IBM VisualAge® for Java™, Version 3.5

SQLJ Tool

Note!
Before using this information and the product it supports, be sure to read the general information under **Notices**.

Edition notice

This edition applies to Version 3.5 of IBM VisualAge for Java and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Database access using SQLJ

SQLJ is a standard way to embed SQL statements in Java programs. SQLJ is less complex and more concise than JDBC. You can use both JDBC and SQLJ statements in the same source code.

The SQLJ standard has three components: embedded SQLJ, a translator, and a runtime environment. The translator translates SQLJ files that contain embedded SQLJ to produce .java files and *profiles* that use the runtime environment to perform SQL operations. The runtime environment usually performs the SQL operations in JDBC, and uses the profile to obtain details about database connections.

VisualAge for Java provides an SQLJ Tool that implements the SQLJ standard, enabling you to simplify database access. The translator component is integrated into the IDE, enabling you to import, translate, and edit SQLJ files. The runtime environment is an installable feature that is added to your workspace. The original .sqlj source files are maintained in your project resources directory, as are the profiles. You must set up the SQLJ Tool before you can use it for the first time.

You can find general SQLJ information at the Web Site http://www.sqlj.org. For information on SQLJ syntax, contact your database vendor. The IBM DB2® SQLJ Web page at http://www.software.ibm.com/data/db2/java/sqlj/ provides a link to the DB2 Application Programming Guide and Reference for Java,which includes reference documentation for SQLJ statements.

Embedded SQLJ
You write SQLJ programs using SQL statements preceded with the #sql token. These SQL statements are static; that is, they are predefined and do not change at run time. The counterpart to static SQL is dynamic SQL, a call interface for passing strings to a database as SQL commands. No analysis or checking of those strings is done until the database receives them at execution time. A dynamic SQL API for Java, called JDBC, has been specified by JavaSoft.

SQLJ programs are made up of SQLJ files which are Java source files that mix standard Java code with #sql statements. Each SQLJ file must have a .sqlj extension.

You cannot add embedded SQLJ statements into your source code in a Source pane in VisualAge for Java. To start using the SQLJ Tool, you must create a .sqlj file with a text editor, and then import and translate the file.

SQLJ translator
The SQLJ translator replaces SQLJ statements in an SQLJ file with calls to the SQLJ runtime environment, creating .java files and profiles. Java programs containing embedded SQL can be subjected to static analysis of SQL statements for the purposes of syntax checking, type checking, and schema validation.

The VisualAge for Java SQLJ tool copies your SQLJ file into a project's resource directory before translating it. The SQLJ tool then translates the SQLJ statements, producing Java source code and profiles. The translated source code is then imported into the project.

SQLJ runtime environment
The SQLJ runtime environment implements SQL operations in real time. The implementation is typically done through JDBC, but this is not mandatory. The runtime environment refers to the profiles generated by the translator for details about the connections to a database schema.

The runtime environment is an installable feature in VisualAge for Java. The runtime environment must be added to your workspace before you can successfully compile and execute translated SQLJ code. The VisualAge for Java SQLJ Tool uses JDBC in the runtime environment to implement SQL operations in real time.

SQLJ profiles
In addition to .java files, the SQLJ translator generates profiles. These files provide details about the embedded SQL operations in the SQLJ source code, including types and modes of data being accessed. Typically, one profile is generated per connection context class. A connection context class usually corresponds to a single database schema. The profiles are used by the runtime environment to provide details about the database schema.

Profiles must be customized to make use of vendor-specific features. The main advantages to customization are vender-supported datatypes and syntax, and vendor optimizations.

The VisualAge for Java SQLJ Tool creates profiles in your project resources directory. These binary files have a .ser extension.

RELATED CONCEPTS

Resource files and directories

RELATED TASKS

Setting up SQLJ
Creating an SQLJ file
Importing and translating an SQLJ file
Translating an imported SQLJ file
Editing an SQLJ file
Keeping your SQLJ file and Java code synchronized
Creating an SQLJ Debug class file
Customizing an SQLJ profile
Changing the SQLJ translator class
Resource files and directories

Chapter 2. Before you begin

Setting up SQLJ

To set up SQLJ for your project:

1. Add the SQLJ Runtime Library feature to your workspace.
2. Add the SQLJ Runtime Library project to your project's Class Path. See the related task in the list of links below.
3. (Optional) Enable online semantics checking.

Once you have set up SQLJ for your project, you need to create a new .sqlj file or import and translate an existing .sqlj file.

Adding the SQLJ Runtime Library feature
Before you can use SQLJ in your project, you need to add the SQL Runtime Library feature to your workspace. You can import and translate SQLJ files without the SQLJ Runtime Library feature, however you will not be able to compile the files.

To add the SQLJ Runtime Library feature:

1. In the Workbench, or a browser, select **File** > **Quick Start**, then select **Features**.
2. Select **Add Feature**, and click **OK**.
3. Select **SQLJ Runtime Library**, and click **OK**.

Enabling online semantics checking
When an SQLJ file is translated, you can have the SQLJ translator validate your SQL statements by comparing them to your database. This ensures your SQL statements can be performed on your database.

To enable online semantics checking:

1. In the Workbench, or a browser, select **Workspace** > **Tools** > **SQLJ** > **Properties**.
2. Select **Perform online semantic checking**.
3. Type the name of the JDBC driver that you use to connect to your database in the **JDBC Driver** field. For example, the DB2 driver is COM.ibm.db2.jdbc.app.DB2Driver.
4. Type the URL of your database in the **Default URL** field. For example, the DB2 sample database URL is jdbc.db2.sample.
5. Type the user ID and password to connect to the database in the **User** and **Password** fields.
6. Click **OK**.

To enable online semantics checking for DB2, do the above procedure, typing COM.ibm.db2.jdbc.app.DB2Driver in the **JDBC Driver** field. Then append the directory x:\\sqllib\\java\\db2java.zip to the additionalclasspath option of the SQLJTranslatorSupportToolProperties file, where x:\\sqllib is the directory where you installed DB2.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting the Class Path
Creating an SQLJ file
Importing and translating an SQLJ file
Translating an imported SQLJ file
Editing an SQLJ file
Setting SQLJ translation options

RELATED REFERENCES

SQLJTranslatorSupportToolProperties file

Chapter 3. Component tasks

Creating an SQLJ file

Once you have set up SQLJ, you need to create a new SQLJ file. You cannot add #sqlj statements directly to your source code in a Source pane. You have to create a new .sqlj file on your file system with a text editor.

To create an SQLJ file:

1. In a text editor, create a new file.
2. Add your Java source code and SQLJ statements.
3. Save your file with a .sqlj extension, and exit the editor.

Once you have completed creating a new .sqlj file, you need to import the file into your project and translate the file.

For information on SQLJ syntax, contact your database vendor. The IBM DB2 SQLJ Web page at http://www.software.ibm.com/data/db2/java/sqlj/ provides a link to the DB2 Application Programming Guide and Reference for Java,which includes reference documentation for SQLJ statements.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Importing and translating an SQLJ file
Translating an imported SQLJ file
Editing an SQLJ file
Keeping your SQLJ file and Java code synchronized

Importing and translating an SQLJ file

Once you have created an SQLJ file, you need to import your SQLJ file into a project and translate the file, before you can use it. Importing an SQLJ file places it in your project resources directory. Translating the SQLJ file creates at least one Java class and at least one profile. The Java class file is imported into your project, and the profile is placed in the project resources directory.

Before importing and translating your SQLJ file, you must have already set up your workspace with the SQLJ Runtime Library feature, and you must have created a .sqlj file with a text editor. You cannot add embedded SQLJ statements to your code in a Source pane .

To import an SQLJ file:

1. Select **Workspace** > **Tools** > **SQLJ** > **Import**. If this is the first time you are importing an SQLJ file, the SQLJ Properties window is displayed.
2. Type a project name in the **Project Name** field, or click **Browse** to select a project.
3. Type an SQLJ file name in the **SQLJ file name** field, or click **Browse** to select a file.

4. Select **Perform translation** to translate the SQL file after importing.
5. Click **Ok** to import the SQLJ file into the project.

Import status messages are displayed in the Console.

To ensure you are working with only one copy of your .sqlj file, you should remove the original file from your file system. Edit the imported .sqlj file from the Resources page of your Project browser.

RELATED CONCEPTS

Database access with SQLJ

RELATED TASKS

Setting up SQLJ
Creating an SQLJ file
Setting SQLJ translation options
Translating an imported SQLJ file
Editing an SQLJ file
Keeping your SQLJ file and Java code synchronized

Translating an imported SQLJ file

Once you have imported an SQLJ file into your project, you can translate it from the Resources page of the Project browser. Normally you translate SQLJ files when you import them, however you are not required to do so. Also, each time you edit an imported SQLJ file you need to translate it to ensure that the SQLJ file in your project resources directory and the source code in your project are synchronized.

To translate an imported SQLJ file:

1. In the Workbench, right click on your project and select **Open**.
2. Select the Resources tab.
3. Right click the .sqlj file you want to translate, and select **Tools** > **SQLJ** > **Translate**.

Translation status messages are displayed in the Console.

RELATED CONCEPTS

Database access using SQLJ
Resource files and directories

RELATED TASKS

Setting up SQLJ
Creating an SQLJ file
Importing and translating an SQLJ file
Editing an SQLJ file
Keeping your SQLJ file and Java code synchronized
Setting SQLJ translation options

Editing an SQLJ file

Once you have imported your SQLJ file into a project, you will probably want to edit it to reflect changes in your database, and to fix problems in your code. Rather than importing a new SQLJ file, you can edit the file directly from the Resources page of the Project browser.

To edit an SQLJ file:

1. In the Workbench, right-click your project and select **Open**.
2. Select the Resources tab.
3. From the .sqlj file's pop-up menu, select **Tools** > **SQLJ** > **Edit**.
4. Edit the file, save it, and exit the editor.
5. From the .sqlj file's pop-up menu, select **Tools** > **SQLJ** > **Translate**.

The last step ensures that you are working with the latest source code in your project. You should translate your .sqlj file *every* time you edit it.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Creating an SQLJ file
Importing and translating an SQLJ file
Keeping your SQLJ file and Java code synchronized

Keeping your SQLJ file and Java code synchronized

When you edit your SQLJ file from the Resource page of the Project browser, it is not automatically translated into Java code for you. Keeping your SQLJ files and Java code synchronized is a manual process.

To synchronize your SQLJ file with your Java code, right-click the SQLJ file in the Resources page of the Project browser and select **Tools** > **SQLJ** > **Translate**.

You should synchronize your SQLJ file with your Java source *every* time you edit your SQLJ file.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Importing and translating an SQLJ file
Editing an SQLJ file

Setting SQLJ translation options

You use the SQLJ Properties window to specify options that you want to use during the translation of your SQLJ file. You can specify:

- the file encoding type
- the option to check your SQL against a database

To view the SQLJ Properties window, from the Workbench or browser, select **Workspace** > **Tools** > **SQLJ** > **Properties**.

Encoding type
The **Encoding** field specifies the NLS encoding to be used on the source code produced by the translator.

Semantics checking
When your SQLJ file is translated, you can check the validity of your SQL semantics against your database. Selecting **Perform online semantics checking** will let the SQLJ translator perform this validity check against your database.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Importing and translating an SQLJ file
Translating an imported SQLJ file

Creating an SQLJ debug class file

For debugging purposes, you can create a .class file that refers to the original .sqlj file, rather than to the intermediate Java source code. When you use the stand-alone debugger on this .class file, the debugger will display the .sqlj source file.

To create an SQLJ debug class file:

1. In the Workbench, from your project's pop-up menu select **Open**.
2. Select the Resources tab.
3. From the .sqlj file's pop-up menu, select **Tools** > **SQLJ** > **Create SQLJ Debug Class File**.
4. Click **OK**, and select **Window** > **Refresh** to view the .class file in the Resources page.

The .class file will have the same name as your .sqlj file, and is located in your project resources directory.

RELATED CONCEPTS

Database access using SQLJ
Resource files and directories

RELATED TASKS

Setting up SQLJ
Debugging during the development cycle with the integrated debugger
Setting breakpoints in external classes

Customizing an SQLJ profile

When an SQLJ file is translated, a profile is created in your project resources directory that contains information about the SQL statements that you want to execute. To use database vendor-specific features in your SQL statements, you need to customize the profile. Your database vendor will provide you with an application to customize your profile.

RELATED CONCEPTS

Database access using SQLJ
Resource files and directories

RELATED TASKS

Setting up SQLJ

Changing the SQLJ translator class

You can use a different SQLJ translator class than the one provided with the SQLJ Tool. Your database vendor may provide an updated SQLJ translator class. To change the SQLJ translator class, you need to update the SQLJTranslatorSupportToolProperties file. This file tells the SQLJ Tool where to find the translator classes.

To change the SQLJ translator class:

1. Open the file x:\IBMVJava\ide\tools\com-ibm-ivj-sqlj\SQLJTranslatorSupportToolProperties.properties in a text editor, where x:\IBMVJava is the directory where VisualAge for Java is installed.
2. Append the directory or file name of your database vendor's translator class to the `additionalclasspath` option.
3. Change the value of the `translatorclassname` option to the name specified by your database vendor.
4. Change the value of the `translatormethodname` option to the value specified by your database vendor.
5. Save your changes and close the text editor.

Contact your vendor for the SQLJ translator class and method names.

After updating the SQLJTranslatorSupportToolProperties.properties file to use the DB2 translator, the file looks like the following.

```
additionalclasspath = x:\\sqllib\\java\\sqlj.zip
translatorclassname = sqlj.tools.Sqlj
translatormethodname = statusMain
```

The directory x:\\sqllib is where you have installed DB2.

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Importing and translating an SQLJ file
Translating an imported SQLJ file

RELATED REFERENCES

SQLJTranslatorSupportToolProperties file

Appendix. SQLJ properties file

The SQLJTranslatorSupportToolProperties.properties file contains SQLJ Tool options that are changed infrequently. Modify this file to specify the translator and profile customizer classes that your database uses. The file is provided with default options.

Options you can modify are:

- additional class path (additionalclasspath)
- translator class name (translatorclassname)
- translator method name (translatormethodname)

The file is located in the x:\IBMVJava\ide\tools\com-ibm-ivj-sqlj directory, where x:\IBMVJava is the directory where you have installed VisualAge for Java.

Additional class path
The additional class path option is used to append another classpath to the existing classpath when the translation is performed. This allows the translator to find the location of a translator class. When adding files or directories to the additionalclasspath option, the files and directories are separated by semicolons.

For example, to use the DB2 Java classes, the additionalclasspath option would look like

```
additionalclasspath=lib\\sqlj-translator.zip;x:\\sqllib\\java\\db2java.zip
```

where x:\\sqllib is the directory where you installed DB2. For other databases, you would replace `x:\\sqllib\\java\\db2java.zip` with the classpath specified by your database vendor.

The `lib\\sqlj-translator.zip` path specifies the location of the default translator class files. Do not replace this entry unless your database vendor instructs you to do so.

Translator class name
The translatorclassname option specifies the class used to perform the translation of your SQLJ files, and the optional semantic check. The default entry is:

```
translatorclassname=sqlj.tools.Sqlj
```

Translator method name
The translatormethodname option specifies the method within the translator class that is called to translate an SQLJ file into its component class and profile files. The default entry is:

```
translatormethodname=statusMain
```

The method must be declared `public static` and have only one argument of type `String[]`.

Default SQLJTranslatorSupportToolProperties.properties file
The SQLJTranslatorSupportToolProperties.properties file that comes with the SQLJ Tool looks like the following.

```
additionalclasspath = lib\\sqlj-translator.zip
translatorclassname = sqlj.tools.Sqlj
translatormethodname = statusMain
```

RELATED CONCEPTS

Database access using SQLJ

RELATED TASKS

Setting up SQLJ
Changing the SQLJ translator class

Notices

Note to U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd.
1150 Eglinton Avenue East
Toronto, Ontario M3C 1H7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow the customer to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

- AIX
- AS/400
- DB2
- CICS
- CICS/ESA
- IBM
- IMS
- Language Environment
- MQSeries
- Network Station
- OS/2
- OS/390
- OS/400
- RS/6000
- S/390
- VisualAge
- VTAM
- WebSphere

Lotus, Lotus Notes and Domino are trademarks or registered trademarks of Lotus Development Corporation in the United States, or other countries, or both.

Tivoli Enterprise Console and Tivoli Module Designer are trademarks of Tivoli Systems Inc. in the United States, or other countries, or both.

Encina and DCE Encina Lightweight Client are trademarks of Transarc Corporation in the United States, or other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

ActiveX, Microsoft, SourceSafe, Visual C++, Visual SourceSafe, Windows, Windows NT, Win32, Win32s and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, or other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Intel and Pentium are trademarks of Intel Corporation in the United States, or other countries, or both.

Other company, product, and service names, which may be denoted by a double asterisk(**), may be trademarks or service marks of others.