The REBOL Documentation Project

-- EN - REBOL Documentation - Technical Articles --

Technical Articles

There is CATCH in REBOL

Ladislav Mecir

Première publication : 10 août 2005, et mis en ligne le mercredi 10 août 2005

Résumé :

Where ? Actually, there are two : CATCH function and [catch] function attribute.

Where ? Actually, there are two : CATCH function and [catch] function attribute.

1 The CATCH function

This function essentially enables us to immediately finish a complicated evaluation.

Sample code

```
catch [
    ; doing complicated evaluation...
    if done? [throw result1]
    ; more evaluations...
    if done? [throw result2]
    ; more evaluations...
    ; now we are done for sure
    result3
]
```

Similar effect can be achieved using ANY, ALL or CASE functions, but in the general case CATCH is the most appropriate for the job.

When using "embedded" CATCH evaluation, we can use the CATCH/NAME refinement to be able to specify the target for the desired results as follows :

```
catch/name [
    ; ...
    catch/name [
        ; ...
        if done-a? [throw/name result 'a] ; this is meant for the 'a CATCH (the "outer"
CATCH)
        ; ...
        if done-b? [throw/name result 'b] ; this is meant for the 'b CATCH (the "inner"
CATCH)
        ; ...
        result-b
    ] 'b
    ; ...
    if done-a? [throw/name result 'a]
    ; ...
```

```
    result-a
] 'a
```

2 The [catch] function attribute

This attribute is meant to enable "above standard" error handling in functions.

Example : let's suppose we need a HANDLE function to handle every integer number except for 66. We consider it an error to call the HANDLE function with number 66 as its argument. Code :

```
handle: func [
    [catch]
    argument1 [integer!]
    argument2 [block!]
] [
    if argument1 = 66 [throw make error! "number 66 not allowed as argument1"]
    do argument2
]
```

Now expression

```
handle 1 ["OK"]
```

yields "OK", while

```
handle 66 ["OK"]
```

is an error.

3 Comparison

- the [catch] function attribute has a different purpose than the CATCH function.
- there isn't a similar "mechanism" of discerning the meaning of THROW for embedded functions

4 Interference

An embedded [catch] - attribute function cannot discern the meaning of THROWS and it catches all throws as far as they are throwing errors. Example :

```
index: 1
block: copy ["OK"]
insert tail block make error! "error"
type? catch [
    ; ...
```

```
    handle 1 [throw pick block index]
]
```

So far so good, but if we change the first line to

```
index: 2
```

, the interference will appear. We may ask whether the /NAME refinement could help, but the answer is negative as Romano found out.

5 Solution

I offered TFUNC functions to solve the throw issue (as well as some other issues) for embedded functions earlier. (see http://www.compkarori.com/vanilla/display/Function-Attributes).

The interference between [catch] and CATCH can be solved by using local throws too, but some users may find this solution uncomfortable.

A simpler solution might use /NAME refinement for THROW, but that should be extended to enable us to specify an appropriate catch name for [catch] functions too. In that case I think, that it would be great if the name comparison was using = ? not just = to use the full power of Rebol.

The End

Post-scriptum : I am curious, whether you prefer a "local-word" solution or a /NAME solution, so consider this a user poll.