COMMONS TEXT

A LIBRARY FOCUSED ON ALGORITHMS WORKING ON STRINGS.

Created by Rob Tompkins (chtompki)
WHO I AM.

- chtompki@apache.org
- Apache Commons Committer
- Apache Commons Text Release Manager
- Software Developer (Java, DevOps)
- Mathematician/Logician (?, sure why not).
INTRODUCING commons-text.

- Goal. Text processing algorithms out of standard java library scope and promote reuse across all of Apache's projects.
- Secondary goal.
  - Remove heavier text processing mechanics from commons-lang.
  - Ensure lang minimally remains all that every Java developer needs.
HISTORY

- Traction on appetite for a Levenshtein Distance began to form in October 2014 in LANG-591.
- Bruno Kinoshita and Benedikt Ritter put together a proposal to create text in the sandbox.
- Last November, I picked up text where they left off, and by March 11, 2017 we had our 1.0.
CURRENT LAYOUT.

- Textier utilities than lang's StringUtils:
  - StrBuilder, FormattableUtils, StrSubstitutor, and StrTokenizer
- Diff utilities
- String similarity and edit distance.
- String translation and escaping (e.g. XML, CSV, JSON, etc.)
FROM lang
**StringBuilder**

- An alternative to `java.lang.StringBuilder`
- Better instance methods
- Not thread safe, not final

```java
StringBuilder sb = new StringBuilder("Test");
sb.readFrom(CharBuffer.wrap(" 123")); // "Test 123"
sb = new StringBuilder("bb");
sb.replaceAll("b", "xbx"); // "xbxbxb"
sb = new StringBuilder("abc");
sb.replace(0, 1, "aaa"); // "aaabc"
```
**FormattableUtils**

- Provides basic control over formatting when using a `java.util.Formatter`
- Primarily concerned with numeric precision and padding
- No generalized alternative forms

```java
FormattableUtils.append("foo",
    new Formatter(),
    FormattableFlags.LEFT_JUSTIFY,
    6, -1, '*').toString(); // "foo***"
FormattableUtils.append("foo",
    new Formatter(),
    FormattableFlags.LEFT_JUSTIFY,
    6, -1).toString(); // "fooo"
```
**StrSubstitutor**

- Provides a convenient way to do string substitutions
- Think of it as a template engine in one class

```java
Map valuesMap = HashMap();
valuesMap.put("animal", "quick brown fox");
valuesMap.put("target", "lazy dog");
String templateString = "The ${animal} jumped " +
                      "over the ${target}";
StrSubstitutor sub = new StrSubstitutor(valuesMap);
String resolvedString = sub.replace(templateString);
// "The quick brown fox jumped over the lazy dog."
```
**StrTokenizer**

- Tokenizes a string based on delimiters (separators) and supporting quoting and ignored character concepts
- Aims to do a similar job to java.util.StringTokenizer
- Offers much more control and flexibility including implementing the java.util.ListIterator interface
StrTokenizer

```java
final String input = "a;b;c;d;\e\f; ; ";
final StrTokenizer tok = new StrTokenizer(input);
tok.setDelimiterChar(';');
tok.setQuoteChar('"');
// Matches the String trim() whitespace characters.
tok.setIgnoredMatcher(StrMatcher.trimMatcher());
tok.setIgnoreEmptyTokens(false);
final String tokens[] = tok.getTokenArray();
// String[]{"a", "b", "c", "d;\e\f", ",", ",", ","};
```
**StringEscapeUtils**

- Provides a convenient way to do escaping

```java
StringEscapeUtils.escapeJson("He didn't say, \"stop!\"");
// "He didn\'t say, \"stop!\"")
StringEscapeUtils.escapeJava("\\b\t\r\n");
// "\\\\\\b\t\r\n"
```
NEW FUNCTIONALITY, UNIQUE TO text
**LongestCommonSubsequence**

The Longest commons subsequence is a classical String similarity algorithm.

```java
LongestCommonSubsequence siml =
    new LongestCommonSubsequence();
siml.apply("abba", "abab"); // 3
siml.apply("frog", "fog"); // 3
siml.apply("PENNSYLVANIA", "PENNCISYLVNIA"); // 11
siml.apply("elephant", "hippo"); // 1
```
LongestCommonSubsequenceDistance

```java
LongestCommonSubsequenceDistance dist =
    new LongestCommonSubsequenceDistance();
dist.apply("abba", "abab"); // 2
dist.apply("frog", "fog"); // 1
dist.apply("PENNSYLVANIA", "PENNCISYLVANIA"); // 3
dist.apply("elephant", "hippo"); // 11
```
LevenshteinDistance

Different algorithm with almost the same results.

```java
LevenshteinDistance dist = new LevenshteinDistance();
dist.apply("abba","abab"); // 2
dist.apply("frog", "fog"); // 1
dist.apply("PENNSYLVANIA", "PENNCISLYNIA"); // 3
dist.apply("elephant", "hippo"); // 7
```
WHAT ELSE IS THERE?

- `org.apache.commons.text.diff` contains a variety of diff tools.
- `org.apache.commons.text.similarity` contains various other similarity/distance tools
  - Cosine similarity and distance, Hamming distance, Jaccard distance, and Jaro-winkler.
- `org.apache.commons.text.translate` mainly supports `StringEscapeUtils`, but has more...
WHAT'S NEXT?

- Release 1.1 in the next month or so? Assuming no 1.0.1......
- WordUtils from lang, with some updates.
- RandomStringGenerator thanks to the rng crew.
- @Deprecated on code taken from lang
QUESTIONS?