

An Introduction to Object- Oriented Concepts Using Popfly

Think about your cell phone and how you might describe it to someone else. It has characteristics (such as manufacturer and model number); there are things it knows how to do (such as dial a number, or set a ring tone); your friend might have the same cell phone as yours, yet the information you store in them makes them different. Each of these describes part of the cell phone's existence as an object in the world.

This lesson introduces basic object oriented concepts within the context of building Popfly mashups to interact with structured data, and also to combine data from different objects together.



Prepared by Mark Frydenberg
Computer Information Systems Department
Bentley College, Waltham, MA
mfrydenberg@bentley.edu

An Introduction to Object-Oriented Concepts Using Popfly



Professor Popfly Mashups Referenced in this Lesson:

- States Starting with M – 1 (<http://www.popfly.com/users/professorpopfly/MStates-1>)
- States Starting with M – 2 (<http://www.popfly.com/users/professorpopfly/MStates-2>)
- Red and Yellow (<http://www.popfly.com/users/professorpopfly/RedAndYellow>)
- My NewsReader (<http://www.popfly.com/users/professorpopfly/MyNewsReader>)



Learning Outcomes

After completing this lesson, you should be able to:

- Relate object oriented modeling concepts of classes, objects, methods, parameters, instances, and properties to Popfly blocks and mashups
- Access data from a Popfly Data Block
- Use the Combine block to combine objects or items from two different Popfly blocks
- Investigate a Popfly block's "entire object" output
- Build a mash up combining two news feeds

Overview

Think about your cell phone and how you might describe it to someone else. It has characteristics (such as manufacturer and model number); there are things it knows how to do (such as dial a number, or set a ring tone); your friend might have the same cell phone as yours, yet the information you store in them makes them different.

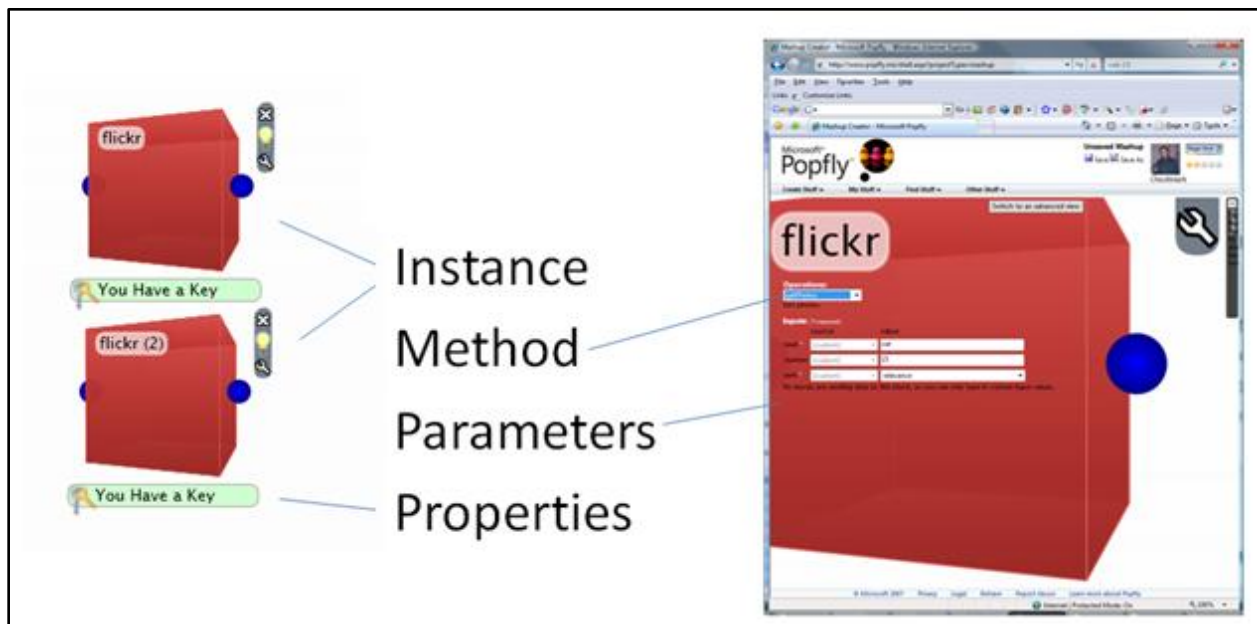
Some actions that the phone "knows how to do" may require additional information in order to accomplish the task (for example, supplying the phone number to dial, or the name of the sound for the ring tone.) Each of these items describes part of the cell phone's existence in the world.

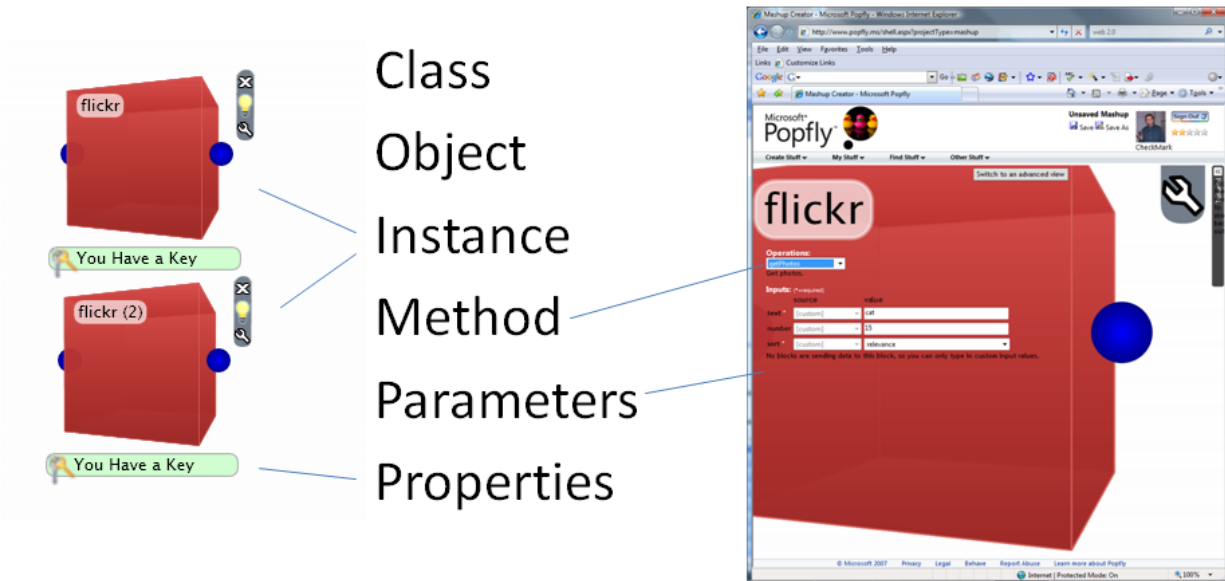
This lesson introduces basic object oriented concepts within the context of building Popfly mashups. You will build mashups to interact with entire objects and parts of them, and also to combine data from objects together.

Living in an Object Oriented World

Objects are ways to model real things in the real world. A *class* is a description of a particular object. For example, you might fully describe a class called CellPhone by specifying its characteristics (or *properties*) such as manufacturer and model number, its *methods* (DialNumber and SetRingTone) and their corresponding *parameters* (inputs): DialNumber requires a phone number parameter to dial; SetRingTone requires the name of an audio file containing the ring tone. Your cell phone and my cell phone each are *instances* of the CellPhone class.

Popfly provides a natural context for introducing object-oriented concepts. Many blocks are objects with methods that describe their functionality and properties to describe their behavior. Some blocks return objects containing a set of related values (such as information about a flickr photograph).

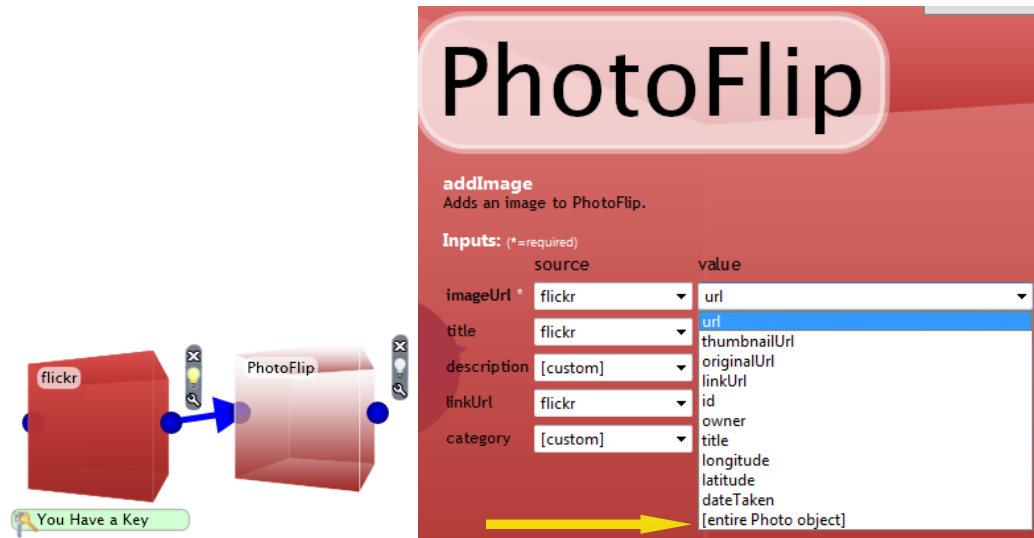




In this example, there are two instances of the flickr block. Each flickr block has a Developer Key property; the method (or operation) named *getPhotos* has three parameters (or inputs): text describing the photos to obtain, number of photos to obtain, and how they should be sorted (by relevance, date taken, interestingness, etc.)

Parameters are values that you *pass* to an operation in Popfly. In object-oriented programming, *method* is the word used to describe an operation—something that an object knows how to do. Tweaking a mashup lets you change the inputs to the methods of any of the blocks without having to go into the Mashup editor, where you would need to click the wrench on each block, and specify new input values. Changing the parameters changes the behavior of the method, and therefore the output of the mashup.

Some Popfly blocks also return objects as their output. For example, the flickr block returns a Photo object containing information about a flickr photo.



As shown above, one way to see the information that a block outputs is to connect that block to another, and then look at the items available for that block's entire object. Here, the flickr block is connected to the PhotoFlip block, and the value dropdown list in the PhotoFlip block shows the items available in the entire Photo object that the flickr block outputs for each flickr photo.

For debugging, sometimes the BlockOutputInspector block is helpful in viewing the data in the object that a block returns. Connect the BlockOutputInspector block to the block in question and run the mashup.

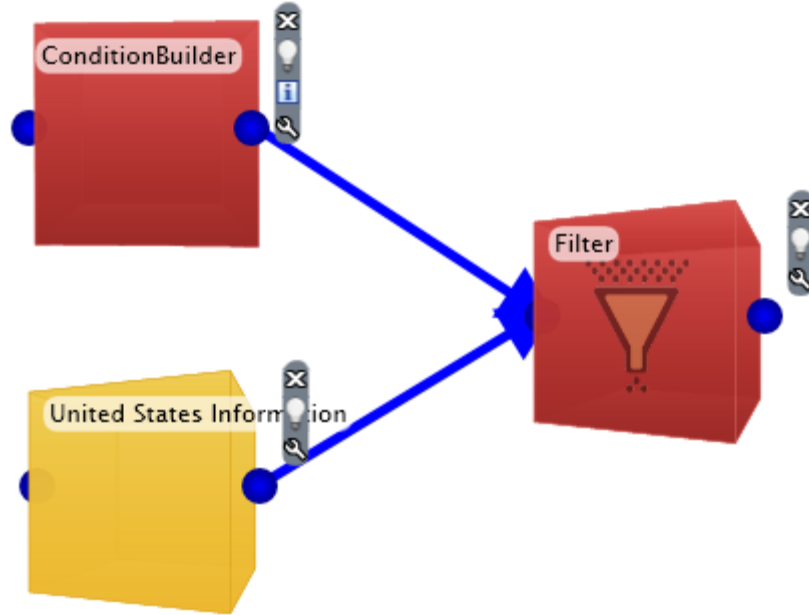
Example 1: Same Blocks; Different Inputs Give Different Results

The next two examples show mashups that search the United States Information data block for states that begin with the letter M.

Both mashups use the same blocks, but depending on the inputs and outputs specified for the condition builder and the Filter blocks, you can get different results. This mashup obtains state information from the United States Information block, and then uses the ConditionBuilder block to set up a condition by which to filter it. See the Logic lesson for more information about using the ConditionBuilder block.

The output of the *MStates-1* mashup is simply a list of state names that begin with the letter M.

The output of the *MStates-2* mashup is a list of objects from the United States Information block for each state that begins with the letter M.



The United States Information block returns a list of DataVal objects, each of which contains information about a state. A portion of the first three DataVal objects returned are shown here:

Alabama	Montgomery	Yellowhammer State	Camellia	...
Alaska	Juneau	The Last Frontier	Forget Me Not	...
Arizona	Phoenix	The Grand Canyon State	Saguaro Cactus Blossom	...

In the example below (MStates-1), the Filter block sets the list to filter on as the list of State values from the United States Information block. Therefore, the `x` as the comparison variable in ConditionBuilder must refer to a single state in that list. The condition `(x [begins with] M)` in conjunction with the Filter list as the list of State names from the United States Information block, indicates that for each item in the list obtained from United States Information block, Popfly will only look at a single value, namely the State name, in order to perform the Filter.

ConditionBuilder

Operations:
FilterCondition
Build a condition for the Filter or RSS Filter block.

Inputs: (*=required)

comparisonVariable * [custom] x

condition * [custom] begins with

comparisonValue * [custom] M

stringComparison * [custom] true

No blocks are sending data to this block, so you can only type in custom input values.

OK

Filter

Operations:
filter
Filters the input list based on an arbitrary condition.

Inputs: (*=required)

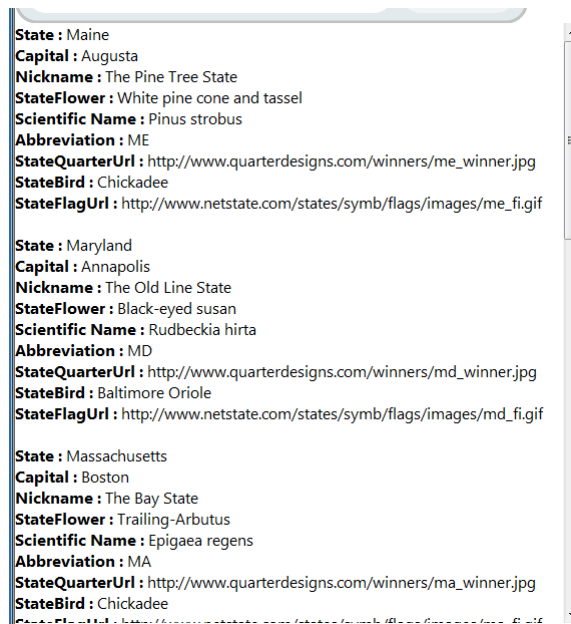
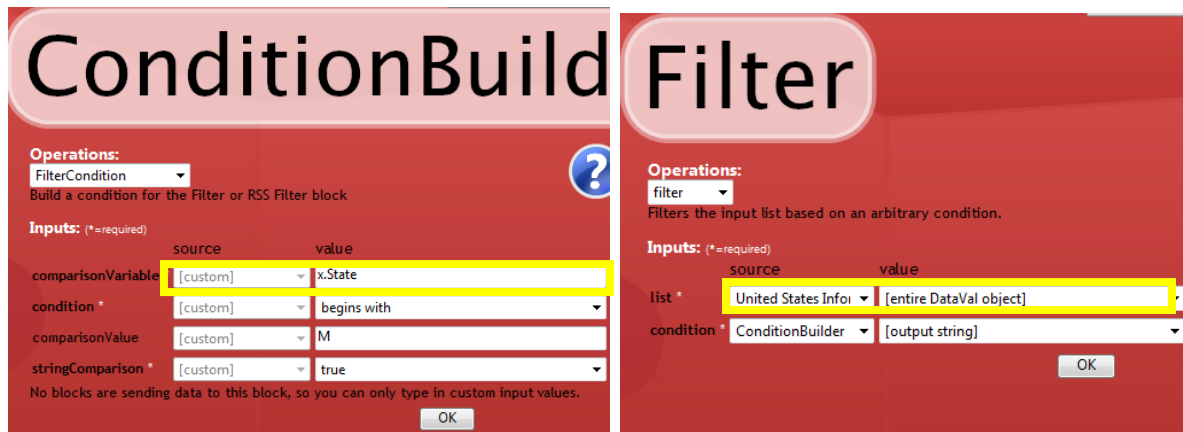
list * United States Info State

condition ConditionBuilder [output string]

OK

The output of the *States MStates-2* mashup is all of the information (state, capital, nickname, state flower, etc.) from the United States Information block for each of the states that starts with the letter M. To accomplish this, the Filter block must use the list of all of the entire DataVal objects returned from the United States Information block to filter on. Therefore, the ConditionBuilder block uses `x.State` as the comparison variable to indicate that the filter condition will examine the State field of each object returned from the United States Information block. In this case, the `x` as the comparison variable in ConditionBuilder refers to the entire object from the United States Information block, and `x.State` indicates the value of the State field of that object. The filter block filters on the State field but has all of the other fields available.

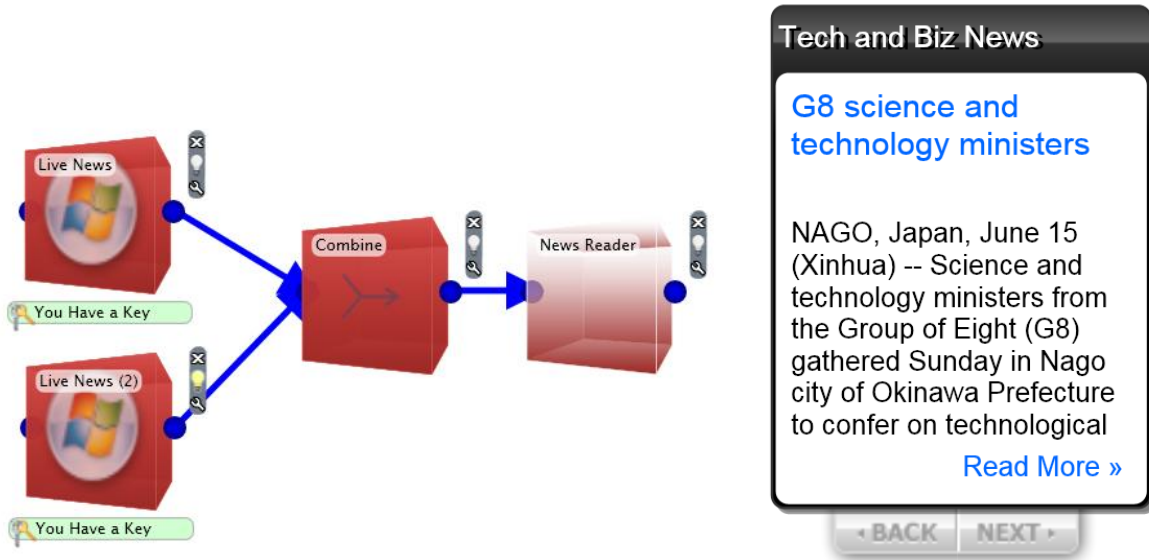
Note that the syntax `<objectName>.<fieldName>` is a common way in many programming languages to specify a field or element from an object.



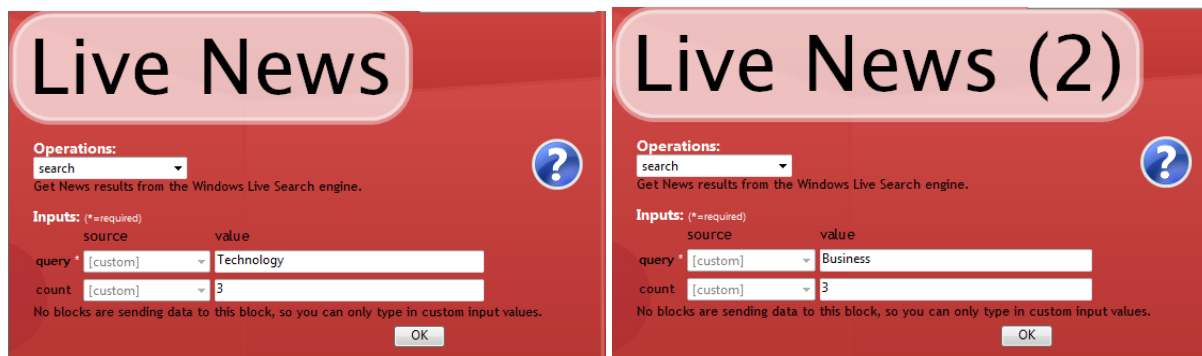
Example 2: Combining Objects of the Same Type

Popfly's Combine block is useful when the objects being combined come from different instances of the same block, for example, combining images from two flickr blocks, or combining the results of two news feeds.

This example uses the Combine block to create a customized news reader.



To construct this mashup, drag two instances of the Live News block onto the design surface. Specify a topic of interest as the query for each block's search method. This example combines both technology and business news.



When combining the two Live News blocks, select the entire LiveNewsResults object to be combined. This way, all of the fields of the LiveNewsResults object will be available to the News Reader block.



Example 3: Combining Properties from Objects of Different Types

You might also use the Combine block to create a new list that combines individual items (often of the same type) from different blocks. For example, Professor Popfly's RedAndYellow mashup combines three flickr photos tagged with "red" and three Live Image Search photos tagged with "yellow." Note that the Combine block combines the originalUrl from flickr with the mediaUrl from Live Image Search to produce the results.

Popfly PopQuiz

1. Think about some real-world “object” that you interact with every day, such as a car, a portable music player, or a coat. What are its properties? What are its methods? What parameters might its methods have? Describe two different instances of your object.
2. When should you specify the entire object from another block vs. only one of its fields (columns) as input to an operation in a Popfly block?
3. What can a Combine block combine?
4. Name a time when you might want more than one instance of the same block in a mashup.
5. Name two ways to find out what values are in the object that a block returns.

Try It in Popfly

Modify or build these Popfly mashups to demonstrate your understanding of this lesson. The more ducks, the bigger the challenge.



Using the United States Information block, create two mashups to determine which states have the Mockingbird as their state bird.

First, display only the names of the states that have the Mockingbird as their state bird.

Second, display all of the information about the states whose bird is Mockingbird.



Create a mashup to combine images from the same image provider. (For example, select dogs and cats from Live Image Search). Display them using your favorite Popfly display block. Use the Combine block to combine the images into a new list.











Recreate the same mashup to combine images from two different image provider blocks (for example, select dogs from Flickr, and cats from Live Image Search). Display them using your favorite Popfly display block.



Using the United States Information block, create a mashup that displays a three column table containing the state name, the name of the state bird, and a picture of the bird, for all states that begin with the letter M.

Hint: Use the SimpleImage block to display the image URL as an image. The output should look like this:

State	Bird	Photo
Maine	Chickadee	
Maryland	Baltimore Oriole	
Massachusetts	Chickadee	
Michigan	Robin	
Minnesota	Common Loon	
Mississippi	Mockingbird	
Missouri	Bluebird	
Montana	Western Meadowlark	

Learn More about Objects

- Wikipedia on Object Oriented Programming
http://en.wikipedia.org/wiki/Object_oriented
- Popfly and Object Oriented Programming
<http://www.guardian.co.uk/technology/2007/oct/21/popfly>

This blog post describes Popfly as a simple object-oriented approach to creating mashups.