import

Add package or class to current import list

Syntax

```
import PackageName.ClassName
```
example

```
import PackageName.FunctionName
```
example

```
import PackageName.*
```
example

```
import
```
example

```
L = import
```

Description

`import PackageName.ClassName` adds the class name to the current import list. Use the `import` function in your code to refer to a class without specifying the entire package name. example

The import list scope is defined as follows:

- Script invoked from the MATLAB® command prompt — Scope is the base MATLAB workspace.
- Function, including nested and local function — Scope is the function and the function does not share the import list of the parent function. If the import list is needed in a MATLAB function or script and in any local functions, you must call the `import` function for each function.

  The import list of a function is persistent across calls to that function and is cleared only when the function is cleared. For more information, see the `clear` function.

To clear the current import list, type `clear import` at the MATLAB command prompt. Do not call `clear import` within a function.

`import PackageName.FunctionName` adds the specified package-based function. Use this syntax to shorten the name of a specific function in a package without importing every function in the package, which might cause unexpected name conflicts. example

`import PackageName.*` adds the specified package name. `PackageName` must be followed by `.*`. example

Avoid using this syntax, as importing packages brings an unspecified set of names into the local scope, which might conflict with names in the MATLAB workspace. One possible use for this syntax is to import a partial package name. Then when you call a function, you use a shorter package name which does not conflict with simple function names. For example, the `matlab.io.hdf4.sd` package has a `close` function, which can conflict with the MATLAB `close` function.

`import` displays the current import list in the scope. example

`L = import` returns the current import list.

Examples

collapse all

Shorten Calls to Java Class Methods

Open This Example

```
import java.util.Currency java.lang.String
```

Create a `java.lang.String` object. There is no need to type the package name, `java.lang`.

```
s = String('hello')
```

```
s =

hello
```

List the `Currency` class methods, without typing the package name.

```
methods Currency
```

```
Methods for class Currency:

equals                   getDisplayName           notify
getAvailableCurrencies   getInstance              notifyAll
getClass                 getNumericCode           toString
getCurrencyCode          getSymbol                wait
getDefaultFractionDigits hashCode
```

Shorten HDF4 Scientific Data Set Package Name

Use partial package names on your import list to simplify calls to `matlab.io.hdf4.sd` package functions and avoid conflicts with the MATLAB `close` function.

```
import matlab.io.hdf4.*
```

Display the full path to the example file `sd.hdf` on your system using the shortened package name `sd`.

```
sdID = sd.start('sd.hdf');
filename = sd.getFilename(sdID)
```

```
filename =

C:\Program Files\MATLAB\R2015a\toolbox\matlab\imagesci\sd.hdf
```

Call the close function with the `sd` package name.

```
sd.close(sdID)
```

There is no name conflict with the MATLAB `close` function when you import the partial package name.

```
which close
```

```
C:\Program Files\MATLAB\R2015a\toolbox\matlab\graphics\close.p
```

If you use the `matlab.io.hdf4.sd.*` syntax to import the entire package name, when you call `close`, MATLAB always chooses the package function. You cannot use `close` to remove a figure.

Import Single Package Function

Import the `matlab.io.hdf4.sd` package function, `readChunk` in a function, `myfunc`. You can call the function using the simple `readChunk` name, but only within the scope of `myfunc`.

```
function data = myfunc(ID,n,m)
import matlab.io.hdf4.sd.readChunk
data = readChunk(ID,[n m]);
end
```

Import Package in Both Script and Function

Open the `sd.hdf` example file and access the `temperature` data set.

```
import matlab.io.hdf4.*
sdID = sd.start('sd.hdf');
idx = sd.nameToIndex(sdID,'temperature');
sdsID = sd.select(sdID,idx);
```

Call the `myfunc` function from the previous example to read the data. `myfunc` must have its own `import` statement in order to use a shorted package name.

```
dataChunk = myfunc(sdsID,0,1);
```

Close the file.

```
sd.endAccess(sdsID)
sd.close(sdID)
```

Display Current Import List for Your System

```
import
```

```
ans =

    'java.util.Currency'
    'java.lang.String'
    'matlab.io.hdf4.*'
    'matlab.io.hdf4.sd.readChunk'
```

Related Examples

- Use import in MATLAB Functions
- Package Function and Class Method Name Conflict

Input Arguments

collapse all

`PackageName` — Name of package
character array

Name of the package, specified as a character array.

Example: `matlab.io.hdf4`

`ClassName` — Name of class
character array

Name of the class, specified as a character array.

Example: `Currency`

`FunctionName` — Name of package function
character array

Name of the package function, specified as a character array.

Example: `readChunk`

Output Arguments

collapse all

`L` — Import list
cell array of character arrays

Import list returned as a cell array of character arrays.

Limitations

- `import` cannot load a Java® JAR package created by the MATLAB Compiler SDK™ product.
- Do not use `import` in conditional statements inside a function. MATLAB preprocesses the `import` statement before evaluating the variables in the conditional statements.

More About

- Import Classes

See Also

`clear` | `importdata` | `load`

Introduced before R2006a