SPL Quick Start Guide

Last modified Monday, January 16, 2012

TextField and TLF

SPL 2.0 contains two sets of components for Flash and Flex: one that only supports spellchecking for TextField (and components that use TextField), and one that supports both TextField and TLF-based components, such as the Flex 4 "Spark" components.

Note: When targeting Flash player versions prior to 10, you must use the TextField SPL components, or you will get compile time errors.

Please ensure that you choose the correct components when building your application. For information on using the source code to develop with SPL, please read the "Plugins and Factories" section.

Flash Professional Components

To install the SPL components for Flash Professional (CS3 or higher), open the "Spelling Plus Library.mxp" file located in the "components > Flash" folder. This should launch the Adobe extension manager will install the components into Flash automatically.

To manually install them, copy the full "Spelling Plus Library - TextField" and "Spelling Plus Library - TLF" (CS5 only) directories to the components directory of the Flash configuration directory at:

Windows XP: `C:\Documents and Settings\<username>\Local Settings\Application Data\Adobe\Flash <CS#>\<language>\Configuration`

Mac OSX: `Macintosh HD:Users:<username>:Library:Application Support:Adobe:Flash <CS#>:<language>:Configuration`

Restart Flash, or select the "Reload Components" option in the components panel drop-down menu, create a new AS3 FLA, and open the component panel. You will see new folders containing all of the SPL components named "Spelling Plus Library - TextField" and "Spelling Plus Library - TLF".

Component List

The Flash components provide a simple visual method for working with SPL in the Flash Professional IDE. It is not recommended to use the Flash component classes when working directly in ActionScript (see the API documentation for more details on working with SPL via code).

SPLAllClasses
Add a single instance of this component to your FLA to import all of the SPL classes. This will allow you to work with SPL directly via code. See the API documentation for a listing of all classes and their methods. You do not need to include this component if you are using the SPLTag component.

SPLCoreClasses
Add a single instance of this component to your FLA to import only the SpellingDictionary class and its dependencies. This will allow you to work with core SPL functionality such as spell checking and spelling suggestions directly via code, without importing the any of the secondary functionality like word list loading, custom LSO wordlists, or text field integration. You can combine this with SPLWordListLoader and SPLCustomWordListLSO to import other functionality. See the API documentation for a listing of all classes and their methods.

SPLCustomWordListLSO
Add a single instance of this component to your FLA to automatically set up custom word list saving to Local Shared Object. Whenever the user modifies their custom word list, it will be saved back to an LSO with the name specified by the "lsoName" component parameter. This component abstracts and simplifies the set up of CustomWordListLSO, which provides more options.

Note: This will only save word lists for a single user. In order to provide a "shared" word list between users, you will have to handle the custom dictionary events manually, and store the words on a server.

SPLTag
Dragging this component onto a text field or component will cause it to snap to the component. At run time, it will set up a SpellingHighlighter instance on its target to provide real-time spell checking. This component abstracts and simplifies the set up of SpellingHighlighter, which provides many more options through code.

SPLWordListLoader
Add a single instance of this component to your FLA to automatically load a word list file at the url specified by the "url" component parameter, and assign it to the SpellingDictionary.. This component abstracts and simplifies the set up of WordListLoader, which provides more options through code.

Test the components in Flash

- Create a new ActionScript 3 FLA
- Place an input textfield on stage. If using CS5, note if it is a TLF or Classic TextField.

- Drag a SPLTag component on top of the text field (it should snap to the top left corner). Ensure that you drag the correct component type: if you are using a TLF TextField, use the TLF component set.
- Drag a SPLWordListLoader component on stage, and set the url to point at a word list file (ex. wordlists/en_us.zip). Note that you will need to save your FLA to use a relative path.
- Test movie. Start typing misspelled words into the text field. Once the word list loads and parses (usually a couple of seconds) misspelled words will be underlined.

Flex Components

To use the Flex SWC, add it as a library to your Flex project. To do this, add the SWC to your project's "libs" folder. Alternately, you can reference it by doing the following:

1. Open your project properties (accessible by right clicking on your project in the project pane and selecting properties)
2. Click on Flex Build Path > Library Path tab and click the "Add SWC..." button
3. Select the SWC to add it to your project.

You can now access all of the classes in the SPL library directly (see the documentation for a listing).

Component List

Provided with SPL are a set of MXML tags analogous to the Flash components outlined above. To quickly set up basic spell checking, use the following code:

```
<spelling:SPLWordListLoader url="wordlists/en_us.zlib" />
<spelling:SPLCustomWordListLSO lsoName="customDictionary" />
<spelling:SPLTagFlex target="{myTextFieldOrComponent}" />
```

The Flex components abstract the core SPL classes, and are set up to work properly when instantiated using MXML. Do not use the Flex components when working directly in ActionScript. See the API documentation for more information on working with SPL in code.

SPLCustomWordListLSOFlex

Add a single instance of this component to your application to automatically set up custom word list saving to Local Shared Object. Whenever the user modifies their custom word list, it will be saved back to an LSO with the name specified by the "lsoName" component parameter. This component abstracts and simplifies the set up of CustomWordListLSO, which provides more options through code.

Note: This will only save word lists for a single user. In order to provide a "shared" word list between users, you will have to handle the custom dictionary events manually, and store the words on a server.

SPLTagFlex
Add an instance of this component to your application for each component you want to add spell-checking to. At run time, it will manage setting up a SpellingHighlighter instance on the component to provide real-time spell checking. This component abstracts and simplifies the set up of SpellingHighlighter, which provides many more options through code.

Note: SPLTagFlex defaults the `updateOnValueCommit` property to `true` for its SpellingHighlighter instance.

SPLWordListLoaderFlex
Add a single instance of this component to your application to automatically load a word list file at the url specified by the "url" component parameter, and assign it to the SpellingDictionary. This component abstracts and simplifies the set up of WordListLoader, which provides more options through code.

Test the SWC in Flex

1. Create a new MXML file, and insert the following code (ensure the url is pointing to a word list file):

```
<spelling:SPLWordListLoader url="wordlists/en_us.zlib" />
<mx:TextArea id="myTextArea" />
<spelling:SPLTagFlex target="{myTextArea}" />
```

2. Test your MXML file. Start typing misspelled words into the text area. Once the word list loads and parses (usually a couple of seconds) misspelled words will be underlined.

Important note on Rich Text Editor:
To apply spell checking properly to a Flex RichTextEditor component, you must target the TextArea instance within it. In order for the highlighter to update appropriately based on text formatting changes, you must also add a simple change event handler to the RichTextEditor.

```
<mx:RichTextEditor id="myRTE"
change="{ splTagHighlighter.spellingHighlighter.update() } />
<spelling:SPLTagFlex id="splTagHighlighter" target="{myRTE.textArea}" />
```

Source code

If you purchased a source code license, you can also use the source code directly. Simply copy the class files into the class path for your project (with packages intact), and work with them as you would any other class in your project. It is recommended that you always work with a copy of the original source code if you intend to modify it, so that it is easier to revert if you encounter problems.

Note that gskinner.com does not provide support for modified versions of the source code.

Plugins and Factories

In order to maintain a single core engine, and support both traditional TextField-based text display as well as the Text Layout Framework (TLF) text fields and components introduced in Flex 4 and Flash Player 10, some of the core functionality in SPL 2.0 related to spell checking and text highlighting has been abstracted to plugins. The APIs to interact with SPL are largely the same as in SPL 1.x.

By default, all the SWCs are compiled using the correct plugin factories, however there is an initial step to specify the appropriate plugin factories when using the source code.

1. **Using the PluginFactoryDefaults**
   Along with each Flash and Flex TextField and TLF component sets, a "PluginFactoryDefaults.as" class is included, located in `com.gskinner.text` package. Each file has been set up to register the correct plugin factories with SPL. Simply include the respective class with your project. Note that you can specify multiple plugin factories, and SPL will use the first one it finds that works with the target component. The default TLF-based factories use the TLF plugin first, followed by the TextField plugin.

   ```
   public static var factories:Array = [new SpellingTextFieldPluginFactory
   (), ...otherFactories];
   ```

2. **Specifying a factory manually**
   When creating a SpellingHighlighter, you can specify a factory in the constructor.

   ```
   new SpellingHighlighter(target, dictionary:SpellingDictionary, [new
   SpellingTextFieldPluginFactory(), ...otherFactories]);
   ```

SPL ships with three sets of TextHighlighter plugins, each with a respective factory. Also included are SpellingHighlighter versions of each plugin and plugin factory.

- TextField for use with Flex 3, Flash Professional CS3, or any classic TextField or TextField-based component.
- TLF for use with Flex 4, Flash Professional CS4 and CS5, or any TLF-based text field or component.
- Flex 4 "spark" components, or instances of RichEditableText.

Compatibility

To ensure that earlier versions of Flash or Flex Builder will compile, multiple versions of the SPL SWCs are provided. The included TextField-based SWCs are compatible with Flash CS3 and later, and Flex 3.0 and later. The new TLF-based SWCs are only compatible with Flash CS5, and Flex 4.1 and later. All sample FLAs made with TLF will only open in Flash CS5.

Note: Specific SWCs are available for Flex 2.01. Due to minimal demand for these SWCs and to keep the size of the SPL package to a minimum we do not distribute them by default. Please contact us via our support form to request these SWCs.

Documentation

The SPL Specifications document contains important information on a variety of topics, including word list formats, performance, capitalization, AIR support and languages other than English. Full API documentation can be accessed by opening the index.html file in the API directory.