

BIRDBASE / 2011

VISH VISHVANATH BIRDBASE

The Book Of Configuration

Spring 2011

TODO LIST

## CONTENTS

---

1	INTRODUCTION: WHAT IS CONFIGURATION?	1
2	A SAMPLE YAML CONFIGURATION	2
3	ACCESSING THE CONFIGURATION FROM ACTIONSCRIPT	4
4	ACCESSING AND MANIPULATING STRINGS IN ACTIONSCRIPT	5
4.1	Using the TextService	5
4.1.1	Registering an appropriate class	5
4.1.2	Using straight text	6
4.1.3	Sprintf replacement in strings	6
4.2	NOT using the TextService	6
5	SUMMARY	7
5.1	Arbitrary variables	7
5.2	Strings	7

## INTRODUCTION: WHAT IS CONFIGURATION?

---

Runtime configuration is, basically, a bunch of keys and values that the SWF loads at bootup. These keys could represent:

- Strings - text used in the site.
- Navigation - a list of items for a menu.
- URLs - that we want to launch in the site or read RSS feeds from, or load media.
- Layout information - x/y coordinates, alpha, etc., so display is configurable.

Clients often want to tailor sites, especially marketing websites, to a variety of locales. This will obviously mean translating the strings, but it may mean turning chapters off in some countries or customizing certain graphics for other countries. Runtime configuration might be a text file, but it can be generated by ASP/PHP/Ruby/Java/Whatever, and a CMS can manage it.

## A SAMPLE YAML CONFIGURATION

I have chosen YAML for the configuration because it's human-readable, processable in a single-pass, and was designed explicitly for data-serialization. It's also lightweight in file size.

A sample YAML configuration file, from the demo application:

```
# --- Start Navigator Configuration ---

# The lines below map between Actionscript classes and URL states.
# the "destination" maps an array of URL routes that should cause the
# "view" class to display when navigated to.

nav:
  - item:
      destination: [ "home", "home/*" ]
      label:      Home Section
      view:      HomeView
  - item:
      destination: [ "media" ]
      label:      Media Section
      view:      MediaView

# --- Begin strings configuration ---
# A key, such as "sound_on" maps to the translated string. You may notice %s
# in some of the strings. This allows for dynamic string replacement in the
# application. Say you wish to include the user's name in a piece of text. You can
# register a Text component with the dynamic string and inject extra values to
# replace the dynamic bits.

strings:
  sound_on:      "on"
  sound_off:     "off"
  tagline:       "This is a tagline, right here. Good Morning %s!"
  newsletter_copy: "Pellentesque nibh felis, eleifend id, commodo in, interdum
  vitae, leo. Praesent eu elit. Ut eu ligula. Class aptent taciti sociosqu ad
  litora torquent per conubia nostra, per inceptos hymenaeos. Pellentesque
  nibh felis, eleifend id, commodo in, interdum vitae, leo. Praesent eu elit.
  Ut eu ligula. Class aptent taciti sociosqu ad litora torquent per conubia
  nostra, per inceptos hymenaeos."

# --- End strings configuration ---

# --- Begin application configuration ---

root:      "home"                # the home page
base:      "assets/en_GB/"       # the base URL for the application
fontsFile: "fonts/fonts.swf"    # the fonts SWF

# These assets below are loaded at bootup time.

assets:
  - item:
      key: "dynamic_library"
```

```

    uri: "assets.swf"           # The dynamic library SWF
  - item:
    key: "background"
    uri: "images/main_background.jpg"
  - item:
    key: "foreground"
    uri: "images/main_foreground.jpg"

# Some entirely arbitrary data for your app

rss_feeds:
  news:
    uri: "http://www.example.com/news.rss"
  video:
    uri: "http://www.example.com/video.rss"
  media:
    #screenshots
    - item:
      uri: "http://www.example.com/screenshots.rss"
    #wallpapers
    - item:
      uri: "http://www.example.com/wallpapers.rss"
    #artwork
    - item:
      uri: "http://www.example.com/artwork.rss"
layout_data:
  - item:
    id:    graphic_one
    x:    190
    y:    320
  - item:
    id:    graphic_two
    x:    390
    y:    170

```

ACCESSING THE CONFIGURATION FROM  
ACTIONSRIPT

---

Take the following chapter of configuration:

```
base:          "assets/en_GB/"          # the base URL for the application

rss_feeds:
  news:
    uri: "http://www.example.com/news.rss"
  video:
    uri: "http://www.example.com/video.rss"
  media:
    #screenshots
    - item:
      uri: "http://www.example.com/screenshots.rss"
    #wallpapers
    - item:
      uri: "http://www.example.com/wallpapers.rss"
    #artwork
    - item:
      uri: "http://www.example.com/artwork.rss"
```

We will access this with the Settings class.

```
[Inject]
public var settings:Settings;

var base:String = settings.getSetting( "base" );
trace( base );
// Prints out "assets/en_GB/"

var news:String = settings.getSetting( "rss_feeds" ).news.uri;
trace( news );
// Prints out "http://www.example.com/news.rss"

var media:Array = settings.getSetting( "rss_feeds" ).media;
trace( media[ 1 ].item.uri );
// Prints out "http://www.example.com/wallpapers.rss"
```

## ACCESSING AND MANIPULATING STRINGS IN ACTIONSRIPT

---

### 4.1 USING THE TEXTSERVICE

You can register any object that implements `IHaveUpdateableText` with the `TextService` class. You register the string name that you want pushed to this object, along with any variables that should be injected into the `%s` points, and the `TextService` will do the the rest by automatically updating your text object.

The `TextService` supports runtime dynamic reloading of strings, so if these are reloaded, in say, a different language, all your `IHaveUpdateableText` instances will be reformatted and updated immediately.

Take the following strings in the configuration:

```
strings:
  sound_on:      "on"
  sound_off:     "off"
  tagline:       "This is a tagline, right here. Good Morning %s!"
  newsletter_copy: "Pellentesque nibh felis, eleifend id, commodo in, interdum
  vitae, leo. Praesent eu elit. Ut eu ligula. Class aptent taciti sociosqu ad
  litora torquent per conubia nostra, per inceptos hymenaeos. Pellentesque
  nibh felis, eleifend id, commodo in, interdum vitae, leo. Praesent eu elit.
  Ut eu ligula. Class aptent taciti sociosqu ad litora torquent per conubia
  nostra, per inceptos hymenaeos."
```

#### 4.1.1 Registering an appropriate class

Use a class like this one:

```
package org.birdbase.framework.view
{
  import flash.text.TextField;
  import org.birdbase.framework.service.text.IHaveUpdateableText;

  public class UpdateableTextField extends TextField implements IHaveUpdateableText
  {
    public function UpdateableTextField()
    {
      super();
    }

    public function setText( value:String ):void
    {
      this.text = value;
    }
  }
}
```

#### 4.1.2 Using straight text

Here's the copy you'd like to use from the above configuration:

```
newsletter_copy: "Pellentesque nibh felis, eleifend id, commodo in, interdum vitae, leo. Praesent eu elit. Ut eu ligula. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Pellentesque nibh felis, eleifend id, commodo in, interdum vitae, leo. Praesent eu elit. Ut eu ligula. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos."
```

To automatically populate the newsletter copy field once the strings are loaded, try this, using the UpdateableTextField class shown previously.

```
[Inject]
public var textService:ITextService;

var newsletter_text:IHaveUpdateableText = new UpdateableTextField();
textService.register( newsletter_text, "newsletter_copy" );
```

#### 4.1.3 sprintf replacement in strings

But you may have noticed that the "tagline" copy contains a %s.

```
tagline: "This is a tagline, right here. Good Morning %s!"
```

To replace that with the user's name, try this code:

```
[Inject]
public var textService:ITextService;

var tagline_field:IHaveUpdateableText = new UpdateableTextField();
textService.register( tagline_field, "tagline", "Gil Scott-Heron" );

// which results in a textfield containing the string:
// "This is a tagline, right here. Good Morning Gil Scott-Heron!"
```

## 4.2 NOT USING THE TEXTSERVICE

As the strings are part of the YAML config, they can be accessed through the settings too.

So if you can't be bothered with all the TextService stuff, simply do this to access the newsletter\_copy above:

```
[Inject]
public var textService:ITextService;

var copy:String = settings.getSetting( "strings" ).newsletter_copy;
```

Done.

# 5

## SUMMARY

---

### 5.1 ARBITRARY VARIABLES

You can place anything you like into the configuration and retrieve it as a String, Array of Strings or a Dictionary. See the YAML documentation at either [yaml.org](http://yaml.org) or on Wikipedia for explanations on how data is represented in YAML.

### 5.2 STRINGS

The TextService provides dynamic updating of your textfields, but strings can be pulled out of the configuration just like other settings too.

The TextService provides helpers for Sprintf string replacement, else you can roll your own.

