with-assoc - a library for creating variables from alists

Cyrus Harmon

<2021-05-13 Thu>

Contents

1 with-assoc **1**
1.1 Overview 1
1.2 common-lisp:assoc 2
1.3 with-assoc 2
1.4 with-assoc-equal 2
1.5 with-assoc* 2
1.6 Default values 2

1 with-assoc

A library for working with association lists providing macros that hide the use of cl:assoc.

1.1 Overview

When working with assoc functions, one ends up calling something like:

```
(cdr (assoc foo :bar))
```

quite frequently. In order to save typing a few characters, we can define some macro-ology.
If we define an association list such as:

```
(defparameter *al* '((:foo . "bar")
                     (:baz . "wizard")))
```

The standrad way to get access to the element whose car is :foo would be:

```
(cdr (assoc :foo *al*))
```

With the with-assoc library we can do the following:

```
(with-assoc:with-assoc (foo)
    *al*
  foo)
```

Obviously this doesn't save us much typing (quite the opposite in fact), but if we're using lots of pairs in association lists and particularly if we're reusing the same values multiple value times, it can be convenient to have some short-hand syntax for

```
(let ((foo (cdr (assoc *al*) :foo)))
  ...)
```

1.2 common-lisp:assoc

The common lisp assoc function takes an item and a list of pairs (conses) and returns the first pair whose car satisfies a test. The specifics of the test are specified by the key and test keyword arguments. If the key argument is nil (the default) then the car of the pair itself is tested, otherwise the results of calling (key (car <pair>)) are tested against item via the provided for test function, or is via #'eql if no test function is supplied.

1.3 with-assoc

with-assoc is meant to be used when item is tested against the items in the associative list with #'eql.

```
(with-assoc:with-assoc (foo)
    *al*
  foo)
```

1.4 with-assoc-equal

with-assoc is meant to be used when item is tested against the items in the associative list with #'equal, such as when the items are strings, rather than keywords.

```
(defparameter *al2* '(("FOO" . "bar")
                      (:baz . "wizard")))

(with-assoc:with-assoc (:foo :baz)
    *al2*
  (list foo baz))
```

1.5 with-assoc*

with-assoc* take keyword arguments key and test which provide the full flexibility of the assoc (except for the deprecated :test-not argument).

```
(with-assoc:with-assoc* ((("FOO" foo) :baz) :test 'equal)
    *al2*
  (list foo baz))
```

1.6 Default values

The various flavors of with-assoc can take an optional default value in the event that none is provided in the alist. Take for example:

```
(with-assoc:with-assoc (:foo :baz (:blort blort "splat"))
    '((:foo . "bar")
      (:baz . "wizard"))
  (list foo baz blort))
```