External methods and the Jade DEMODLL example (readme file[1])

This file contains details of the contents of the DEMODLL directory which is installed as a sample as part of the JADE installation.

This directory contains the following files:

readme.txt	This file you are reading.
demodll.cpp	C++ source of sample JADE external methods
demodll.def	Windows Module definition file for DEMODLL.DLL.
demodll.vcproj	Microsoft Visual Studio 2005 C++ project file.
demodll_vs2008.vcproj	Microsoft Visual Studio 2008 C++ project file.
Makefile	UNIX/Linux make file
demodll.scm/demodll.ddb	JADE schema and ddb file defining the DEMODLL library and associated external methods.

File locations

1 The sample files for demodll are located in

<jade install directory>\examples\demodll for Windows and

<jade install directory>/examples/demodll for Linux.

The project build files for each architecture assume the default install layout to locate the necessary files and directories

<jade install directory>\bin,

<jade install directory>\include,

<jade install directory>\library (Win32 Windows),

<jade install directory>\library (CompactJADE Win32 Windows);

<jade install directory>\library\armv4i-msoft-wce50 (CompactJADE Device Mobile 5.0),

<jade install directory>\library\armv4i-msoft-wm60 (CompactJADE Device Mobile 6.x if present), and

<jade install directory>/include,

<jade install directory>/lib on Linux

2. If the locations are changed, the provided project build files will need to be modified.

Steps to build the sample DEMODLL.DLL on Windows

1. Using required documented version of Microsoft Visual Studio IDE open demodll.vcproj.
2. Select the appropriate "Solution Configuration". Provided are the following released solution configurations:

   Using demodll.vcproj:

   Debug_Ansi and Debug_Unicode build un-optimized Win32 binaries,

   Release_Compact (plus appropriate "Solution Platforms") to build optimized binaries.

   Using demodll_vs2008.vcproj

   Release_Ansi and Release_Unicode build optimized Win64 binaries.

   Win32 and Compact configurations are also included in the Visual Studio 2008 project file.

[1] From the corresponding readme.txt file.

3. Build the solution.

   The resulting demodll.dll will be placed in the bin directory along with the standard (Win32 Windows) Jade binaries.

   The demodll.dll output for CompactJADE devices will be placed in the <jade install directory>/mobile/<architecture>/bin directory along with the other device binaries.

Steps to build the sample libdemodll.so on Linux

1. In a terminal window, change into the examples/demodll directory.
2. The supplied Makefile is setup to build an Ansi library. If a Unicode library is required uncomment (remove the # character from the front of the line) the "UNICODE = -DUNICODE" line.
3. Type "make" to build the library. The resulting libdemodll.so will be placed in the lib directory along with the standard Jade libraries.

Using the DEMODLL sample

1. Initiate a JADE session.
2. Load the DEMODLL.SCM provided. This will create a new schema -DemoExternalMethods.
3. To test the execution of the sample methods open a Class Browser and execute any of the JadeScript methods, eg.useToUpper or useIntegerArrayMerge
4. There is a simple application DemoExternalMethods that can be run that will also test the same code. Useful to test on CompactJADE devices.
5. The DEMODLL dynamic library can also be used as a test Jade Developement Security library, to enable this alter the JADE .INI file entry [JadeSecurity] / DevelopmentSecurityLibrary from <default> to demodll. The code as provided will simply log to a file jadeDevelopment.log in the current directory the actions performed.

Dm79_ExternalMethods

The textbook's \Demos\ file contains Dm79_ExternalMethods which in turn contains Jadescript methods to call upon the same external methods

All the files referenced above have copies placed into the text's folder \Demos\ExternalMethodsFiles.