Archived Code Engineering (Copy)

MagicDraw Professional Java Edition possesses reverse engineering capability for Java. You may also reverse descriptors and will get a model describing your Enterprise Java Beans.

More information about MagicDraw editions you may find at www.nomagic.com/editions.php
Yes, it does. We call this "round-trip engineering". You can reverse your source code, make changes in the model and regenerate it back without losing already coded parts (e.g. method bodies). You must use the code engineering set (the MagicDraw name for a collection of classes for code engineering). If you only do framework generation (Ctrl+G), you will receive only the generated framework (all other information will be lost), so be accurate at this point.
Performing the Quick Reverse you may instantly add new model elements or merge with the created ones in your model; you don't need to create a new Round Trip Set. Use it if you don't want to use code generation on the reverse code constantly.
Choose the 'New From Template' command from the 'File' menu. You may also reverse the needed Java classes and import them to the project.
You can use bytecode engineering set for reversing class files placed in your jar. Right click on "Code Engineering sets" node in browser, "New -> Java bytecode" and then edit created code engineering set by adding class files from jar file. Execute "Reverse" action on created code engineering set and classes will be created in model.
When I reverse engineer my project, there are a number of classes that are referenced in my code that cannot be found and are placed in the "Default" package. All of the classes are found in .jar files that my code depends on. I have checked the box that tells the reverse engineer to look in the classpath for classes. Is there somewhere in magicdraw where I should set the classpath?
Since MagicDraw 12.1 SP2, you may reverse QT library.

To reverse QT library using MagicDraw 12.1 SP2, you should add the following macro into MagicDraw > Options->Project... C++ Language Options ... Use explicit macros (these macros are included into MagicDraw starting from v12.5)

```
// QT

#define Q_AUTOTEST_EXPORT

#define Q_CLASSINFO(name, value)

#define Q_COMPAT_EXPORT

#define Q_CORE_EXPORT

#define Q_CORE_EXPORT_INLINE inline

#define Q_D(Class)

#define Q_DECL_IMPORT

#define Q_DECLARE_ASSOCIATIVE_ITERATOR(map)

#define Q_DECLARE_BUILTIN_METATYPE(TYPE, NAME)

#define Q_DECLARE_EXTENSION_INTERFACE(IFace, IId)

#define Q_DECLARE_FLAGS(Flags, enum)

#define Q_DECLARE_INTERFACE(IFace, IId)

#define Q_DECLARE_METATYPE(txt)

#define Q_DECLARE_MUTABLE_SEQUENTIAL_ITERATOR(c)

#define Q_DECLARE_MUTABLE_ASSOCIATIVE_ITERATOR(c)

#define Q_DECLARE_OPERATORS_FOR_FLAGS(Flags)

#define Q_DECLARE_PRIVATE(Class)

#define Q_DECLARE_PUBLIC(Class)

#define Q_DECLARE_SEQUENTIAL_ITERATOR(name)

#define Q_DECLARE_SHARED(name)

#define Q_DECLARE_TYPEINFO(TYPE, FLAGS)

#define Q_DECL_DEPRECATED

#define Q_DISABLE_COPY(Class)

#define Q_DUMMY_COMPARISON_OPERATOR(c)

#define Q_ENUMS(x) #define Q_FLAGS(x)

#define Q_GADGET #define Q_GUI_EXPORT

#define Q_GUI_EXPORT_INLINE inline
```

#define Q_INLINE_TEMPLATE

#define Q_INTERFACES(x)

#define Q_NETWORK_EXPORT

#define Q_NOREPLY

#define Q_OBJECT

#define Q_OPENGL_EXPORT

#define Q_OUTOFLINE_TEMPLATE inline

#define Q_OVERRIDE(text)

#define Q_PRIVATE_SLOT(d, signature)

#define Q_PROPERTY(text)

#define Q_Q(Class)

#define Q_REQUIRED_RESULT

#define Q_SCRIPTABLE

#define Q_SIGNALS protected

#define Q_SLOTS

#define Q_SQL_EXPORT

#define Q_SVG_EXPORT

#define Q_TESTLIB_EXPORT

#define Q_TYPENAME typename

#define Q_XML_EXPORT

#define QDBUS_EXPORT

#define QDESIGNER_COMPONENTS_EXPORT

#define QDESIGNER_EXTENSION_EXPORT

#define QDESIGNER_SDK_EXPORT

#define QDESIGNER_SHARED_EXPORT

#define QDESIGNER_UILIB_EXPORT

#define QDESIGNER_WIDGET_EXPORT

#define QDOC_PROPERTY(text)

#define QT_ASCII_CAST_WARN

#define QT_ASCII_CAST_WARN_CONSTRUCTOR

#define QT_ASSISTANT_CLIENT_EXPORT

#define QT_BEGIN_HEADER

#define QT_DEPRECATED

#define QT_DEPRECATED_VARIABLE

#define QT_DEPRECATED_CONSTRUCTOR

#define QT_END_HEADER

#define QT_FASTCALL

#define QT_FT_BEGIN_HEADER

#define QT_FT_END_HEADER

#define QT_MOC_COMPAT

#define QT_MODULE(name)

#define QT_STATIC_CONST static const

#define QT_STATIC_CONST_IMPL const

#define QT_TR_NOOP(x)

#define QT_TRANSLATE_NOOP(scope, x)

#define slots

#define signals protected

// Microsoft extension

#define __forceinline inline

#define CALLBACK

#define LRESULT void

#define _ENTRY(p1, p2, p3, p4) p1
Data reading error: ORA-06502: PL/SQL: numeric or value error

LPX-00210: expected '<' instead of 'n'

ORA-06512: at "SYS.UTL_XML", line 0

ORA-06512: at "SYS.DBMS_METADATA_INT", line 3296

...

There is a bug in Oracle 9 METADATA package.

Solution can be to reverse with user having all privileges or applying latest patch to Oracle 9.x.

As a workaround is to export database to DDL file and reverse with MagicDraw.
Our code engineering supports Java 6. Java 7 isn't supported yet. This capability is scheduled for our 17.0.2 release.