Utility Function: Pluralize()

Posted At : December 15, 2008 5:29 AM | Posted By : Jon Hartmann
Related Categories: Utility Function

So, I've been putting up a bunch of functions to work with Java patterns, especially working with groups. Now its time to put some of them to work to make a function to handle pluralization. Pluralizing arbitrary strings is important in dynamic systems, and used heavily in some ORM schemes, so I thought I'd try to make one. My first version is based on **pluralization regular expressions found at ThinkSharp**, although I've made some modifications and changes to make the substitutions more dynamic, and preventing pluralization rules from altering a word that already appears to be pluralized. Click more to check out the code and the tests.

```
<cffunction name="pluralize" output="false" returntype="string">
    <cfargument name="item" type="string" required="true" />

    <cfset var local = StructNew() />

    <!--- Things that are singular or plural, or not countable --->
    <cfset local.uncountable = "sheep,fish,series,species,money,rice,information,equipment" />

    <!--- Does not follow the normal pluralization rules --->
    <cfset local.irregular = {
        move    = "moves",
        sex        = "sexes",
        child    = "children",
        person    = "people"
    } />

    <!--- Pluralization rules, array to keep priority --->
    <cfset local.pluralizations = ArrayNew(2) />

    <cfset local.pluralizations[1][1] = "(quiz)$" />
    <cfset local.pluralizations[1][2] = "$1zes" />

    <cfset local.pluralizations[2][1] = "([a-zA-Z]+)?man$" />
    <cfset local.pluralizations[2][2] = "$1men" />

    <!--- Escape to prevent double pluralization --->
    <cfset local.pluralizations[3][1] = "^(oxen)$" />
    <cfset local.pluralizations[3][2] = "$1" />

    <cfset local.pluralizations[4][1] = "^(ox)$" />
    <cfset local.pluralizations[4][2] = "$1en" />

    <!--- Escape to prevent double pluralization --->
    <cfset local.pluralizations[5][1] = "^([m|l])ice$" />
    <cfset local.pluralizations[5][2] = "$1ice" />

    <cfset local.pluralizations[6][1] = "([m|l])ouse$" />
    <cfset local.pluralizations[6][2] = "$1ice" />

    <cfset local.pluralizations[7][1] = "(vertex|ind)ex$" />
    <cfset local.pluralizations[7][2] = "$1ices" />
    <cfset local.pluralizations[6][2] = "$1ice" />

    <cfset local.pluralizations[8][1] = "(matr)ix$" />
    <cfset local.pluralizations[8][2] = "$1ices" />

    <cfset local.pluralizations[9][1] = "(x|ch|ss|sh)$" />
```

```
    <cfset local.pluralizations[9][2] = "$1es" />

    <cfset local.pluralizations[10][1] = "([^aeiouy]|qu)y$" />
    <cfset local.pluralizations[10][2] = "$1ies" />

    <cfset local.pluralizations[11][1] = "(hive)$" />
    <cfset local.pluralizations[11][2] = "$1s" />

    <cfset local.pluralizations[12][1] = "(?:([^f])fe|([lr])f)$" />
    <cfset local.pluralizations[12][2] = "$1$2ves" />

    <cfset local.pluralizations[13][1] = "sis$" />
    <cfset local.pluralizations[13][2] = "ses" />

    <!--- Escape to prevent double pluralization --->
    <cfset local.pluralizations[14][1] = "([ti])a$" />
    <cfset local.pluralizations[14][2] = "$1a" />

    <cfset local.pluralizations[15][1] = "([ti])um$" />
    <cfset local.pluralizations[15][2] = "$1a" />

    <cfset local.pluralizations[16][1] = "(buffal|tomat)o$" />
    <cfset local.pluralizations[16][2] = "$1oes" />

    <cfset local.pluralizations[17][1] = "(bu)s$" />
    <cfset local.pluralizations[17][2] = "$1ses" />

    <cfset local.pluralizations[18][1] = "(alias|status)$" />
    <cfset local.pluralizations[18][2] = "$1es" />

    <!--- Escape to prevent double pluralization --->
    <cfset local.pluralizations[19][1] = "(octop|vir)i$" />
    <cfset local.pluralizations[19][2] = "$1i" />

    <cfset local.pluralizations[20][1] = "(octop|vir)us$" />
    <cfset local.pluralizations[20][2] = "$1i" />

    <cfset local.pluralizations[21][1] = "(ax|test)is$" />
    <cfset local.pluralizations[21][2] = "$1es" />

    <cfset local.pluralizations[22][1] = "xi$" />
    <cfset local.pluralizations[22][2] = "xies" />

    <cfset local.pluralizations[23][1] = "s$" />
    <cfset local.pluralizations[23][2] = "s" />

    <cfset local.pluralizations[24][1] = "$" />
    <cfset local.pluralizations[24][2] = "s" />

    <!--- Check if the item is in the uncountable list --->
    <cfif ListFindNoCase(local.uncountable, arguments.item)>
        <!--- If it is, set it as the return value --->
        <cfset local.returnValue = arguments.item />
    </cfif>

    <!--- Check if this value is in the irregular struct --->
    <cfif NOT StructKeyExists(local, "returnValue")>
        <!--- Loop over each irregular item --->
        <cfloop collection="#local.irregular#" item="word">
            <cfif arguments.item eq word OR arguments.item eq local.irregular[word]>
```

```
                <cfset local.returnValue = local.irregular[word] />
                <cfbreak />
            </cfif>
        </cfloop>
    </cfif>

    <!--- Test for pluralization rules --->
    <cfif NOT StructKeyExists(local, "returnValue")>
        <!--- Loop over each rule --->
        <cfloop from="1" to="#ArrayLen(local.pluralizations)#" index="x">
            <!--- Get a new pattern for this rule --->
            <cfset local.pattern = PatternNew(local.pluralizations[x][1]) />

            <!--- See if the pattern matches --->
            <cfif PatternFind(local.pattern, arguments.item)>
                <cfset local.returnValue = PatternReplace(local.pattern, arguments.item, local.pluralizations[x][2]) />

                <cfbreak />
            </cfif>
        </cfloop>
    </cfif>

    <cfif NOT StructKeyExists(local, "returnValue")>
        <cfset local.returnValue = arguments.item />
    </cfif>

    <cfreturn local.returnValue />
</cffunction>
```

Results of tests.

Pluralizations

matrix	matrices
complex	complexes
dictum	dicta
quiz	quizzes
ox	oxen
mouse	mice
index	indices
bench	benches
lily	lilies
dwarf	dwarves
thesis	theses
atrium	atria
tomato	tomatoes
bus	buses
alias	aliases
virus	viri
axis	axes
census	census

taxi	taxies
cat	cats
woman	women
man	men

Pre-Pluralized Tests

matrices	matrices
complexes	complexes
dicta	dicta
quizzes	quizzes
oxen	oxen
mice	mice
indices	indices
benches	benches
lilies	lilies
dwarves	dwarves
theses	theses
atria	atria
tomatoes	tomatoes
buses	buses
aliases	aliases
viri	viri
axes	axes
census	census
taxies	taxies
cats	cats