SPL 2.0, Specifications

Last updated Friday, April 29, 2011

Overview

This document covers some of the specifications and expected behaviors of the gskinner.com Spelling Plus Library (SPL).

Word Lists and Dictionaries

The APIs and documentation make a distinction between word lists and dictionaries. A word list is simply an array of word strings. A dictionary is an instance of SpellingDictionary (or a subclass). A dictionary composes a word list. Please see the "Languages Other Than English" section of this document for information on multi-lingual support.

Loading Word Lists and Deferred Initialization

On initialization, SpellingHighlighter instances will check to ensure that the SpellingDictionary has a master word list assigned prior (via SpellingDictionary.active). If not, they will subscribe to the SpellingDictionary "active" event, and disable themselves until the dictionary is active.

Word List Formats

The SpellingDictionary class requires that word lists are passed in as a sorted (ascending case-insensitive alphanumeric) array of word strings. The WordListLoader class manages loading and converting word list files into this format.

The WordListLoader class supports two word list file formats: Plain text and GSPL. A plain text word list is a sorted list of words (ascending case-insensitive alphanumeric) delimited by line feed characters and encoded in UTF-8. The GSPL format allows for a simple compression scheme whereby prefixes and suffixes are defined at the beginning of the file, designated by a PFX or SFX code. Each of these affixes is assigned a bit position in the order they appear in the file. These bits are combined into a single base 32 integer and appended with a leading "/" after words to indicate which affixes can be applied to create derivative words. These affixes are not combined – only one affix is applied to the base word at a time. This format results in file savings of up to 65%, but due to the high CPU requirements of parsing and sorting the list it can result in a 2-4 second pause when the list is parsed. This pause is mitigated by using polite parsing, whereby the system will parse the data in small chunks spread over a number of frames. This slightly increases total parsing time, but reduces or eliminates problems with choppy animation during parsing. See WordListLoader.maximumParseTime in the documentation for more information.

As an example, a GSPL file containing the following:
PFX re

SFX s
donut/2
run/2
write/1
fun

Would be parsed into the following word list:
donut
donuts
fun
rerun
rewrite
run
runs
write

Note that the word “reruns” is not created (the combination of the “re” prefix and “s” suffix). It would have to be listed separately in the GSPL file. Also note that the results are sorted.

For forwards compatibility, the WordListLoader will ignore any lines prefixed with “/”, and will ignore all lines including and following a line beginning with “//”. This will allow for the addition of meta data in the future.

** You can use any other word list format by writing your own loader/parser that returns a sorted array of words that can be passed to a dictionary.*

Compressed Word Lists

WordListLoader also supports compressed word list files (in either format). Compressing the word list can save 60% or more in file size. For example an American English word list with over 150,000 words in compressed GSPL format is less than 250kb. The files must use zlib compression compatible with Flash Player 9’s `ByteArray.uncompress` method.

Choosing a Word List Format

Word lists are provided in 4 formats, as explained above:

1. Uncompressed word list
2. Uncompressed GSPL format
3. zlib compressed word list
4. zlib compressed GSPL format

Selecting a word list format generally involves balancing file size against parsing time. Because uncompressing the word lists does not have a major impact on parsing time, your choices will usually be between the two compressed formats. The first two uncompressed formats are provided mainly for reference.

The standard compressed word list is about 60% larger than the compressed GSPL format, but it will parse in about 100ms, versus 2-3 seconds. The longer parse time for GSPL is alleviated somewhat by the ability to do polite background parsing (to avoid application stalling or animation stuttering).

If you are targeting the desktop (with AIR for example), or high bandwidth online users, you may want to choose the compressed word list format. If you are targeting lower bandwidth users, or are seeking to reduce server load, you may want to use the compressed GSPL format.

404 Errors With Wordlists on IIS

Some servers (ex. IIS) will not serve files with unknown file extensions such as ".gspl". This can lead to SPL being unable to load the wordlist, and never enabling. Simply change the file extension to .zip or .txt to resolve this issue (SPL does not require any specific file extension).

Wordlist Editor

The SPL package includes the gWordListEditor AIR application that makes importing, editing, exporting, and merging word lists simple. Note that you must have the release version of the AIR runtime installed to use this application. You can download and install the AIR runtime at labs.adobe.com.

Some of the features of gWordListEditor include:

- Import & Export of .txt and .gspl files, in both compressed and uncompressed formats
- Drag & Drop support of .txt & .gspl files
- Add and remove words from your custom word lists
- Merge word lists (simply import multiple lists to merge them)
- Specify prefixes and suffixes that your word list uses, to decrease the file size of your .gspl files

The AIR 2.5 runtime is required to install and run the gWordListEditor application. You can download the AIR runtime from adobe.com.

Expected Behaviors

Case Sensitivity

SPL provides basic support for case sensitivity. Words will be flagged as misspelled if they contain capitalized characters in the word list, but lack capitalization or are capitalized differently in the source text. Words will not be flagged if the word is not capitalized in the word list, but are capitalized in the source text.

For example, with the following words in the word list: “Canada”, “IBM”, “orange”

Word from source text	Flagged as misspelled?
canada	YES
CAnada	YES
ibm	YES
Ibm	YES
Orange	NO
orANGe	NO

Ignoring Capitalized Words

SpellingDictionary has three properties that affect how it deals with capitalized words:

ignoreInitialCaps – ignores words with *only* the first character capitalized
ignoreAllCaps – ignores words with all characters capitalized
ignoreMixedCaps – ignore words with capital letters other than falling under the above two options

Each of these properties is exclusive of the other. So for example, a dictionary with ignoreMixedCaps set to true and the other two properties set to false will not ignore all caps words or words with only the first character capitalized.

Word Definition

The SpellingHighlighter class currently checks words comprised of two or more characters from the following case-insensitive character sets:

Word Characters:

abcdefghijklmnopqrstuvwxyzàáâãäåæçèéêëìíîïðñòóôõöøùúûüýþÿß

Inner Word Characters:
'' (single quote and apostrophe)

Inner word characters bounding a word will be ignored. Two or more inner word characters in a row will also be considered a word break. Words containing and of the following "invalid word characters" will be ignored:

Invalid Word Characters:
0123456789

SpellingDictionary.checkString uses the same rules as above. SpellingDictionary.checkWord and checkList will check any strings passed to them (treating the full string as a single word), regardless of the characters.

You can change the word character set at runtime with the `com.gskinner.text.CharacterSet` class.

Performance

On a body of text with 5% of words misspelled, the engine can check and highlight over 25,000 words per second. This translates into about 15-20ms to check and highlight a large text area containing 500 words. Performance scales with the number of misspelled words. SpellingHighlighter currently checks the entire *visible* body of text whenever the text field is scrolled or the text is edited.

Spelling suggestions for a 6 letter word with a 150,000 word list, and a match tolerance of 0.5 takes approximate 200ms. This is highly variable (40-300ms) depending on the word (length in particular affects the time). The time to return a match typically increases with higher tolerances, longer words, and larger word lists. Because this is a user initiated action, we consider a delay of up to 500ms acceptable, however SpellingDictionary does support a "fast mode" for suggestions that operates approximately ten times faster, but will only return words that begin with the same letter as the source word.

All tests performed on a 2ghz CPU.

Languages Other Than English

Currently, we do not provide official support languages other than English. However, we have done a lot of work since SPL 1.2 to support other languages, and initial testing indicates that

languages using roman character sets (ex. German, French, Spanish, Italian, etc) work very well, provided you have a good word list. We are providing a number of word lists at the URL below, but do not presently support them directly.

http://gskinner.com/products/spl/wordlists/

We welcome feedback, and are happy to try to help you with non-English languages if you run into difficulty. We plan to formally support this feature in the future.

In some cases, you may need to tweak the character sets SPL uses for a particular language. For example, with French you should add "-" as an inner word character like this:

```
CharacterSet.innerWordChars = CharacterSet.DEFAULT_INNER_WORD_CHARS+"-";
```

Compatibility

To ensure that earlier versions of Flash or Flex Builder will compile, multiple versions of the SPL SWCs are provided. The TextField-based SWCs are compatible with Flash CS3 and later, and Flex 2.01 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 download for Flex 2.01. You can download both Hotfix 2 and 3 versions at http://gskinner.com/spl/flex2/

Adobe AIR Support

SPL provides full support for Adobe AIR since version 1.2. It also includes an AIRMenuHelper class (found in the "extras" directory) that will insert Cut, Copy, Paste, and Select All into the context menu of a target textfield (these are removed by default when you set a custom context menu in AIR). If you wish to include custom items in a text field using SPL or the AIRMenuHelper you must assign a ContextMenu instance as the `.contextMenu` for that text field. SPL does not support NativeMenu.