flickr panel search

Creator
セトウナオ（STAC STAR）

画像とタグをシームレスにザッピング

写真共有サービス「flickr」からREST経由でデータを取得する。
ポイントは、必要な写真情報を取得する手続き。

Title
flickr panel search

Sample URL
http://stacstar.jp/flickr_panel/

Archive
flickrpanelsearch.zip

File
14

スクリプト
ActionScript 2.0

対応プレーヤー
Flash Player 8以上

制作アプリケーション
Flash 8

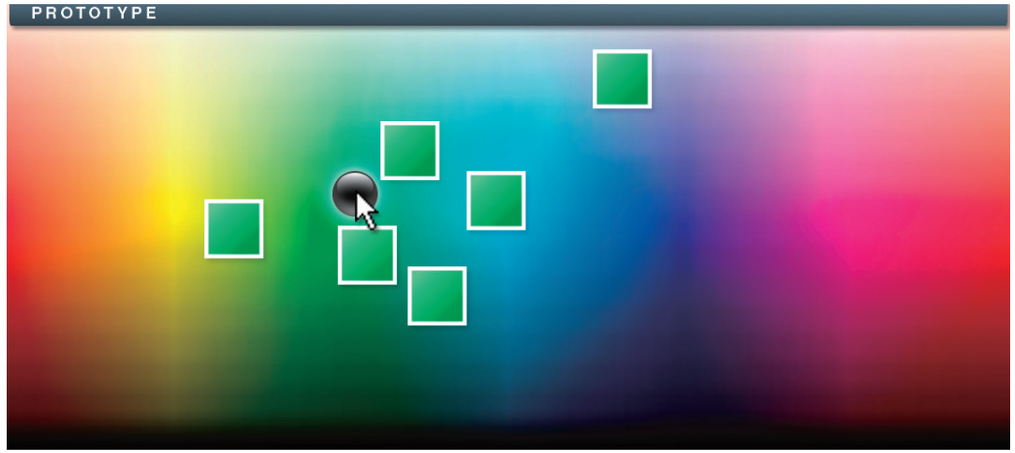
発案～デザイン
「駒」を動かし「検索結果」を取り出す

このコンテンツは、「flickr」の提供している Web サービスを使って写真をザッピングのように探していくコンテンツです。特に、「flickr」のおもしろい概念である「写真（画像）」と「言葉（タグ）」をつなぐ部分だけをピックアップし、それを HTML とは違うインターフェースとして提供しました。「flickr」の提供している HTML のインターフェースとは違い、画像とタグをシームレスにザッピングできるのが特徴です。

○ コンセプト

これを作る前に、頭の中では作ってみたいインターフェースがありました。それは、ステージ上に「検索条件」となるようなものをしきつめ、そのうえに「駒」を配置させ、それを動かすたびに該当する「検索結果」が集まってくるインターフェースでした。たとえば、ステージにカラーパレットをしき、チェスの駒のようなものを動かすたびにその置いた位置にあるカラーに近い画像が集まってくるというようなものを想像していました。

プロトタイプインターフェース



○ 具現化

ここで、想像していたインターフェースと「実際にできること」と「できないこと」の落としどころを模索する作業になります。

まずは、必要となる

- **検索結果**
- **検索条件**
- **駒**

をどういった情報にするかということを考えました。

検索結果は、集まってきたときにそれが即座に“結果”だとユーザーにダイレクトに伝わるほうがよいため、画像（サムネイル）にすることにしました。

検索条件は、使用する Web サービスに大きく依存するので、いくつかの Web サービスを調べてみました。そこで、言葉（タグ）と画像（写真）の結び付きを提供する flickr の Web サービスを利用し、検索条件として「言葉（タグ）」を選択しました。

残るは、駒です。駒に関しては、サムネイルが「言葉（タグ）」を持っているのを利用し、検索結果のサムネイル自身を駒としました（つまり「検索結果」＝「駒」です）。

こうすることで、

「言葉（タグ）」→「画像（写真）」→「言葉（タグ）」→「画像（写真）」

というサイクルが生まれます。

あとは、シームレスに言葉と画像を探索するインターフェースということでデザインを考えました。

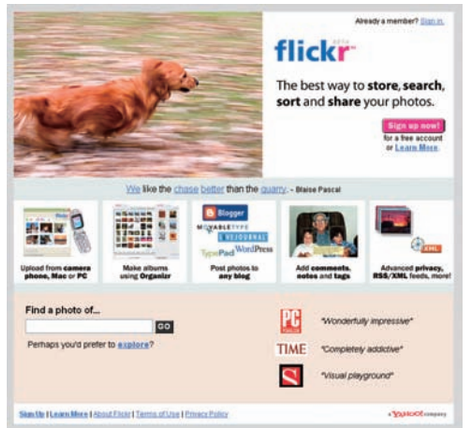
TIPS 写真共有サイト「flickr」

flickr とは、自分の撮った写真や作成した画像をアップロードし、写真を公開できるサービスです。写真を公開するだけではなく、写真に対してタグ（写真へのキーワード）を設定することができます。このタグを付けることにより、画像に「絵」としての要素の他、「言葉」が加わり、写真としての意味が広がります。

たとえば、今、窓から撮った写真をアップロードして公開しても、それは窓から撮った「風景写真」でしかありません。ところが、それに「december」というタグを付ければ写真に時間が加わり、「tokyo」というタグを付ければ、場所の情報が加わります。これにより、写真の表情が広がります。

さらに、ユーザーの検索するキーワードと写真の持つキーワードをマッチングさせれば、検索したり、その写真が持つタグから他の写真を見つけることができます。他にも、グループを作成して同じ興味を持つ人たちが写真を共有したり、特定の所有者の写真を閲覧できたりとさまざまな使い方があります。

flickr では、これらのサービスを外部から利用できるように Web サービスとして提供しています。



URL台 **flickr**
http://www.flickr.com/

スクリプト
Flickr API と Flash の連携

○ flickr の Web サービスへ接続するための準備

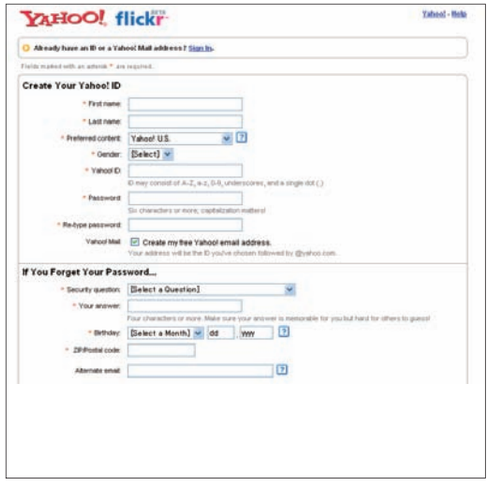
最初に、flickr の Web サービスにつなぐ準備から見ていきます。flickr の Web サービスを使うには、まず flickr のアカウントを取る必要があります。flickr は Yahoo! にてサービスを展開しているので、はじめに Yahoo! への登録が必要です（Yahoo!Japan の ID は使えません）。

登録し flickr にログインした後に、下記 URL にて Flickr API Key を発行してもらう必要があります。API Key とは Web サービスを利用する際に使用する識別子です。

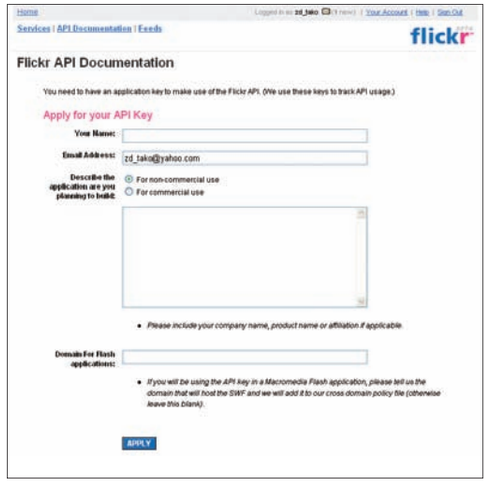
URL㊟ http://www.flickr.com/services/api/key.gne

API Key の発行が完了すると、Web サービスへつなぐ準備は完了です。

Yahoo.com アカウントの登録



Flickr API Key の取得



● REST による Web サービスの利用

flickr の Web サービスでは、XML-RPC、SOAP、REST（REpresentational State Transfer）の 3 つの方式でサービスを提供しています。今回のコンテンツでは、実装が簡単な REST を採用しました。

なぜ簡単かというと、他の 2 つの方法はデータの送受信に XML を使って行うのに対して、REST は URL にクエリ文字列を付加して HTTP GET で送信するだけで結果を取得することができるからです。

簡単にいうと、特定の URL をたたけばその結果を XML で返してくれます。ためしに「flash」というキーワードで関連する写真の一覧を取得してみましょう（＊＊＊＊には、取得した API Key を入れてください）。

```
http://www.flickr.com/services/rest/?method=flickr.photos.search&api_key=＊＊＊＊&tags=flash
```

すると、結果となる写真情報群が XML 形式で取得できます。

写真情報を XML 形式で取得

```
<?xml version="1.0" encoding="utf-8" ?>
- <rsp stat="ok">
  - <photos page="1" pages="120" perpage="100" total="11959">
      <photo id="86829350" owner="16377475@N00" secret="faed9c8b21" server="38" title="Clementine consumption #3" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86829086" owner="16377475@N00" secret="6d1325e659" server="38" title="Clementine consumption #2" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86828712" owner="16377475@N00" secret="7c6b6b6710" server="42" title="Clementine consumption #1" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86792618" owner="49947202@N00" secret="c06a50d857" server="6" title="i turn my back..." ispublic="1" isfriend="0" isfamily="0" />
      <photo id="86750223" owner="88744053@N00" secret="8f0cd5ab86" server="43" title="With flash, without flash." ispublic="1" isfriend="0" isfamily="0" />
      <photo id="86749898" owner="88744053@N00" secret="dbcf78cdaf" server="39" title="Today is my mom's birthday." ispublic="1" isfriend="0" isfamily="0" />
      <photo id="86741420" owner="59239216@N00" secret="c0eab8c25a" server="40" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86741399" owner="59239216@N00" secret="8b0e99d727" server="36" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86741382" owner="59239216@N00" secret="9a31ec6d27" server="40" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86737849" owner="59239216@N00" secret="bdbb5fc149" server="38" title="branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86737802" owner="59239216@N00" secret="64d15813f5" server="6" title="branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86737769" owner="59239216@N00" secret="ec63e97147" server="6" title="branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736214" owner="59239216@N00" secret="866d44d957" server="39" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736196" owner="59239216@N00" secret="90d2334494" server="37" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736175" owner="59239216@N00" secret="adc4628624" server="6" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736162" owner="59239216@N00" secret="7b72a76a4a" server="43" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736146" owner="59239216@N00" secret="450c31b446" server="41" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736125" owner="59239216@N00" secret="faac596ca3" server="39" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736104" owner="59239216@N00" secret="0a96a06482" server="9" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
      <photo id="86736074" owner="59239216@N00" secret="218316b7b8" server="41" title="rain droplets on branchtips" ispublic="1" isfriend="1" isfamily="1" />
```

このURLでは、flickr.photosパッケージのsearch() というメソッドに対して api_key と tags の 2 つの引数をわたして呼び出しました。flickr の REST 方式の書式は、以下の URL のあとに特定のパッケージのメソッドを指定し、API に合った引数を付加して呼び出すだけです。

```
http://www.flickr.com/services/rest/?
```

メソッド群については下記の URL にまとめて解説があるので、どんなメソッドがあるかを最初に把握しておいたほうがいいでしょう。

URL㊟ http://www.flickr.com/services/api/

また、今回このコンテンツでは使用していませんが、flickr の Web サービスに接続するための処理を担うクラス群の AS ファイルも上記 URL からダウンロードできます。

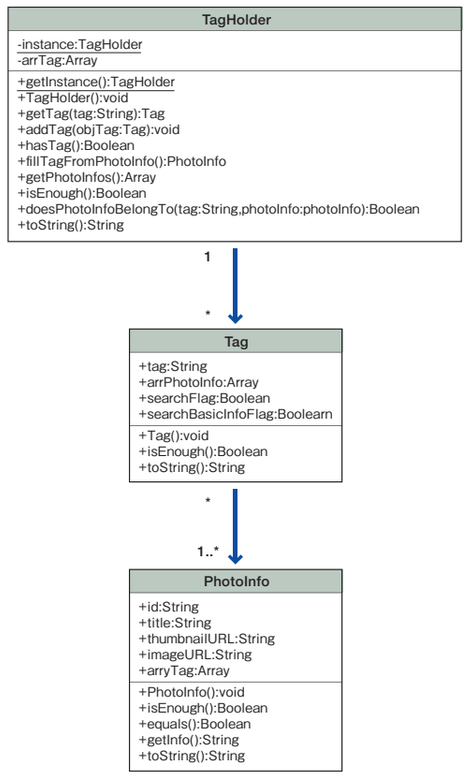
○ データ構造

flickrから取得するデータを格納するクラスとして、このコンテンツでは以下の３つのモデルとなるクラスとして設計しました。

flickr から取得するデータを格納するクラス

クラス	概要
PhotoInfo	写真の情報を格納するクラス。また、isEnough() メソッドにより、このコンテンツで使う写真情報がすべて取得できているかを返す
Tag	タグに関する情報を持つクラス。タグとなるキーワードと写真情報を配列として保持しています。設定されているタグ名での検索が実行されたかどうか、写真の基本情報までの取得が行われているかどうかのフラグを持つ。また、PhotoInfo.isEnough() メソッドを使って格納している写真情報すべてが十分な情報を持っているかを返すメソッドも用意
TagHolder	タグをまとめてあるクラス。fillTagFromPhotoInfo() メソッドは写真情報を引数に取り、その写真情報が持つタグに該当する Tag クラスのインスタンスに写真情報を格納していくメソッド。また、Tag.isEnough() メソッドをダイレクトに呼び出すメソッドも用意

flickr から取得するデータを格納するクラスの構造



○ 写真情報の取得

このスクリプトでキーとなる部分は、flickr から必要な写真情報の取得と読み込まれた後の動作のふるまいです。まずは、写真情報の取得について見ていきます。

このコンテンツにおいて、写真に必要な情報はタイトル、サムネイルの URL、大きな画像の URL、写真に割り当てられたタグ（複数）になります。Web サービスの仕様から、関連しそうなメソッドを探してみます。

写真情報の取得に関連するメソッド（Flickr API）

メソッド	概要
flickr.photos.search	タグから写真の概要の一覧を取得するメソッド
flickr.tags.getListPhoto	写真の ID からタグの一覧を取得するメソッド
flickr.photos.getSizes	画像の URL とそのサイズを取得するメソッド

この辺が写真情報を埋めるのに関係がありそうです。しかし、これを見ると必要な画像情報が一度の接続では取得できないことがわかります。一度の接続で取得できないということは、複数回の接続で取得しなくてはならないということになります。

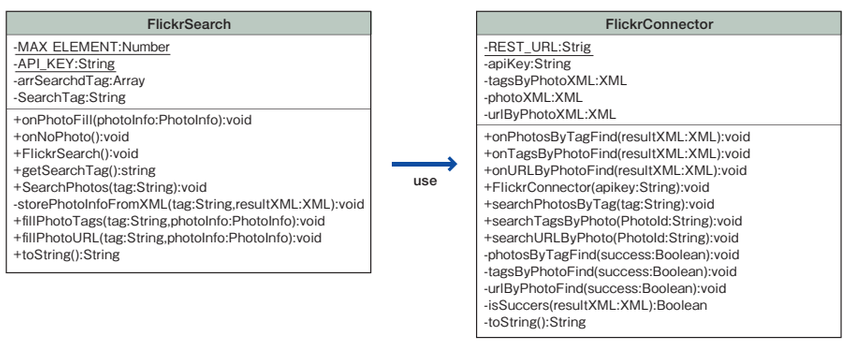
その順番を追ってみましょう。

1. タグに対する画像概要情報の一覧を取得（flickr.photos.search）
2. 一覧から 1 つずつ写真情報を取得し、タグの一覧を取得（flickr.tags.getListPhoto）
3. 画像の URL の取得（flickr.photos.getSizes）

この流れを介して、はじめてこのコンテンツで利用する 1 枚の写真の情報がそろいます。この写真の情報をそろえる役割を担うのが、FlickrSearch クラスと FlickrConnector クラスになります。flickr との直接的な接続を担うのが FlickrConnector クラスです。FlickrConnector の役割は flickr の Web サービスから要求したデータを取得し、エラーがあればエラーの発行・処理を行い、問題がなければ特定のイベントハンドラを実行します。

そして、その FlickrConnector を使ってこのコンテンツ用のふるまいを記述したのが、FlickrSearch クラスになっています。FlickrSearch は、FlickrConnector から取得した XML データを解析し、先に設計したモデルとなるクラスを埋めていく役割です。

Flickrsearch と FlickrConnector



この流れを組んでいるスクリプトのキーとなる部分を抜粋して、一連の流れを追ってみます。まずは、このコンテンツの主となる制御をするFlickrPanelMediatorクラスからFlickrSearch.searchPhotos()メソッドが呼び出されるところからスタートします。

SOURCE FlickrSearch.searchPhotos() メソッド

```
// flickrConnector へ接続の準備
var flickrConnector = new FlickrConnector(FlickrSearch.API_KEY);
var target = this;

flickrConnector.onPhotosByTagFind = function(resultXML:XML) {
     target.storePhotoInfoFromXML(tag, resultXML);
};

flickrConnector.searchPhotosByTag(tag);
```

ここでキーになるのは、FlickrConnectorのイベントハンドラの上書きと検索にかける部分です。検索が終了し、正しいレスポンスが返ってきたときにはstorePhotoInfoFromXML()メソッドが呼び出されるように設定しています。

次にstorePhotoInfoFromXML()メソッドを見てみましょう。

SOURCE FlickrSearch.storePhotoInfoFromXML() メソッド

```
// 写真の基本情報を設定
photoInfo.id = currentNode.attributes['id'];
photoInfo.title = currentNode.attributes['title'];


// TagHolder へ格納
objTag.arrPhotoInfo.push(photoInfo);


// 写真の情報を埋める
this.fillPhotoTags(tag, photoInfo);
```

ここでの主な処理は、取得した写真の概要一覧のXMLを解析しPhotoInfoを作成してTagインスタンスに格納することです。また、そのTagインスタンスをTagHolderへ格納しています。

しかし、これだけでは画像の概要情報（IDとタイトル）しかそろっていません。まずは、その画像の持つタグ情報を取得するためにfillPhotoTags()メソッドを呼び出します。

SOURCE FlickrSearch.fillPhotoTags() メソッド

```
// FlickrConnector.onTagsByPhotoFind イベントハンドラを上書きし、
// 写真情報のタグを補完する。
flickrConnector.onTagsByPhotoFind = function(resultXML) {
          .
          .
```

次のページに続く


```
          .
     while (currentNode != null) {
          if (currentNode.nodeName != 'tag')
               throw new FlickrException(this.toString() + ' fillPhotoTags() 予期しないレスポンスを受け取りました。');

          photoInfo.arrTag.push(currentNode.firstChild.nodeValue);

          currentNode = currentNode.nextSibling;
     }

     // 画像の URL を埋める
     target.fillPhotoURL(tag, photoInfo);


// 写真 ID からタグを取得する
flickrConnector.searchTagsByPhoto(photoInfo.id);
```

ここでは、FlickrConnectorを通してタグ情報を検索させ、取得に成功した際のonTagsByPhotoFind()イベントハンドラを上書きしています。その上書きしたメソッドの中で取得したXMLを解析し、写真情報にタグを追加しています。

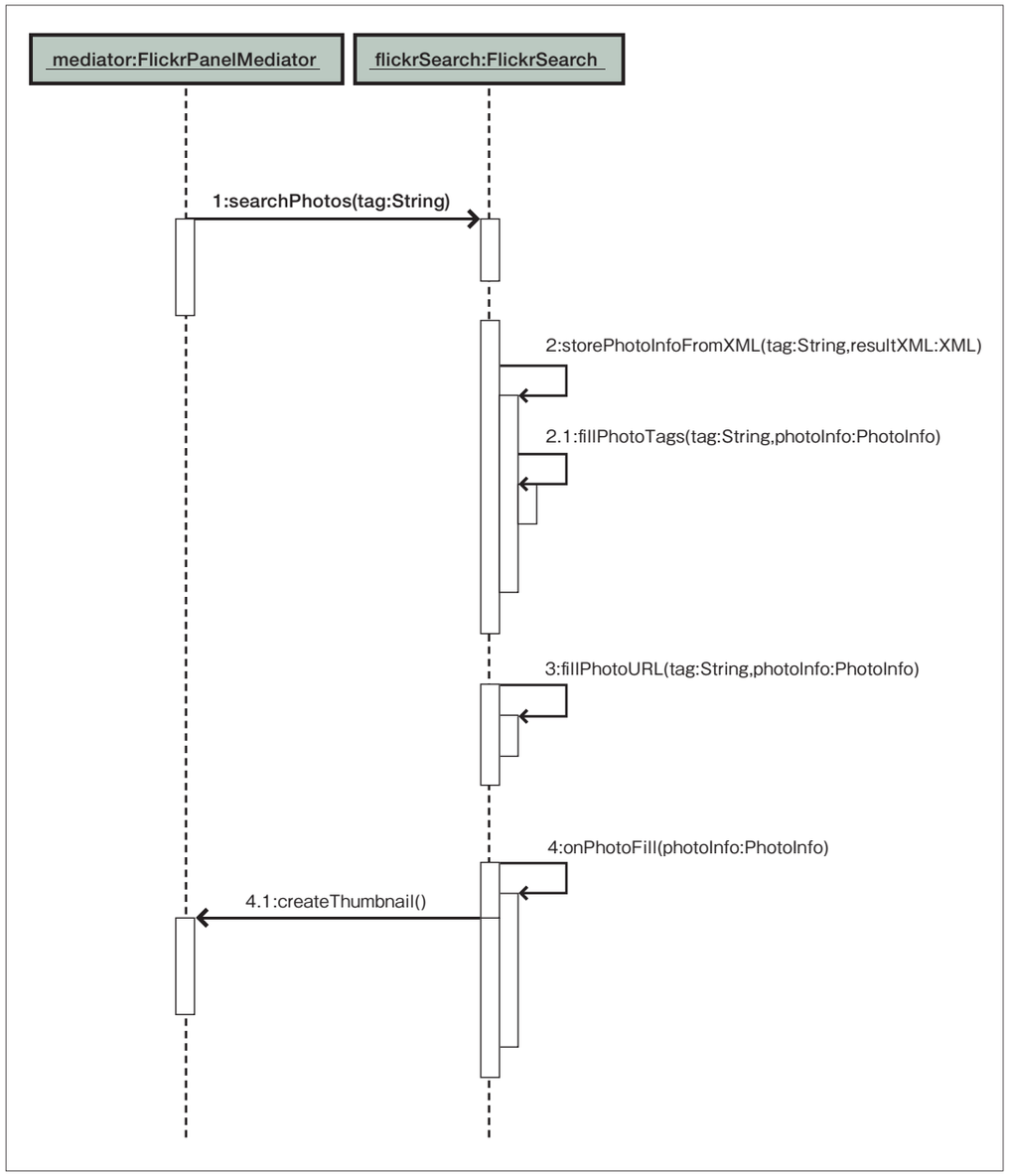
しかし、これだけでもまだ写真情報はそろいません。FlickrSearch.fillPhotoURL()を呼び出してURLを補完します。

SOURCE FlickrSearch.fillPhotoURL() メソッド

```
// FlickrConnector.onTagsByPhotoFind イベントハンドラを上書きし、
// 写真情報のタグを補完する。
flickrConnector.onURLByPhotoFind = function(resultXML) {
          .
          .
          .
     while (currentNode != null) {
          photoInfo.thumbnailURL = currentNode.attributes['source'];
          photoInfo.imageURL = currentNode.attributes['source'];

          currentNode = currentNode.nextSibling;
     }


     // 持っているタグに属するようにする
     tagHolder.fillTagFromPhotoInfo(photoInfo);


     // 1 つの画像情報が整ったイベントハンドラを発行
     target.onPhotoFill(photoInfo);
}

// 写真 ID から画像の URL 情報を取得する
flickrConnector.searchURLByPhoto(photoInfo.id);
```

ここでも、FlickrConnector.onURLByPhotoFind() イベントハンドラを上書きし、写真情報のサムネイルの URL と大きな画像の URL を埋めていきます。そして、最後にこの画像の情報がすべてそろったというイベントハンドラをたたいて、1 つの写真の情報の取得が完了します。

FlickrPanelMediator クラスと FlickrSearch.searchPhotos() メソッド



○ 画像情報の読み込み完了の際の処理

画像情報がそろえば、あとはサムネイルをロードする作業です。しかしロードしたサムネイルを指定のタグに集まるようにすればいいかというと、少し違います。それは、すでにユーザーが検索しているタグが変わっている可能性があるからです。その処理の流れを見てみます。

画像の情報がそろうと、FlickrSearch.onPhotoFill イベントハンドラが呼ばれます。このメソッドは、FlickrPanelMediator.createThumbnail で上書きされています。その中で FlickrPanel にサムネイルをロードするよう命令を出します。

FlickrPanelはサムネイルとなる MC（Thumbnail) をステージに attachMovie() させて、サムネイルの画像を呼び出します。このサムネイルの読み込みが終わったことを告げる Thumbnail.onThumbnailInit を受けて、FlickrPanel.initThumbnail メソッドが呼ばれます。

その中では、ロードした Thumbnail が現在検索されているタグを持っているかをチェックして、持っていなければ即座に removeMovieClip を行っています。

SOURCE FlickrPanel.initThumbnail メソッド

```
        // 現在のターゲットのタグに属してなければ fadeOut させる
        var tagHolder = TagHolder.getInstance();

        if (!tagHolder.doesPhotoInfoBelongTo(this.targetPanel.getTag(),
                                                    thumbnail.photoInfo))
            thumbnail.removeMovieClip();
```

こういったチェックを行うことにより、読み込み開始から準備完了までのタイムラグでの状態の変化をうまくコントロールしています。