ForEach package

Mark A. Caprio, Center for Theoretical Physics, Yale University

Version 1.0 (January 12, 2005)

Introduction

The ForEach package provides iteration control constructs analogous to the "for each" loop control structures found in many scripting languages. These constructs repeatedly evaluate an expression, with the iteration variable successively taking on each value in an enumerated set.

Basic syntax

`DoForEach[` *expr* `, {` *var* `, {` *value1*`, ` *value2*`, ...}}]`	Evaluates *expr* multiple times, with *var* successively taking on each value *value1*, *value2*, ...
`TableForEach[` *expr* `, {` *var* `, {` *value1*`, ` *value2*`, ...}}]`	Generates a list of values of *expr*, for *var* successively taking on each value *value1*, *value2*, ...

"For each" iteration constructs.

Two constructs are provided. `DoForEach` simply evaluates the given expression, while `TableForEach` returns a list of the results. These constructs are closely modeled upon the usual *Mathematica* `Do` and `Table` constructs but accept a different format for the iterator specification. The values of the iteration variables in `DoForEach` or `TableForEach` are local, and any global value for a variable will not be affected by use its as the loop iteration variable. Here are some examples:

```
DoForEach[Print[x], {x, {"First", "Second", "Third"}}]
```

```
First
Second
Third
```

```
TableForEach[x, {x, {"First", "Second", "Third"}}]
```

```
{First, Second, Third}
```

```
ValueList = {1, 2, 3, 4};
TableForEach[x^2, {x, ValueList}]
```

```
{1, 4, 9, 16}
```

`DoForEach[` *expr* `,` `{` *var*`,` *index* `,` `{` *value1*`,` *value2*`,` ... `}}]`	Evaluates *expr* multiple times, with *var* successively taking on each value *value1*, *value2*, ..., and with *index* taking on successive values starting from 1.
`TableForEach[` *expr* `,` `{` *var*`,` *index* `,` `{` *value1*`,` *value2*`,` ... `}}]`	Generates a list of values of *expr*, for *var* successively taking on each value *value1*, *value2*, ..., and with *index* taking on successive values starting from 1.

Alternate syntax for "for each" iteration constructs, with optional index variable.

It is often useful to have access to an "index" or "counter" variable, indicating the number of the iteration. An alternate format for specifying iterators is provided for this.

```
WordList = {"First", "Second", "Third"};
DoForEach[
 Print["Word #", i, " is `", x, "'."],
 {x, i, WordList}
]
```

```
   Word #1 is `First'.

   Word #2 is `Second'.

   Word #3 is `Third'.
```

Note that the `DoForEach` and `TableForEach` constructs provide shorthands for more the cumbersome constructs usually used to manually iterate over sets of values, such as

```
ValueList = {1, 2, 3, 4};
Block[
 {i},
 Table[
  ValueList[[i]]^2,
  {i, 1, Length[ValueList]}
 ]
]
```

```
{1, 4, 9, 16}
```

Multiple iterators

Multiple iterators may be specified, resulting in nested loops, much as for the usual `Do` and `Table` constructs. The variable specified in the rightmost iterator varies most rapidly. Iterators with or without counters may be freely intermixed as arguments in calls to `DoForEach` and `TableForEach`.

By default, if `TableForEach` is given multiple iterators, it produces a nested list of values. Thus, a call with one iterator results in a vector, a call with two iterators results in a matrix, a call with three iterators results in a rank-3 tensor, *etc.*

```
VariableList1 = {u, v, w};
VariableList2 = {x, y, z};
TableForEach[
 a * b,
 {a, VariableList1}, {b, VariableList2}
]
% // MatrixForm
```

```
{{u x, u y, u z}, {v x, v y, v z}, {w x, w y, w z}}
```

$$\begin{pmatrix} u\,x & u\,y & u\,z \\ v\,x & v\,y & v\,z \\ w\,x & w\,y & w\,z \end{pmatrix}$$

option name	*default value*	
`Flatten`	`False`	Controls whether a flat or nested list of values is returned.

List nesting control option for `TableForEach`.

However, with the option setting `Flatten→True`, a single flat list of results is returned. Compare the result here with that above:

```
TableForEach[a * b, {a, VariableList1}, {b, VariableList2}, Flatten → True]
```

```
{u x, u y, u z, v x, v y, v z, w x, w y, w z}
```

© Copyright 2005, Mark A. Caprio.