MathGroup Archive 2002

RE: creating packages

To: mathgroup at smc.vnet.net

Subject: [mg32918] RE: [mg32889] creating packages

From: "David Park" <djmp at earthlink.net>

Date: Tue, 19 Feb 2002 02:30:00 -0500 (EST)

Sender: owner-wri-mathgroup at wolfram.com

```
Poonam,

Many people seem to have trouble creating packages and even people who are
quite expert at Mathematica create packages in unconventional and difficult
to use ways. Few packages that are on MathSource seem to follow a standard
convention. So I would like to lay out my humble opinion of how Mathematica
packages should be done. It is actually quite simple.

1) If you want to make your package easy to use, especially to other users,
then it should operate and behave just like a standard package. The way to
do this is to put the package in a subdirectory of the AddOns/ExtraPackages
directory. Mathematica automatically looks there when you load a package and
so it will load just like a regular package.

Suppose you want to write a package that contains extra algebra routines
beyond what Mathematica has. Create a folder in ExtraPackages called
Algebra. (This parallels a similar folder in StandardPackages, but you could
use any name you wanted for the folder.) Suppose you want to call your
package PoonamAlgebra. Then you want to create a package named
PoonamAlgebra.m and put it in the ExtraPackages/Algebra folder. Then the
package could be loaded with

Needs["Algebra`PoonamAlgebra`"]

Mathematica will automatically look in both the StandardPackages and the
ExtraPackages folders for an Algebra folder containing PoonamAlgebra.

2) The name used in the BeginPackage statement must correspond to the path
from the ExtraPackages folder to your package. So, for this example you
would use:

BeginPackage["Algebra`PoonamAlgebra`"]

3) The best method for writing and creating a package is to use a standard
Default style notebook. In our example the notebook would have the name
PoonamAlgebra.nb, i.e., the same name as the package except with a .nb
instead of a .m. In practice, just start with a fresh notebook, write your
package statements and save it. It is best practice to write each of your
package statements as a separate cell. Don't try to put all the code in one
cell! Put this notebook in the same folder that will contain the
PoonamAlgebra.m file.
```

```
4) This is how the package is actually created. Make all of the code cells
initialization cells. You can do this by selecting the cells and using the
menu item Cell/Cell Properties/Initialization Cell. Then, when you first
save the notebook, Mathematica will ask you if you want to create an
AutoSave package. Answer Yes. Mathematica will then automatically create the
PoonamAlgebra.m package. Furthermore, anytime you update the notebook and
save it, the package will also be updated.

5) Each routine that you want to export needs a usage message after the
BeginPackage statement. The routine itself is put in the Private section. At
the end of this posting I have put a notebook expression for a toy package.
Pasting it into a fresh notebook will give you a bare bones outline of a
package.

6) In developing new routines for a package I usually develop them in an
applications notebook and then copy and paste them into the package
notebook. The main problem that sometimes occurs is when you are using a
global symbol. In a notebook x is the same as Global`x. If that is what you
want in a package you must explicitly write Global`x. This should not occur
too often since everything used in a routine should generally be passed as
an argument. But here is an example, suppose you have a routine that
generates a polynomial from some data. Perhaps you want the polynomial
variable to usually be x, but allow it to be changed with a default
argument. In a notebook the routine might be defined as:

GenPoly[data_, var_:x]:=...

In a package this must be changed to

GenPoly[data_, var_:Global`x]:=...

7) An excellent book covering most aspects of writing packages is
"Programming in Mathematica: Third Edition" by Roman Maeder.

David Park
djmp at earthlink.net
http://home.earthlink.net/~djmp/



> From: poonam [mailto:poonam_pandey at hotmail.com]
To: mathgroup at smc.vnet.net
>
> I am a first time user of mathematica and currently using version 4.0.
> I am trying to create a package. I am not very clear about how and
> where to write the package. I tried to use the dumpsave method, but
> its not creating the binary format .mx file. Where do i have to write
> the package .nb or in text editor and how do i run the dumpsave
> command? Or do i need to do this in text editor and save as .m?
>
> Can somebody clear my doubts. ?
>
>
> Thanks
>

The following is a notebook expression for a toy package. Copy the entire
expression and paste it into a fresh notebook, letting Mathematica interpret
it.

Notebook[{

Cell[CellGroupData[{
Cell["A Toy Package", "Title"],

Cell[BoxData[
    \(BeginPackage["\<Algebra`ToyPackage`\>"]\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(\(Algebra`ToyPackage::usage = "\<This is a usage mesage for the \
entire package.\>";\)\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(\(SquareTheQuantity::usage = "\<SquareTheQuantity[x] will take \
the square of x.\>";\)\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(Begin["\<`Private`\>"]\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(SquareTheQuantity[x_] := x\^2\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(End[]\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(\(Protect[Evaluate[$Context <> "\<*\>"]];\)\)], "Input",
  InitializationCell->True],

Cell[BoxData[
    \(EndPackage[]\)], "Input",
  InitializationCell->True]
}, Open  ]]
},
FrontEndVersion->"4.1 for Microsoft Windows",
ScreenRectangle->{{0, 1280}, {0, 943}},
AutoGeneratedPackage->Automatic,
WindowSize->{494, 740},
WindowMargins->{{Automatic, 188}, {Automatic, 5}}
]
```

- Prev by Date: **Re: creating packages**
- Next by Date: **Re: FindMinimum within a specific interval**
- Previous by thread: **Re: creating packages**
- Next by thread: **Output control**