

Recital/4GL and Java™

Using Java™ with the Recital/4GL

Recital Corporation,
100 Cummings Center, Suite 318J
Beverly, MA 01915

Recital may have patents and/or patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents.
COPYRIGHT ©1988-2007 Recital Corporation. All rights reserved. All Recital products are trademarks or registered trademarks of Recital Corporation, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.

Last Updated September, 2007

INTRODUCTION

All Recital products can call Java™ classes residing on the server machine, providing that server is a Java™ enabled platform. This is achieved using the Recital/4GL in Recital Terminal Developer and via the Recital Universal Application Server (UAS) for client products.

DB_CLASSDIR

The DB_CLASSDIR environment variable (set in the file profile.db for character mode products and the file profile.uas for the Universal Application Server) defines the default classes directory. When a request is made to instantiate a new Java™ object, the class file is sought in the directory defined by DB_CLASSDIR. If the class file does not exist in this directory, the objects.ini file is read.

OBJECTS.INI

The objects.ini file is a system-wide definition file for Java™ and other classes. It resides in the directory defined by the environment variable DB_ROOT in the profile.uas/profile.db file. DB_ROOT defines the home directory of the product in question.

The objects.ini file allows Java™ classes to be referred to by a single name from the Recital/4GL, while residing anywhere on the system. For example:

```
# Sample objects.ini file
[jnitest]
type=JNI
progid=/usr/recital/uas/myobjects/jnitest.class
```

The first line defines the name that will be used in the Recital/4GL to refer to the class. The **JNI** type defines the class as a Java™ class and the progid gives the full path for the class.

USING Java™ OBJECTS

The Recital/4GL NEW operator is used to instantiate a new object based on a Java™ class. For example:

```
ojava = new jnitest()
```

Once the object has been instantiated, its properties can be queried or set using the <object>.<property> syntax. For example:

```
? ojava.AVALUE  
hello world
```

Its methods are called using the <object>.<method>() syntax. For example:

```
// Returns parameter1 multiplied by (parameter2 + parameter3)  
? ojava.mIII(2,3,4)  
14
```

The example jnitest Java™ source file is included in the Universal Application Server distribution in the netobjects directory.