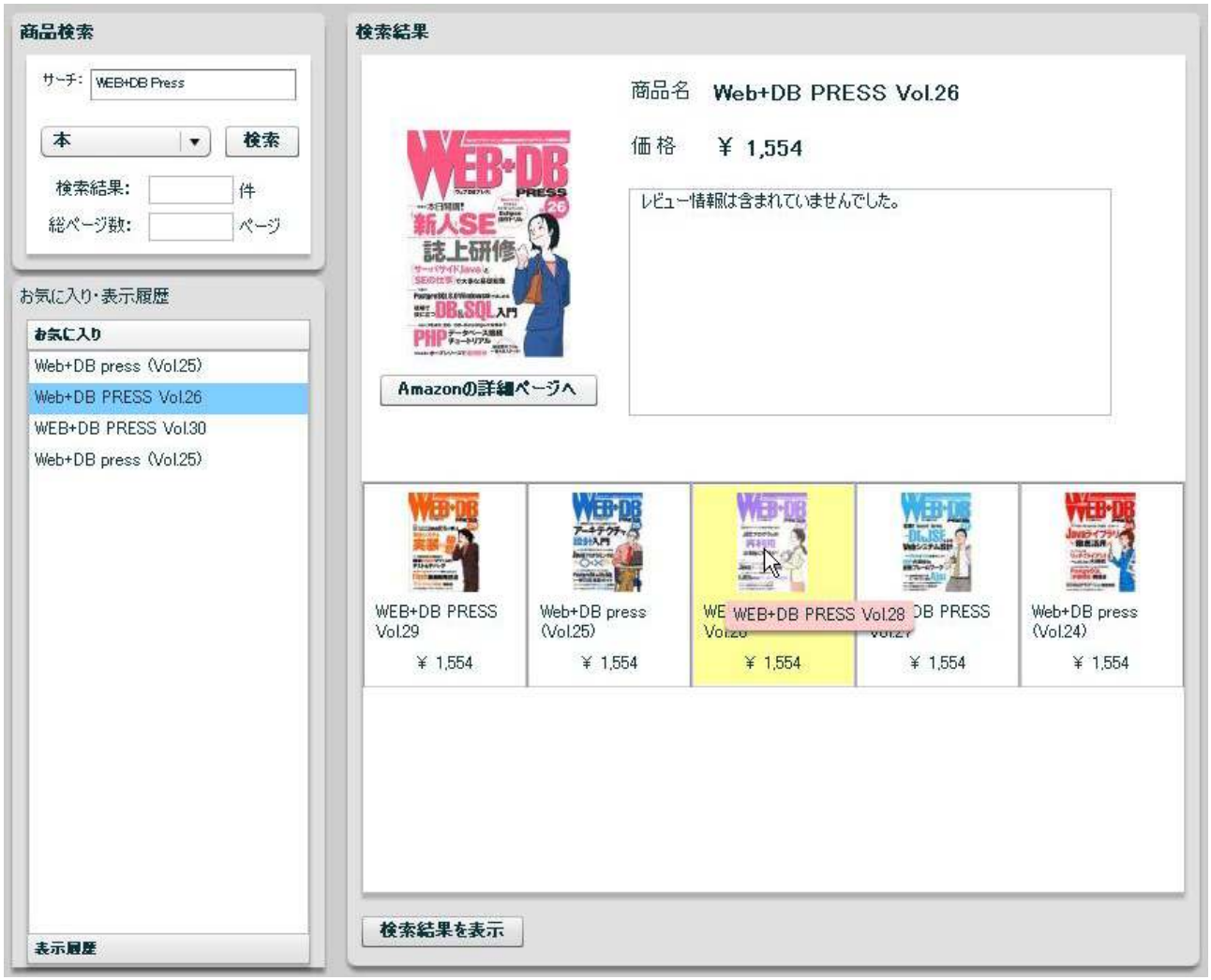
2. Amazon Web サービスの登録IDをお持ちで無い方は、Amazon Web サービスに登録し、登録ID(Subscription ID)を取得する必要があります。 無料で取得することができます。

3. サンプルソース内の`<mx:Script>`タグに記述してある `_AWS_ACCESS_KEY_ID` に、取得した登録 ID(Subscription ID)を設定する必要があります。
   例：`private var _AWS_ACCESS_KEY_ID:String="0123456789ABCDEFGHI";`

Amazon.co.jp Web サービスについて
http://www.amazon.co.jp/exec/obidos/subst/associates/join/webservices.html/249-2990936-5649937

●アプリケーション概要

・Amazon Web サービスを使用し、キーワード検索の結果を商品サムネイルで表示します。
・検索結果商品サムネイルをクリックすると、その商品詳細情報を表示し、その詳細表示商品に関連する商品をサムネイル表示します。商品サムネイルをクリックする度に次から次へと関連商品サムネイルが差し変わっていきます。
・商品詳細情報を表示した商品の履歴表示し、お気に入りリストを表示します。
・お気に入りリストには商品サムネイルをドラッグ＆ドロップすることで登録することができます。



●チュートリアル

AmazonExplorer についてのチュートリアルです。また付属ソースコード内のコメントなども参考にしてください。

AmazonExplorer には以下の機能があります。

- カテゴリ・キーワード検索から検索結果をサムネイル表示をする
- 検索結果表示のページング
- 検索結果サムネイルをクリックすると商品詳細を取得し、詳細表示する
- 検索結果サムネイル表示を詳細を表示している商品に関連するものに差し替える
- 検索パネルをリサイズし、検索結果数、総ページ数を表示する
- 詳細表示した商品の表示履歴をリスト表示する
- 商品サムネイルをお気に入りにドラッグ＆ドロップで登録

1. アプリケーション全体のレイアウト

完成版サンプルでは左側に検索パネル、お気に入り・履歴パネル、右側に商品サムネイル表示、商品詳細情報を表示するレイアウトになります。
実際の作業の前に全体の画面構成と画面遷移を把握しておくと良いでしょう。

<mx:Hbox>でアプリケーションのレイアウトを左右に分け、<mx:Spacer/>で間隔を調節しています。左側の<mx:VBox>に検索用パネル<mx:Panel>（searchPanel）、お気に入り・履歴表示パネル<mx:Panel>（favoriteAndHistoryPanel）が配置されます。
右側には商品サムネイルを表示する<mx:TileList>（itemThumbnailList）と、詳細表示用にCanvasを含む<mx:Panel>（resultPanel）が配置されます。

商品詳細表示時の最終的なアプリケーションレイアウトは以下の様になります。（）内はid名。

```
<mx:Application>
<mx:HBox>
      <!--左側-->
      <mx:VBox>
         <!--検索パネル-->
         <mx:Panel>(searchPanel)
         <!--お気に入り・履歴パネル-->
         <mx:Panel>(favoriteAndHistoryPanel)
            <!--お気に入り・履歴リスト用アコーディオン-->
            <mx:Accordion>
                  <mx:List>(favoriteList)
                  <mx:List>(histotyList)
             </mx:Accordion>
       </mx:Panel>
    </mx:VBox>

   <mx:Spacer/>

   <!--右側-->
   <!--検索結果パネル-->
   <mx:Panel>(resultPanel)
      <mx:VBox>
         <!--商品詳細表示-->
         <mx:Canvas>
            <mx:HBox>
               <mx:VBox>(detailImageBox)
               <mx:VBox>(itemDetailsBox)
            </mx:HBox>
         </mx:Canvas>
         <!--商品サムネイル表示-->
         <mx:TileList>(itemThumbnailList)
      </mx:VBox>
    </mx:Panel>

</mx:HBox>
</mx:Application>
```

また、Base stateを含む、４つのstateを使用しています。

●Base stae
商品検索のためのパネル、右側にはsearchResultStateと同様にサムネイル表示<mx:TileList>用パネルが配置されていますが、非表示になっています。

●searchResultState
検索結果を表示する画面です。サムネイル表示<mx:TileList>用パネルが表示され、商品サムネイルが表示されます。

●showItemDetailsState
商品詳細を表示、関連商品サムネイル表示、表示履歴、お気に入りリストなどが配置されたメイン画面です。

●moveItemListState（エフェクト用）
商品サムネイル表示 <mx:TileList>を下に動かすための画面です。（機能的には無くても良いstateです。）effectEndでshowItemDetailsStateに遷移します。

Base stateからの画面遷移は、searchResultState（検索結果表示）からmoveItemListState（サムネイル下移動）に遷移し、サムネイル下移動終了後、showItemDetailsState（商品詳細、関連商品サムネイル表示）に遷移します。

> メモ：
> 必ずしもmoveItemListStateステートに移動してから商品詳細を表示するというわけではありません。あくまで画面遷移に関しての状態となります。
> このサンプルでは、商品サムネイル表示、詳細表示内容は画面遷移を待たずして、バインディングによってダイレクトに表示内容が変化します。
> 最初のサムネイル下移動時には、moveItemListStateには<mx:Canvas>(detailCanvas)が無いので表示されていないだけで、一度サムネイルが下に移動した後、サムネイルを選択するとmoveItemListStateに遷移しますが、すでにサムネイルのトランジションは済んでいるので（effectEnd）、showItemDetailsStateに遷移します。

2. Base stateでレイアウトを作成

アプリケーション全体のレイアウトに関するプロパティを以下の様に設定しました。

Application　：layout="absolute" backgroundColor="0xCCCCCC"

Hbox　　：x="10" y="10"

デザインビュー表示時には、画面のレイアウト階層構造を視覚的に分かりやすく表示する show Container OverLays（F4 キー）機能を使用することができます。またコンポーネントをドラッグ＆ドロップ時に[Ctrl]キーを押しながら行うと、通常のデザインビューと show Container OverLays 状態とを入れ替えて配置することが出来ます。

<mx:Application>のアプリケーションの width、height、などの設定を変更し、実行しても変更が反映されていない場合、メニューバーから [Project]-[Clean] を選択します。

3. コンポーネントを配置し、画面を作成

左側の検索パネルと、右側の検索結果パネルにサムネイル表示をするものを作ってみます。
まずはStateを使用しないで、Base Stateでレイアウト作業をします。

検索パネル<mx:Panel>（searchPanel）に各コンポーネントを配置します。
検索パネル<mx:Panel>の下には後のstateで、お気に入り・履歴表示パネルが後に追加されるためあらかじめ <mx:VBox>を使用しています。

```
<mx:VBox>
   <!--検索パネル-->
   <mx:Panel>(searchPanel)
      <mx:Label>
      <mx:TextInput>(searchText)
      <mx:ComboBox>(categoryCombobox)
      <mx:Button>(searchButton)
      <mx:ControlBar>
   <mx:Panel>
</mx:VBox>
```

<mx:Panel>（searchPanel） には<mx:Label>、<mx:TextInput>、<mx:ComboBox>、<mx:Button>を配置しています。<mx:ControlBar>にはButtonなどを配置することができますが、ここではデザインとして配置してみました。

```
<mx:Panel>
   <mx:Label>
   <mx:TextInput>
   <mx:ComboBox>
   <mx:Button>
   <mx:ControlBar>
</mx:Panel>
```

各プロパティは以下の通りです。

```
Panel       : width="220" height="120" layout="absolute"
              id="searchPanel" title="商品検索" cornerRadius="8"
Label       : x="10" y="10" id="keywordLabel" text="サーチ："
TextInput   : x="45" y="10" width="145" id="searchText"
              text="Amazon"
ComboBox    : x="10" y="50" width="120" id="categoryCombobox"
Button      : x="140" y="50" id="searchButton" label="検索"
ControlBar  : height="10" id="searchPanelControlBar"
```

またPanelでの使用フォント・サイズは fontFamily="_ゴシック" fontSize="12" とし、Label・TextInput では、fontFamily="_ゴシック" fontSize="10"と指定しています。
ComboBox と Button ではfontFamily="_ゴシック"のみ指定しました。

4. CcomboBoxの内容をdataProviderに

ComboBox の「中身」を dataProvider という形式で記述します。Flex Builder をコードビューに切り替えて、<mx:dataProvider>、<mx:ArrayCollection>、<mx:source>、<mx:Object>を<mx:ComboBox>～</mx:ComboBox>の中に記述します。

```
<mx:ComboBox>
   <!--コンボボックスにデータ登録 dataProvider用にArrayCollectionへ-->
   <mx:dataProvider>
      <mx:ArrayCollection>
         <mx:source>
            <mx:Object label="本" data="Books"/>
            <mx:Object label="ミュージック" data="Music"/>
            <mx:Object label="ソフトウェア" data="Software"/>
            <mx:Object label="DVD" data="DVD"/>
         </mx:source>
      </mx:ArrayCollection>
   </mx:dataProvider>
</mx:ComboBox>
```

5. 検索結果表示用パネル

右側の検索結果表示用パネルにサムネイル表示用の<mx:TileList>（itemThumbnailList）を配置します。検索取得結果をサムネイル表示するために、<mx:TileList>の dataProvider に thumbnailCollection を登録します。検索ボタンのクリックによって Amazon Web サービスへの検索が行われた後、取得結果を thumbnailCollection へ格納するような構造になります。

```
<mx:Panel>(resultPanel)
   <mx:VBox>(vboxRight)
      <!--商品サムネイル表示-->
      <mx:TileList>(itemThumbnailList)
   </mx:VBox>
   <mx:ControlBar>
</mx:Panel>
```

resultPanel などのプロパティ：

Panel　　: width="600" height="370" id="resultPanel" title="検索結果" cornerRadius="8" backgroundColor="#FFFFFF"

VBox　　: width="100%" id="vboxRight"

ControlBar　: height="50" id="resultPanelControlBar"

TileList のプロパティは以下の様に設定します。

```
x="0" y="0" id="itemThumbnailList" dataProvider="{thumbnailCollection}"
width="100%" height="290" columnCount="5" backgroundColor="#ffffff"
selectionColor="0xFFCC66" rollOverColor="0xFFFF99"
```

HTTP サービスでデータが返された時に実行する関数 itemSearchResultHandler（）で、<mx:TileList>の dataProvider="{thumbnailCollection}" に、取得データを整形したものを登録しています。これによって、thumbnailCollection の値が変化すると<mx:TileList>の表示内容も同じように変化します。

```
<mx:TileList id="itemThumbnailList" dataProvider="{thumbnailCollection}">
```

<mx:TileList>でリスト表示する際に、画像<mx:Image>、商品名<mx:Text>、価格<mx:Label>をまとまりにしてひとつのリスト内のアイテムとして表示させたるため、itemRenderer プロパティを独自に設定しています。
通常は itemRenderer="Button" など<mx:TileList>のプロパティで設定し、"Button"と設定した場合にはリスト内のアイテムが全てボタンとなります。
（参考：Flex Samples Explorer では別 mxml ファイルに分けて同様の処理をしています。）

```
<mx:TileList>
   <!--カスタムitemRendererを設定-->
   <mx:itemRenderer>
     <mx:Component>
        <!--VBoxにツールチップ表示-->
          <mx:VBox>
            <mx:Image>
            <mx:Text>
            <mx:Label>
        </mx:VBox>
     </mx:Component>
   </mx:itemRenderer>
</mx:TileList>
```

itemRenderer プロパティをカスタマイズするには以下の様に<mx:itemRenderer><mx:Component>～</mx:Component></mx:itemRenderer>の中に表示する内容を記述します。ここでは<mx:Vbox>の中に画像<mx:Image>、商品名<mx:Text>、価格<mx:Label>をまとまりにして表示します。

```
<!--itemRenderer をカスタマイズします-->
<mx:itemRenderer>
   <mx:Component>
      <!--画像・商品名・価格を Vbox に入れます。-->
      <mx:Vbox height="140" horizontalAlign="center"
          toolTip="{data.TitleName}"
          borderColor="0xCCCCCC" borderStyle="inset">
        <!--画像-->
        <mx:Image id="itemImage" source="{data.imageURL}" height="70"
         horizontalAlign="center" verticalAlign="middle"/>
        <!--商品名-->
        <mx:Text id="itemTitle" height="30" width="100"
         text="{data.TitleName}"
         fontFamily="_ゴシック" fontSize="12" textAlign="left"
         selectable="false"/>
        <!--価格-->
        <mx:Label id="itemPrice" width="100" text="{data.PriceAmount}"
         fontFamily="_ゴシック" fontSize="12"/>
```

```
      </mx:VBox>
   </mx:Component>
</mx:itemRenderer>
```

<mx:itemRenderer>内では「data.dataProvider の内容」と dataProvider のデータにアクセスします。{data.TitleName}・{data.imageURL}などのようになります。dataProvider のデータを参照し<mx:Image>、<mx:Text>、<mx:Label>に表示する内容を設定します。

この<mx:VBox>にツールチップを表示する設定を追加してみます。
<mx:VBox>の toolTip プロパティに「toolTip="{data.TitleName}"」とツールチップで表示する内容を設定するだけです。

ツールチップのスタイル設定も以下の様にタグで簡単に設定することができます。

```
<mx:Style>
    ToolTip {
        fontFamily: "_ゴシック";
        fontSize: 12;
        backgroundColor: #FFCCCC;
    }
</mx:Style>
```

これで検索キーワードを入力し、検索ボタンをクリックした後、検索結果商品サムネイルが表示される画面ができあがりました。

6. stateの作成

新しいstateを作成し、画面遷移の設定をします。
検索結果がサムネイル表示される検索結果パネルを徐々にフェイド・インして表示するため、新しい state を作成し、画面遷移の状態を作成します。

また、検索パネルではパネルがリサイズされ検索数とページ数を表示、検索結果パネルのコントロールバーには検索結果のページングボタンを配置します。

States パネルから New State を選択し、新規 state を作成します。

名称を searchResultState と設定し、作成元とするステート（Based On）を、<Base State>とします。これで Base state を元に searchResultState が作られました。

検索パネル（searchPanel）

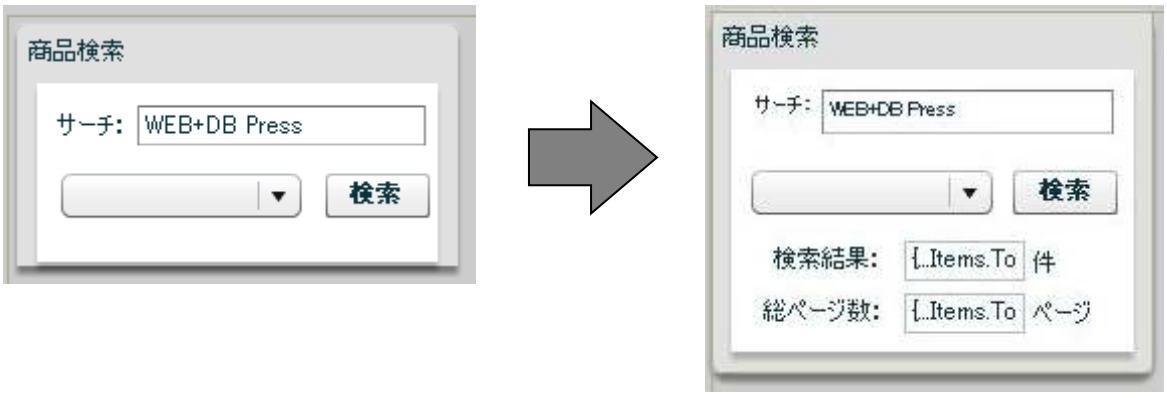
searchResultState で検索パネル searchPanel の height を 120 から 180 に変更し、検索結果件数と総ページ数を表示する<mx:TextArea>（searchTotal・pageTotal）と、付随する<mx:Label>を配置します。

検索結果数を表示する searchTotal の text プロパティに HTTPService(itemSearchSrv)で返される XML 内の TotalResults をバインディングします。（HTTPService については後述。）

```
<mx:TextArea text="{itemSearchSrv.result.Items.TotalResults}"/>
```

同じように総ページ数を表示する<mx:TextArea>(pageTotal)も設定します。

```
<mx:TextArea text="{itemSearchSrv.result.Items.TotalPages}"/>
```



検索結果パネル（resultPanel）

検索結果パネルの ControlBar には検索結果をページングする Button を配置します。
検索結果パネルの ControlBar に<mx:Button>コンポーネントをドラッグすると配置することができます。Button を click した時の処理として<mx:Script>タグの pageSearch()を設定しています。それぞれ引数を 1（次へボタン）と-1（前へボタン）としています。

検索結果パネルのフェイド・インの初期状態（非表示）を設定するために、<Base state>の<mx:Panel>（resultPanel）のプロパティで visible="false"を設定します。これによって初期表示時には resultPanel が非表示になります。searchResultState では resultPanel のプロパティで visible="true"を設定します。設定時にはプロパティパネルを詳細表示にすると visible など各種プロパティが項目として表示されます。state を切り替えながら確認をしてください。

State の遷移設定

Base state と searchResultState という画面の遷移状態を作ることができました。しかし、どのタイミングで state を変えるかというイベントを設定しなければ画面の状態は変化しません。
Base state の検索ボタン<mx:Button>（searchButton）をクリックした時に Amazon Web サービス問い合わせと開始と同時に searchResultState へ遷移させるので、searchButton の click プロパティに currentState='searchResultState'を追加します。

```
<mx:Button x="140" y="50" label="検索" id="searchButton" fontFamily="_ゴシック"
click="currentState='searchResultState'"/>
```

また、resultPanel が非表示から表示する時にフェイド・インするエフェクトを付けるには resultPanel のプロパティで showEffect="Fade"を設定します。
showEffect プロパティを"Fade"と指定するだけでフェイド表現できます。

Base state

```
<mx:Panel width="600" height="370" id="resultPanel" title="検索結果"
fontFamily="_ゴシック" fontSize="12" visible="false" showEffect="Fade">
```

searchResultState

```
<mx:State name="searchResultState">
   <mx:SetProperty target="{resultPanel}" name="visible" value="true"/>
</mx:State>
```

7. HTTPサービスの指定

HTTPサービスの指定と<mx:Script>タグを記述します。

HTTPサービス<mx:HTTPService>の設定

Amazon Web サービスに接続して検索結果を取得するために、<mx:HTTPService>タグを使用し、次のように HTTP サービスを設定します。

```
<mx:HTTPService
    id="itemSearchSrv"
    url=http://webservices.amazon.co.jp/onca/xml
    useProxy="false"
    result="itemSearchResultHandler(event)"
    fault="requestFaultHandler(event)"
    showBusyCursor="true"
    resultFormat = "e4x"/>
```

HTTP サービスで返される XML のフォーマットを E4X(ECMAScript for XML) に指定し、取得データを簡単に扱えるようにしています。ActionScript3.0 では E4X のクラスがサポートされ、XML データを従来よりも簡単に扱うことができるようになっています。

以下の HTTP サービスで返される XML(itemSearchSrv.result)に対して、戻り値を_itemObj に代入したとすると（_itemObj=itemSearchSrv.result）、
_itemObj.Items.Request.ItemSearchRequest.Keywords と値にアクセスすることができます。(値：Amazon)

```
<ItemSearchResponse>
  <Items>
    <Request>
      <IsValid>True</IsValid>
      <ItemSearchRequest>
        <Keywords>Amazon</Keywords>
```

XML の NameSpace
XML に namespace がある場合、E4X では正常に値が取り出せないので<mx:Script>タグ内で、デフォルトの namespace を指定しています。

```
default xml namespace = "http://webservices.amazon.com/AWSECommerceService/2005-10-05";
```

参考：
akihiro kamijo blog
E4X
http://weblogs.macromedia.com/akamijo/archives/2006/02/e4x.cfm
E4X：XML 名前空間の使用
http://weblogs.macromedia.com/akamijo/archives/2006/02/xml_2.cfm

<mx:Script>タグ：itemSearch ()
<mx:ComboBox>（categoryCombobox）・<mx:TextInput>（searchText）から検索用カテゴリとキーワードを取得し、Amazon Web サービスにリクエストします。
<mx:HTTPService>itemSearchSrv.send（）メソッドを実行します。

<mx:Script>タグ：itemSearchResultHandler ()
<mx:HTTPService>（itemSearchSrv）の結果を取得し、必要な情報を取り出します。

検索結果パネル内<mx:TileList>（itemThumbnailList）で検索結果商品を表示するため、<mx:TileList>の dataProvider（dataProvider="{thumbnailCollection}"）に登録されている thumbnailCollection にデータを代入します。
データバインディングで<mx:HTTPService>の結果が<mx:TileList>の表示に反映されます。

<mx:Script>タグ：requestFaultHandler ()
レスポンスが得られない場合 Alert を表示します。

<mx:Button>（searchButton）の click 設定
HTTP サービス用の result、fault イベントに前述の itemSearchResultHandler()、requestFaultHandler()を設定します。
また検索ボタン（searchButton）の click プロパティに itemSearch()を追加します。
これで検索ボタンをクリックすると検索パネルの内容から HTTPService の send パラメータを設定し、要求を送信、Amazon Web サービスからの取得結果を検索結果としてサムネイル表示することになります。

```
<mx:Button x="140" y="50" label="検索" id="searchButton" fontFamily="_ゴシック"
click="itemSearch(),currentState='searchResultState'"/>
```

```
<mx:HTTPService
    id="itemSearchSrv"
    url=http://webservices.amazon.co.jp/onca/xml
    useProxy="false"
    result="itemSearchResultHandler(event)"
    fault="requestFaultHandler(event)"
    showBusyCursor="true"
resultFormat = "e4x"/>
```

コンパイルして確認

<mx:Script>タグの _AWS_ACCESS_KEY_ID に、取得した登録 ID(Subscription ID)を設定し、（例：private var _AWS_ACCESS_KEY_ID:String="0123456789ABCDEFGHI";）コンパイルします。

ここまでで、コンパイルして動作確認をしてみます。
検索キーワードを入力し、ComboBox から検索カテゴリを選択、検索 Button をクリックすると、Amazon Web サービスから取得した結果が検索結果パネル内の TileList にサムネイル表示されます。検索パネルのリサイズや検索結果のフェード・インも確認できると思います。
キーワードやカテゴリを変えてみたり、検索結果のページ数やページングボタンが機能していることを確認してください。

次からは、商品サムネイルを click した時に商品詳細を表示する画面 searchResultState を作成し、動作を設定します。

8. 商品詳細表示画面の設定

商品詳細を表示する画面showItemDetailsStateをsearchResultStateをベースに作成し、お気に入り・表示履歴パネル、商品詳細Canvasを作成します。

お気に入り・表示履歴パネルの配置

検索パネルが配置してある<mx:Vbox>(vboxLeft)に お気に入り・表示履歴パネル<mx:Panel>(favoriteAndHistoryPanel)を追加します。
お気に入り・表示履歴リストは<mx:Accordion>(accordionSet)の中に入れ、アコーディオンパネルで切り替えられるようにしています。

```
<mx:VBox>
   <!--検索パネル-->
   <mx:Panel>(searchPanel)
   <!--お気に入り・履歴パネル-->
   <mx:Panel>(favoriteAndHistoryPanel)
      <!--お気に入り・履歴リスト用アコーディオン-->
      <mx:Accordion>(accordionSet)
         <mx:Canvas>
            <mx:List>(favoriteList)
         </mx:Canvas>
         <mx:Canvas>
            <mx:List>(histotyList)
         </mx:Canvas>
      </mx:Accordion>
   </mx:Panel>
</mx:VBox>
```

商品詳細Canvasの配置

検索結果パネルに商品詳細 Canvas を配置するために、検索結果パネルと<mx:vBox>(vboxRight)のサイズをします。

```
<mx:SetProperty target="{resultPanel}" name="height" value="670"/>
<mx:SetProperty target="{vboxRight}" name="height" value="590"/>
```

サムネイル表示<mx:TileLis>(itemThumbnailList)が配置してある<mx:Vbox>(vboxRight)に 商品詳細<mx:Canvas>(detailCanvas)を追加します。商品詳細 Canvasn には商品詳細表示するためのコンポーネント類を配置しています。

商品詳細<mx:Canvas>(detailCanvas)を Hbox で左右に分け、左側には商品詳細画像<mx:Image>(detailImage)、商品の Amazon リンクボタン<mx:Button>(itemLinkButton)を、<mx:VBox>(detailImageBox)の中に配置しています。

これら<mx:Image>の source プロパティの表示画像 URL や Button>(itemLinkButton)のリンク先などは<mx:Script>から動的に設定しています。

右側には商品名<mx:Text>(titleText)・価格<mx:Text>(priceText)をそれぞれ<mx:HBox>でセットにして、商品レビューは<mx:TextArea>(reviewText)として配置しています。それらを<mx:VBox>(detailTextBox)を使用し縦に並べて配置しています。

```
※<mx:Canvas>(detailCanvas)<mx:HBox>(itemDetailsBox)タグ内つづき
   <!–商品名・価格・商品レビュー-->
   <mx:VBox>(detailTextBox)
      <mx:Spacer/>
      <!--商品名-->
      <mx:HBox>
         <mx:Label>
         <mx:Text>(titleText)
      </mx:HBox>
```

```
            <mx:Spacer/>

            <!--価格-->
            <mx:HBox>
               <mx:Label>
               <mx:Text>(priceText)
            </mx:HBox>

            <mx:Spacer/>

            <!--商品レビュー-->
            <mx:TextArea>(reviewText)
         </mx:VBox>

   </mx:HBox>
</mx:Canvas>
```

表示される商品詳細情報も<mx:Script>から動的に設定しています。またページングボタンを消し、検索結果に戻るボタンを表示を配置しています。検索結果に戻るボタンは再検索を行います。

```
<mx:Button label="検索結果を表示" id="showResultButton" fontFamily="_ゴシック"
fontSize="12" click="itemSearch()"/>
```

9. お気に入り・履歴表示パネルの設定

アコーディオンパネル内のお気に入り・履歴表示リストにデータバインディングを設定し、リスト内容が動的に反映されるようになっています。

お気に入りリストドラッグ＆ドロップの設定

お気に入りリスト<mx:List>（favoriteList）には商品表示サムネイルをドロップできるように設定します。

drag 対象：<mx:TileList>（itemThumbnailList）
drop 対象：<mx:List>（favoriteList）

リストをドロップの対象に設定するには **dropEnabled** プロパティを true と設定するだけです。

すでにドラッグ対象の商品サムネイル表示<mx:TileList>（itemThumbnailList）には dataProvider="{thumbnailCollection}" と、商品のデータがバインディングされています。そのため商品サムネイルがドロップされると itemThumbnailList に登録されているドラッグ中の商品データが、favoriteList の dataProvider に登録されている favoriteCollection に追加されます。ドラッグ対象の<mx:TileList>にはプロパティで **dragEnabled**="true" を設定します。

dataProvider="{favoriteCollection}"とバインディングされいるので、ドロップされたことで favoriteCollection の値が変わり、リスト表示内容が変化します。

リストに商品名を表示するため<mx:List>の labelField プロパティに dataProvider に登録されている TitleName を指定しています。

```
<mx:List id="favoriteList" dataProvider="{favoriteCollection}"
    labelField="TitleName"  width="198" height="100%"
    variableRowHeight="true" wordWrap="true"
    itemClick="getItemDetailsFromfavorite(),setHistoryFromFavorite()"
    dropEnabled="true">
```

表示履歴リストの設定

表示履歴リスト<mx:List>（histotyList）の場合でも、同じように dataProvider に指定している historyCollection の値が変化するとリストの表示内容が変化します。
商品表示サムネイル<mx:TileList>（itemThumbnailList）の click、お気に入りリストの click、表示履歴リストの click で historyCollection に登録され、表示履歴リストに項目が追加表示されます。
HistotyList でも、dataProvider="{historyCollection}" とバインディングされいるので、historyCollection の値が変わるとリスト表示内容が変化するというわけです。

表示履歴リストでも labelField プロパティに TitleName を指定しています。

```
<mx:List id="histotyList" dataProvider="{historyCollection}"
    labelField="TitleName"  width="198" height="100%"
    variableRowHeight="true" wordWrap="true"
    itemClick="getItemDetailsFromHistory()">
```

10. stae移動時のトランジション設定

state 遷移時のトランジションを設定するのに<mx:Transition>タグを使用しています。
現在の state と遷移する state を指定し、状態を変化させる targets やエフェクトの種類<mx:Move>や<mx:Resize>、また duration などプロパティを指定します。

```
<mx:transitions>

   searchResultState への遷移時 Transition（検索件数パネルのリサイズ）
   <mx:Transition id="searchTransition"
        fromState="*" toState="searchResultState">
        <mx:Resize target="{searchPanel}" duration="800" />
   </mx:Transition>

   moveItemListState への遷移時 Transition（商品サムネイルの下移動）
   <mx:Transition id="moveDownTransition"
        fromState="searchResultState" toState="moveItemListState">
        <mx:Move target="{itemThumbnailList}" duration="800" />
   </mx:Transition>

</mx:transitions>
```

トランジションで複数のエフェクトを、順番に再生させる場合は<mx:Sequence>タグ、同時に再生させる場合は<mx:Parallel>タグを使用します。

<mx:Sequence>タグ使用例

```
<mx:transitions>
   <mx:Transition id="searchTransition"
     fromState="*" toState="searchResultState">
      <mx:Sequence id="searchParallel" targets="{[searchPanel]}">
         <mx:Move duration="400" />
         <mx:Resize duration="800" />
      </mx:Sequence>
   </mx:Transition>
<mx:transitions>
```

ちなみにフェード・インの duration タイムを調節する場合は以下の通りです。

```
<mx:Fade id="Fade" duration="400"/>
```

変更する<mx:Fade>エフェクトの id を指定します。デフォルトの id が Fade のため、この例では id="Fade"となっています。

11. <mx:Script>タグ（ActionScript）

サンプル中に記述した ActionScript 部分でいくつかポイントを説明します。
ActionScript は<mx:Script>タグ内に記述します。

```
<mx:Script>
   <![CDATA[
      trace("ActionScript3.0");
   ]]>
</mx:Script>
```

また、<mx:Script>タグの CDATA 以下を外部 ActionScript ファイルにして include しても同様です。

例：amazonExplorer.as を外部 ActionScript として用意した場合は以下の様になります。

```
<mx:Script>
   <![CDATA[
      include "amazonExplorer.as";
   ]]>
</mx:Script>
```

●dataProvider に指定した変数

<mx:TileList> など dataProvider に指定した変数（例：ArrayCollection）に値が入って無い状態では、
コンパイル時に警告「Data binding will not be able to detect assignments to　・・・」が表示されます。
商品サムネイルの <mx:TileList> の dataProvider="{thumbnailCollection}"には<mx:HTTPService>(itemSearchSrv)の結果が返ってくるまで値が無い状態です。
お気に入りリストや表示履歴リストでも値が代入されるまで同じ事が言えます。
この警告は変数 thumbnailCollection を[Bindable]と指定することで解消されます。

```
[Bindable]
privete var thumbnailCollection:ArrayCollection = new ArrayCollection();
```

●画像の埋め込み

サンプルで使用している詳細表示用「No Image」画像(no_image_mm.gif)はローカルの「images」フォルダにありますが、コンパイル時に画像をアプリケーションに埋め込んでいます。

Embed する画像を以下の様に指定します。

```
[Embed(source="images/no_image_mm.gif")]
public var noImage_mm:Class;
```

サンプル中では商品詳細の画像<mx:Image>(detailImage)で、Amazon Web サービスで取得した結果に画像の URL が無い場合、ローカルの「No Image」画像を表示しています。
この商品詳細用「No Image」画像を Embed し、画像 URL が無かった場合には<mx:Image>タグの source プロパティに指定し、Embed 画像を表示しています。

```
detailImage.source=noImage_mm;
detailImage.load();
```

商品サムネイル表示時の「No Image」画像(no_image_ss.gif)は、ローカルの「images」フォルダ中の画像を表示する度にロードしていますが、商品詳細用「No Image」画像(no_image_mm.gif)はコンパイル時に埋め込まれているので、コンパイル済みの「bin/images」フォルダの中を確認すると、Embed した商品詳細用の no_image_mm.gif が含まれていないのが確認できます。

●コンパイルして動作確認

<mx:Script>タグの _AWS_ACCESS_KEY_ID に、取得した登録 ID(Subscription ID)を設定し、（例：private var _AWS_ACCESS_KEY_ID:String="0123456789ABCDEFGHI";）コンパイルします。

検索キーワードを入力し、ComboBox から検索カテゴリを選択、検索 Button をクリックすると、Amazon Web サービスから取得した結果が検索結果パネル内の TileList にサムネイル表示されます。

商品サムネイル表示を click すると、その商品の詳細情報が表示され、表示されていた商品サムネイルが、その商品に関連する商品のサムネイルに差し変わっていきます。

※サンプルについて
FlexBuilder のデバッグ時に Amazon から画像を取得する際、セキュリティー警告が表示されます。