Name

xs - XDC Script Interpreter

Synopsis

```
xs [options] xdc-package [script-arg1 script-arg2 …]
xs [options] -c xdc-capsule [script-arg1 script-arg2 …]
xs [options] -f xs-script-file [script-arg1 script-arg2 …]
xs [options] -m xdc-module [script-arg1 script-arg2 …]
xs [options] -i
```

Description

The `xs` command is used to execute XDC-Script programs. XDC-Script is a general-purpose programming language based upon the ECMA-262 Edition 3 standard – popularly known as JavaScript 1.5. A central element of XDC, the language primarily serves as an implementation vehicle for hosted *meta-content* used to automate integration of companion *target-content* within executable programs tailored for a specific hardware platform and optimized for a specific application environment. XDC also leverages the language to manage the flow of *packages* – standardized containers holding re-usable target content plus its attendant meta-content – between producers and consumers. And of course, the language is readily suited for standalone scripting tasks.

The `xs` command is implemented using the Java programming language. The actual JavaScript interpreter is based on the Mozilla Rhino package. Consult the documentation for Rhino to learn more about the particular features of this implementation of JavaScript. In particular, look for material on LiveConnect, which enables JavaScript programs to transparently invoke Java APIs. This powerful feature provides a mechanism to tap into vast amount of functionality in the Java ecosystem.

Options

`@opts-file` read the specified and include its contents as arguments to `xs`

`-D<name>=<val>`

define the (Java) environment variable <name> to have the value <val>

`--help` display usage help and exit

`--xdcpath <xdc-path>`

specify a list of repositories (directories) to be used as part of the XDC Package Path. The <xdc-path> argument should be a semi-colon separated list of directories.

`-i` start an interactive session. A prompt is issued to the console where JavaScript statements can be entered for immediate execution. The session can be terminated by typing *quit*

`-c <capsule> arg1 arg2 …`

execute the script in the function `main()` of the specified capsule, passing any remaining arguments to the script

`-f <file> arg1 arg2 …`

execute the script in the contained in the specified file, passing any remaining arguments to the script

`-m <module> arg1 arg2 …`

execute the script in the function `main()`of the specified module, passing any remaining arguments to the script

`<package> arg1 arg2 …`

execute the script in the function `main()` in the file `Main.xs` of the specified package, passing any remaining arguments to the script

Usage

The basic model for using the `xs` command is to specify a script to be executed along with any (optional) arguments to be passed to the script:

```
xs [options-to-xs] script arg1 arg2
```

The variations in invocation are due to how the "script" is located by `xs`. The four choices are:

- Script file located anywhere on the filesystem (-f option)
- Script code contained in an XDC Capsule (-c option)
- Script code is part of an XDC Module (-m option)
- Script code is part of an XDC Package (default: no option required)

See the Examples section below for sample invocations.

An interactive session can be started by using the –i option. In this mode of operation, the `xs` command displays a prompt - `js>` - and reads lines of script entered by the user. This can be useful for exploring the XDC-Script environment. To exit an interactive session, enter *quit*.

Arguments supplied to scripts can be retrieved using the standard JavaScript "arguments" object. This is an array-like object containing the arguments passed to the function, indexed beginning at 0. The "length" property of the object can be used to determine the number of arguments actually passed. This is true even for scripts that do not have a function defined, but rather just start execution with the first line in the file.

Controlling the `xs` Context/Environment

It can be useful to set, what is in effect, a "global" variable when running an XDC-Script program. This can be done using the `-D` option to `xs` and then retrieving that value from the "environment" in the script being run. The environment is a hash table indexed by the name of the environment variable. Note that these values are not the same as the command shell's environment variables.

As an example, the following will set an environment variable called "junk" to the value "car".

```
xs –Djunk=car –f envvar.xs
```

The environment could be accessed and printed as shown below:

```
print("Env var: " + environment["junk"]);
```

The `--xdcpath` option is used to set the user portion of the XDC Package Path. (See the documentation for the `xdc` command for more details on the XDC Package Path.) To specify a directory to be used as a repository, add it to a list of semicolon-separated directories supplied with the `--xdcpath` option. See the following section for an example.

Taking Options and Arguments from a File

The `@` option enables re-using common options and arguments captured in a file. One common scenario is the specification of the user part of XDC Package Path. If several repositories (directories containing packages) are being used, the

argument to the `--xdcpath` option can get quite long and tedious to re-enter each time `xs` is used. By creating a file named opts-file that contains two lines:

```
--xdcpath
dir-path1;dir-path2;dir-path3
```

and then starting `xs` as shown below,

```
xs @opts-file …
```

the `xs` command effectively is invoked as

```
xs --xdcpath dir-path1;dir-path2;dir-path3 …
```

In general, each line of the file supplied with the @, is added to the argument list of the `xs` command. The process is recursive so that if there is a line with an @ option, that file will be opened and added to the argument list, and so on.

Examples

Hello World

The following command will run the code in the file hello.xs.

```
xs –f hello.xs world
```

Assuming hello.xs contains the line:

```
print(“hello ” + arguments[0]);
```

the text “hello world” will be echoed to the standard output of the command shell.

Interactive Session

The following command will start the XDC-Script interpreter and issue a prompt for an interactive session.

```
xs –i
```

The prompt used by `xs` is “`js>` “ (the “`$` “ below is the command-line prompt from the shell being used):

```
$ xs -i
```

```
js> print("hello world")

hello world

js> quit

$
```

Setting and Retrieving an Environment Variable

This example shows how to set and retrieve an environment variable. This approach works for any of the methods for finding and running a script.

```
$ xs -Djunk=car -i

js> print("Env var: " + environment["junk"]);

Env var: car

js> quit

$
```

Running a Script Contained in a Capsule

An XDC-Script Capsule is simply a Package that contains XDC-Script code that can be loaded and utilized via the xdc.`loadCapsule()` method. If the Capsule contains a `main()` method, it can be run using `xs` by specifying the Capsule with the `-c` option.

```
xs -c sample/stuff/hello.xs world
```

The Package is `sample.stuff` and will be located along the Package Path. Note that instead of using the Package name directly, the "." separator is replaced with "/". The script is contained in the file `hello.xs`. There are no requirements on the naming of the script file (different file extensions can be used) or the actual location of the file within the package (sub-directories can be used).

Running a Script Contained in a Module

A meta-only Module that contains a method named "main" can be run using `xs` by specifying the Module with the `-m` option.

```
xs -m sample.stuff.Hello world
```

The Package is `sample.stuff` and will be located along the Package Path. The module Hello must contain a function named "main", as specified in the corresponding XDC-Spec file, `Hello.xdc`, and implemented in `Hello.xs`.

Running a Script Contained in a Package

An XDC Package that contains a `main()` method in a module named `Main` can be invoked directly as shown below.

```
xs sample.stuff world
```

The Package is `sample.stuff` and will be located along the Package Path. The script is contained in the file `Main.xs`. Note that this is equivalent to the following invocation using the Module option:

```
xs -m sample.stuff.Main world
```

Environment Variables

`XDCPATH` See the description of this environment variable in the documentation for the `xdc` command.

Exit Status

The exit status of the `xs` command is one of the following values:

0 Successful completion.

Non-zero An error occurred.

A script can indicate an error by throwing a JavaScript exception, which will cause `xs` to exit with a non-zero status.

See Also

Mozilla Rhino Project: http://www.mozilla.org/rhino

JavaScript Arguments:
http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference:Functions:arguments