AmazonExplorer_tutorial 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_tutorial 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_tutorial.zip を解凍します。
解凍後の SampleFiles_tutorial 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_tutorial について.pdf**
- **TutorialsMXML フォルダ**
  - ├tutorial1.mxml
  - ├tutorial2.mxml
  - ├tutorial3.mxml
  - ├tutorial4.mxml
  - └tutorial5.mxml
- **AmazonExplorer_tutorial フォルダ**
  - ├tutorial.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）

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_tutorial.zip を解凍し、Flex Builder で新規プロジェクトを作成します。

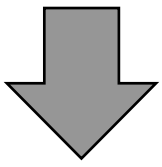
1. Flex Builder の File メニューから New＞Flex Project や Navigator View を右クリックし、同じ様に New＞Flex Project を選択します。

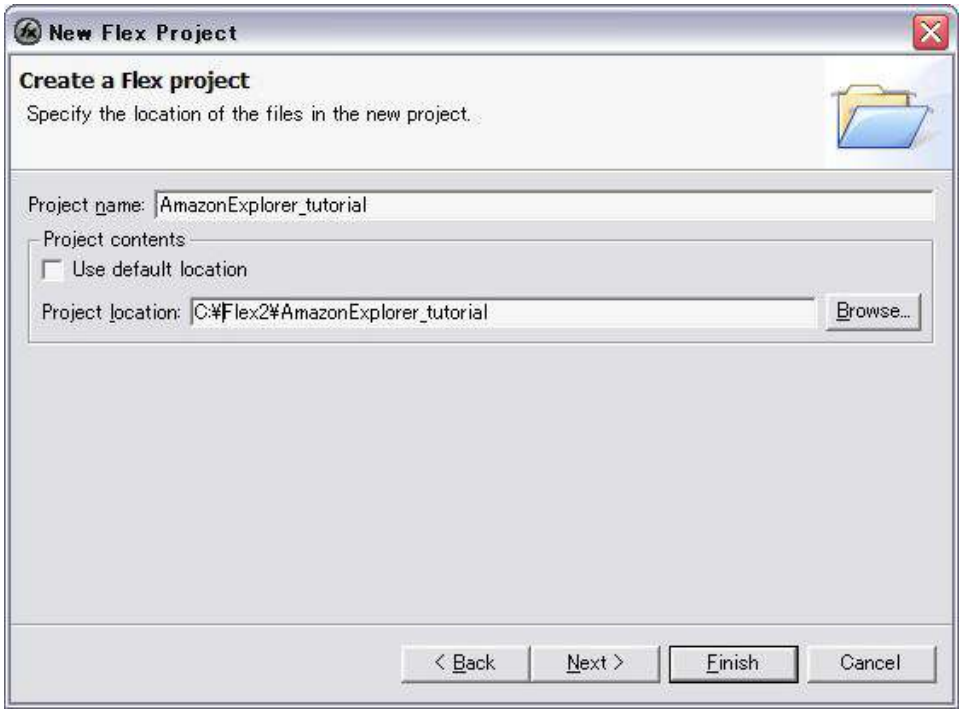
2. New Flex Project ダイアログで、まず Flex Enterprise Services の使用確認が表示されますが、Flex Enterprise Services は使用しないので「No」ラジオボタンが選択されている状態で次へ進みます。

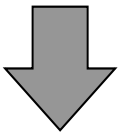
3. Project name に AmazonExplorer_tutorial など任意の名称を設定し、Project location に解凍して出来た **AmazonExplorer_tutorial フォルダを指定**します。

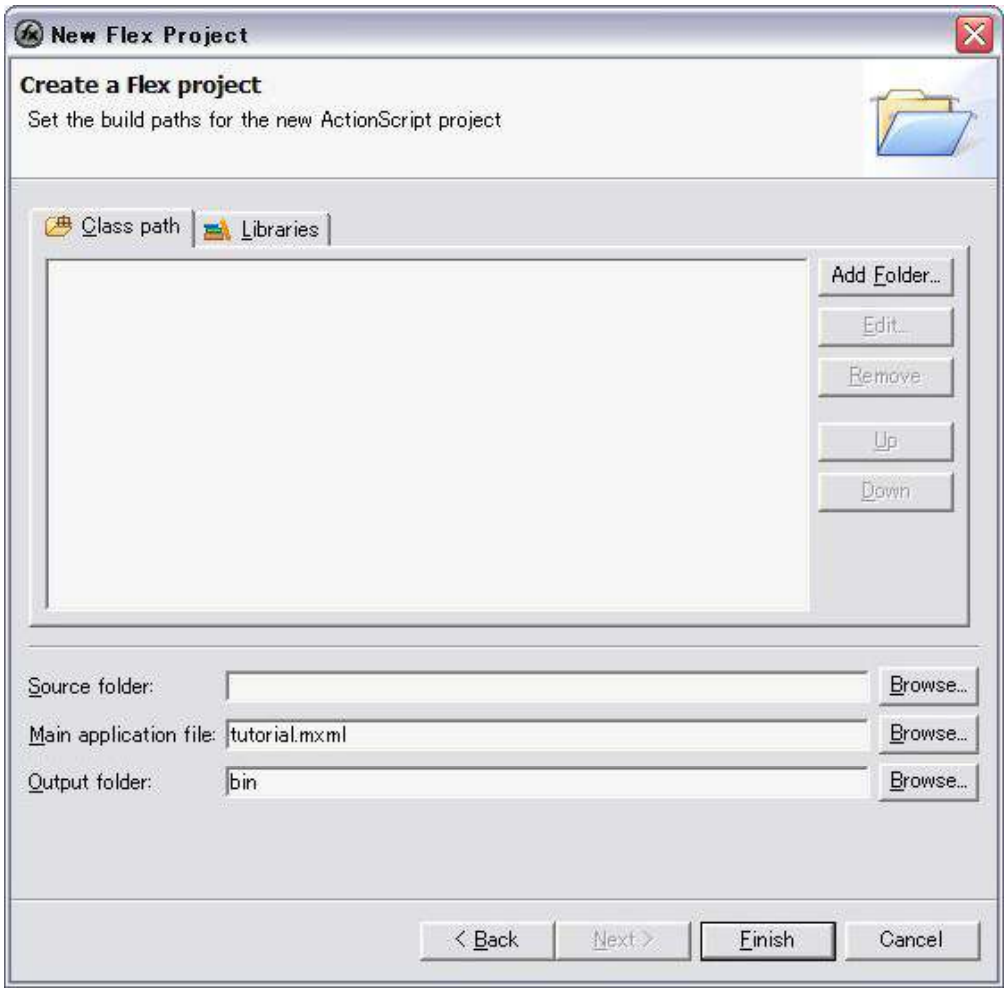
4. Main application file に **AmazonExplorer_tutorial.mxml** を指定し、「Finish」ボタンを押すと、ダウンロードしたソースが新規プロジェクトとして作成されます。

ダウンロードソースから新規プロジェクトを作成




New Flex Project
Create a Flex project
Specify the location of the files in the new project.
Project name: AmazonExplorer_tutorial
Project contents
Use default location
Project location: C:¥Flex2¥AmazonExplorer_tutorial
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: tutorial.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

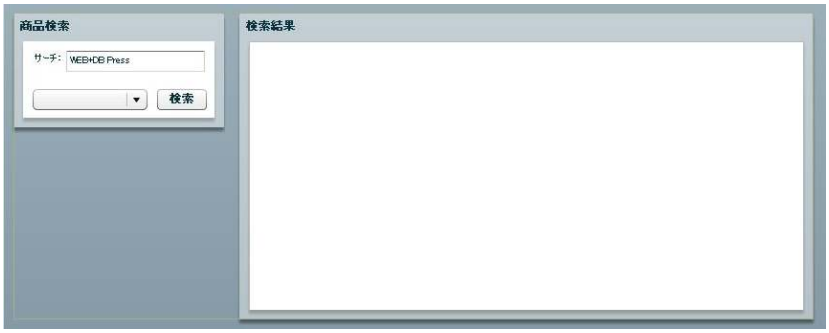
●tutorial1.mxml ～ tutorial5.mxml 解説

チュートリアル記事の段階によってそれぞれの MXML ファイルを用意しています。

1. tutorial1.mxml

アプリケーションの全体的なレイアウト作成。
<mx:Hbox>でアプリケーションのレイアウトを左右に分け、左右それぞれに<mx:Panel>を配置し、左側の Panel には<mx:Label>、<mx:TextInput>、<mx:ComboBox>、<mx:Button>を配置しています。

```
<mx:Application>
  <mx:HBox>
    <mx:Panel>
      <mx:Label>
      <mx:TextInput>
      <mx:ComboBox>
      <mx:Button>
    </mx:Panel>
    <mx:Spacer/>
    <mx:Panel>
    </mx:Panel>
  </mx:HBox>
</mx:Application>
```



各プロパティは以下の通りです。

Application : layout="absolute"

Hbox : x="10" y="10"

Panel : width="220" height="120" layout="absolute" id="searchPanel" title="商品検索"

Label : x="10" y="10" text="サーチ : " id="keywordLabel"

TextInput : x="45" y="10" width="145" id="searchText" text="WEB+DB Press"

ComboBox : x="10" y="50" id="categoryCombobox" width="120"

Button : x="140" y="50" label="検索" id="searchButton"

Panel : width="600" height="321" id="resultPanel" title="検索結果"

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。

2. tutorial2.mxml

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>
```

3. tutorial3.mxml

検索結果表示用パネルにサムネイル表示用の<mx:TileList>（id="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>
  <mx:TileList>
</mx:Panel>
```

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 Explore 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>
```

4. tutorial4.mxml

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ー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。

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。

フェードインの初期状態（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。

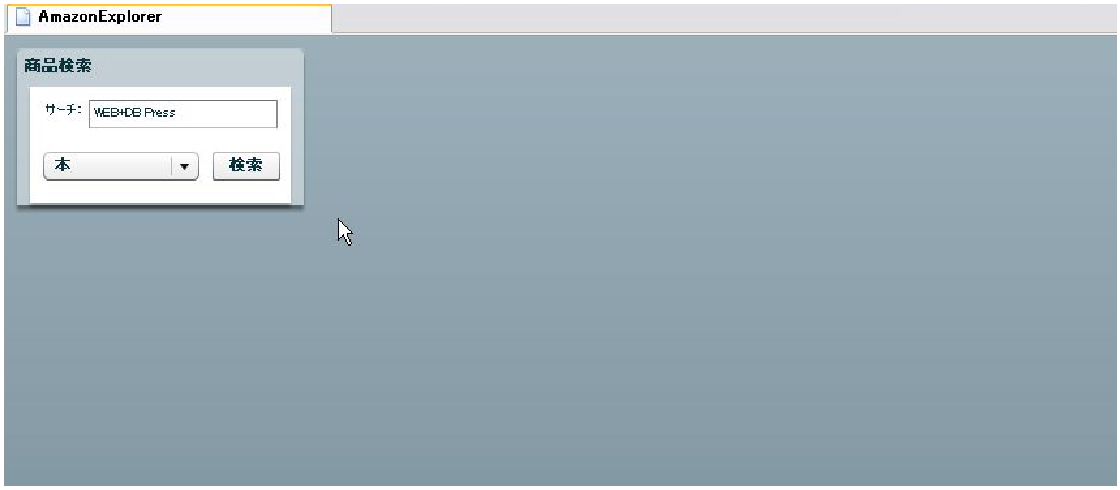
Base state と searchResultState という画面の遷移状態を作ることができました。しかし、どのタイミングで state を変えるかというイベントを設定しなければ画面の状態は変化しません。
Base state の検索ボタン<mx:Button>（searchButton）をクリックした時に Amazon Web サービス問い合わせと開始と同時に searchResultState へ遷移させるので、searchButton の click プロパティに currentState='searchResultState'を追加します。

```
<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="321" 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>
```

Base stateとsearchResultStateの違いはresultPanelのvisibleプロパティのみとなります。

5. tutorial5.mxml

HTTPサービスの指定と<mx:Script>タグを記述します。
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。

※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。