Spiral: splitters for identifiers in source code files

Michael Hucka[1]

1 Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA

Summary

Spiral is a Python 3 package that implements numerous identifier splitting algorithms. *Identifier splitting* (also known as *identifier name tokenization*) is the task of breaking apart program identifier strings such as `getInt` or `readUTF8stream` into component tokens: [`get`, `int`] and [`read`, `utf8`, `stream`]. The need for splitting identifiers arises in a variety of contexts, including natural language processing (NLP) methods applied to source code analysis and program comprehension.

Spiral is easy to use. Here are some examples of calling the Ronin splitter algorithm from a Python program. The following input,

```
from spiral import ronin
for s in [ 'mStartCData', 'nonnegativedecimaltype', 'getUtf8Octets',
           'savefileas', 'nbrOfbugs']:
    print(ronin.split(s))
```

produces the following output:

```
['m', 'Start', 'C', 'Data']
['nonnegative', 'decimal', 'type']
['get', 'Utf8', 'Octets']
['save', 'file', 'as']
['nbr', 'Of', 'bugs']
```

Spiral also includes a command-line program named `spiral`; it will split strings provided on the command line or in a file, and is useful for experimenting with Spiral.

Motivations for developing Spiral

Splitting identifiers is deceptively difficult. It is a research-grade problem for which no perfect solution exists today. Even in cases where the input consists of identifiers that strictly follow conventions such as camel case, ambiguities can arise. For example, to split `J2SEProjectTypeProfiler` into [`J2SE`, `Project`, `Type`, `Profiler`] requires the reader to recognize J2SE as a unit. The task of splitting identifiers is made more difficult when there are no case transitions or other obvious boundaries in an identifier.

Spiral provides some several basic naive splitting algorithms, such as a straightforward camel-case splitter, as well as more elaborate heuristic splitters, including a novel algorithm we call *Ronin*. Ronin uses a variety of heuristic rules, English dictionaries, and tables of token frequencies obtained from mining source code repositories. It includes a default table of term frequencies derived from an analysis of over 46,000 randomly selected software projects in GitHub that contained at least one Python source code file.

Splitter algorithms available in Spiral

The following table lists the splitters implemented in Spiral at this time:

Splitter name	Operation
delimiter_split	split only at characters $ ~ _ . : / @
digit_split	split only at digits
pure_camelcase_split	split at forward camel case transitions (lower to upper case)
safe_simple_split	split at hard delimiter characters and forward camel case only; won't split strings that don't follow strict camel case
simple_split	split at hard delimiter characters and forward camel case, even if a string doesn't follow strict camel case conventions
elementary_split	split by hard delimiters, forward camel case, and digits
heuristic_split	split by hard delimiters, forward camel case, and digits, but recognize special cases such as utf8, sha256, etc.
Samurai	frequency-based algorithm published in the literature
Ronin	frequency-based algorithm originally based on Samurai

The name "Ronin" is a play on the use of the name "Samurai" (Enslen et al. 2009) for their identifier splitting algorithm. The core loop of Ronin is based on Samurai, but substantially modified and extended. A goal for Ronin was to produce a splitter that had good performance even without using a local frequency table.

The name *Spiral* is a loose acronym based on "*SPlitters for IdentifieRs: A Library*".

References

Enslen, E, E Hill, L Pollock, and K Vijay-Shanker. 2009. "Mining Source Code to Automatically Split Identifiers for Software Analysis." In *2009 6th IEEE International Working Conference on Mining Software Repositories*, 71–80. ieeexplore.ieee.org. https://doi.org/10.1109/MSR.2009.5069482.