Using Java™ DataBase Connectivity

Sun™ ONE Studio 5 Programming Series

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 817-2332-10
June 2003, Revision A

Send comments about this document to: `docfeedback@sun.com`

Please
Recycle

Adobe PostScript™

Contents

Figures

Tables

Before You Begin

Welcome to the *Using Java DataBase Connectivity* book of the Sun ONE Studio 5 Programming Series. This book focuses on programming with persistent data—data stored in a database or other data store that is external to your applications. The book discusses the different persistence programming models supported by Sun ONE Studio 5. It focuses on the Transparent Persistence technology provided by the Sun ONE Studio 5 integrated development environment (IDE).

This book is written for programmers who want to learn how to use the persistence programming models supported by Sun ONE Studio 5. The book assumes a general knowledge of Java and database access technology. Before reading it, you should be familiar with the following subjects:

- Java programming language
- Relational database concepts (such as tables and keys)
- How to use the chosen database

You can create the examples in this book in the environments listed in the release notes on the following web site:

`http://forte.sun.com/ffj/documentation/index.html`

Screen shots vary slightly from one platform to another. You should have no trouble translating the slight differences to your platform. Although almost all procedures use the Sun™ ONE Studio 5 user interface, occasionally you might be instructed to enter a command at the command line. Here too, there are slight differences from one platform to another. For example, a Microsoft Windows command might look like this:

```
c:\>cd MyWorkDir\MyPackage
```

A UNIX command might look like this:

```
% cd MyWorkDir/MyPackage
```

Before You Read This Book

This book is written for programmers who want to learn how to use the persistence programming models supported by Sun ONE Studio 5. The book assumes a general knowledge of Java and database access technology. Before reading it, you should be familiar with the following subjects:

- Java programming language
- Relational database concepts (such as tables and keys)
- How to use the chosen database

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document and does not endorse and is not responsible or liable for any content, advertising, products, or other materials on or available from such sites or resources. Sun will not be responsible or liable for any damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services available on or through any such sites or resources.

How This Book Is Organized

This book describes JDBC™ productivity enhancement tools provided by Sun ONE Studio 5. These automate many JDBC programming tasks in building client components or applications that interact with a database.

Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`**
		`Password:`

AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line variable; replace with a real name or value	To delete a file, type `rm` *filename.*

Related Documentation

Sun ONE Studio 5 documentation includes books delivered in Acrobat Reader (PDF) format, release notes, online help, readme files for example applications, and Javadoc™ documentation.

Documentation Available Online

The documents described in this section are available from the `docs.sun.com`SM web site and from the documentation page of the Sun ONE Studio Developer Resources portal (`http://forte.sun.com/ffj/documentation`).

The `docs.sun.com` web site (`http://docs.sun.com`) enables you to read, print, and buy Sun Microsystems manuals through the Internet. If you cannot find a manual, see the documentation index installed with the product on your local system or network.

- Release notes (HTML format)

  Available for each Sun ONE Studio 5 edition. Describe last-minute release changes and technical notes.

  - *Sun ONE Studio 5, Standard Edition Release Notes* - part no. 817-2337-10

- Getting Started guides (PDF format)

  Describe how to install the Sun ONE Studio 5 integrated development environment (IDE) on each supported platform and include other pertinent information, such as system requirements, upgrade instructions, application server information, command-line switches, installed subdirectories, database integration, and information on how to use the Update Center.

  - *Sun ONE Studio 5, Standard Edition Getting Started Guide* - part no. 817-2318-10
  - *Sun ONE Studio 4, Mobile Edition Getting Started Guide* - part no. 817-1145-10

- Sun ONE Studio 5 Programming series (PDF format)

  This series provides in-depth information on how to use various Sun ONE Studio 5 features to develop well-formed J2EE applications.

  - *Building Web Components* - part no. 817-2334-10

    Describes how to build a web application as a J2EE web module using JSP pages, servlets, tag libraries, and supporting classes and files.

  - *Building J2EE Applications* - part no. 817-2327-10

    Describes how to assemble EJB modules and web modules into a J2EE application, and how to deploy and run a J2EE application.

  - *Building Enterprise JavaBeans Components* - part no. 817-2330-10

    Describes how to build EJB components (session beans, message-driven beans, and entity beans with container-managed or bean-managed persistence) using the Sun ONE Studio 5 EJB Builder wizard and other components of the IDE.

  - *Building Web Services* - part no. 817-2334-10

    Describes how to use the Sun ONE Studio 5 IDE to build web services, to make web services available to others through a UDDI registry, and to generate web service clients from a local web service or a UDDI registry.

  - *Using Java DataBase Connectivity* - part no. 817-2332-10

    Describes how to use the JDBC productivity enhancement tools of the Sun ONE Studio 5 IDE, including how to use them to create a JDBC application.

- Sun ONE Studio 5 tutorials (PDF format)

  These tutorials demonstrate how to use the major features of each Sun ONE Studio 4 edition.

  - *Sun ONE Studio 5 Web Application Tutorial* - part no. 817-2320-10

    Provides step-by-step instructions for building a simple J2EE web application.

- *Sun ONE Studio 5 J2EE Application Tutorial* - part no. 817-2322-10

  Provides step-by-step instructions for building an application using EJB components and Web Services technology.
- *Sun ONE Studio 4, Mobile Edition Tutorial* - part no. 816-7873-10

  Provides step-by-step instructions for building a simple application for a wireless device, such as a cellular phone or personal digital assistant (PDA). The application will be compliant with the Java 2 Platform, Micro Edition (J2ME™ platform) and conform to the Mobile Information Device Profile (MIDP) and Connected, Limited Device Configuration (CLDC).

You can also find the completed tutorial applications at:
`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

Online Help

Online help is available in the Sun ONE Studio 5 IDE. You can open help by pressing the help key (F1 in Microsoft Windows and Linux environments, Help key in the Solaris environment), or by choosing Help → Contents. Either action displays a list of help topics and a search facility.

Examples

You can download examples that illustrate a particular Sun ONE Studio 5 feature, as well as completed tutorial applications, from the Sun ONE Studio Developer Resources portal at:

`http://forte.sun.com/ffj/documentation/tutorialsandexamples.html`

The site includes the applications that are used in this document.

Javadoc Documentation

Javadoc documentation is available within the IDE for many Sun ONE Studio 5 modules. Refer to the release notes for instructions on installing this documentation. When you start the IDE, you can access this Javadoc documentation within the Javadoc pane of the Explorer.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Email your comments to Sun at this address:

`docfeedback@sun.com`

Please include the part number (817-2332-10) of your document in the subject line of your email.

Using Java DataBase Connectivity

Sun ONE Studio 5 provides a JDBC (Java Database Connectivity) module that automates many programming tasks that you use when building client components or applications that interact with a database.

The goal of the Sun ONE Studio 5 JDBC module is to increase your productivity when programming visual forms that contain Swing (Java Foundation Class) components that use JDBC to retrieve and update database tables. You can use this module to assist you in generating simple, two-tiered application architectures.

This chapter describes the following JDBC productivity enhancement tools provided by Sun ONE Studio 5, and begins with a brief description of the steps you follow in creating a JDBC application. The tools include:

- Database Explorer
- JDBC JavaBeans components
- JDBC Form Wizard

Programming JDBC

This section provides a brief introduction to JDBC programming tasks.

General Programming Steps

When you perform JDBC programming, you follow these general programming steps:

1. Import relevant classes within your code.
2. Load a JDBC driver.

3. Establish a connection with a database.
4. Create a Main method.
5. Create try and catch blocks and retrieve exceptions and warnings.
6. Set up and use database tables.
    a. Create a table.
    b. Create JDBC statements.
    c. Execute Statements to perform persistence operations.
        i. Enter data into a table.
        ii. Obtain data from a table.
        iii. Create an updatable result set (`RowSet`).
        iv. Insert and delete rows programmatically.
    d. View changes in a `ResultSet` by managing the Transaction Isolation Level.

Sun ONE Studio 5 simplifies most of these tasks, generating JDBC code either through your editing of the Sun ONE Studio 5 JDBC JavaBeans component properties or through your use of the JDBC Form Wizard.

JDBC Reference Materials

While this chapter provides a discussion of JDBC programming in the context of the Sun ONE Studio 5 IDE, it assumes familiarity with the basics of the JDBC programming model. For additional information about JDBC, you can review the following reference materials, grouped by function.

Learning JDBC Programming

The Java Developer Connection provides an excellent tutorial on JDBC:

`http://developer.java.sun.com/developer/onlineTraining/new2java/programming/learn/jdbc.html`

In addition, the Java Developer Connection supplies a JDBC Short Course:

`http://developer.java.sun.com/developer/onlineTraining/Database/JDBCShortCourse/index.html`

Technical Articles

Sun has produced a document entitled:

“Duke’s Bakery – A JDBC Order Entry Prototype – Part I”:

`http://developer.java.sun.com/developer/technicalArticles/`
`Database/dukesbakery/`

Getting Started With JDBC

The following index is a reference when starting to program using JDBC:

`http://developer.java.sun.com/developer/technicalArticles/`
`Interviews/StartJDBC/index.html`

Another document is “Of Java, Databases, and Really Cool Dead Guys”:

`http://developer.java.sun.com/developer/technicalArticles/`
`Interviews/Databases/index.html`

JDBC Basics

You can find additional information on JDBC within the Sun tutorial:

`http://java.sun.com/docs/books/tutorial/index.html`

This tutorial also provides some references:

`http://java.sun.com/docs/books/tutorial/jdbc/basics/index.html`

Using the Database Explorer

Before you begin the process of writing JDBC code, you need to understand the database that your application will use. To obtain database information, you can use the Sun ONE Studio 5 Database Explorer.

Using the Sun ONE Studio 5 Database Explorer, you can perform the following tasks:

- Browse database structures
- Examine all tables present in the database, including column and index information

- Examine SQL views related to the database
- Examine all stored procedures defined in the database
- View database data
- Create tables
- Create views
- Take “snapshots” of database structures
- Monitor SQL commands sent to the database
- Connect to a database

To learn how to perform these tasks, refer to the Database Explorer Help within the Sun ONE Studio 5 IDE.

Using JDBC Components

Sun ONE Studio 5 provides database connectivity and JDBC code generation tools for visual forms and components, specifically providing two basic types of components that you can use with your JDBC application:

- Visual Components—Swing components let you display tabular database information. Within Sun ONE Studio 5, use Swing visual components to create forms that relay database data to the user; swing components provide the means to let you manipulate row data and display columns. Sun ONE Studio 5 generates the appropriate Swing code for you. Another type of visual component is a Data Navigator–a JDBC component that you add to a form to manipulate the display of data to the user.
- Non-visual components—JavaBeans components that do not have visual representation, but can be used to manipulate data from a database. One type of non-visual component is a `RowSet`, which is a type of row group that contains information from the database. To understand how to use JDBC JavaBean components, you need to:
- Understand the JDBC tab
- Understand how to program applications with JDBC components by:
  - Creating a Visual Form with Sun ONE Studio 5
  - Using the Sun ONE Studio 5 Component Inspector with JDBC JavaBeans components

The JDBC Tab

The JDBC tab in the component palette contains icons for a number of JDBC JavaBeans components that you can use to facilitate the interaction of Java Swing components with a database. These components have properties that you customize using the Sun ONE Studio 5 Component Inspector.

The components include:

- `Connection Source`
- `Pooled Connection Source`
- `NB Cached RowSet`
- `NB JDBC RowSet`
- `NB Web RowSet`
- `Stored Procedure`
- `Data Navigator`

Connection Source

A Connection source is a non-visual component that provides a connection to a JDBC compliant database. When you configure the `Connection Source`, you set:

- database URL
- JDBC driver name
- user name
- password

Pooled Connection Source

A `Pooled Connection Source` component is similar to a `Connection Source`. However, when you specify the use of a `Pooled Connection Source` with your application, database connections that are established during application runtime are not closed when the application ceases to use the connection.

Instead, Sun ONE Studio 5 retains the connection in a pool for subsequent use within the runtime application. You can use a `Pooled Connection Source` when your application performs frequent open and close requests against a database to which it is connected.

Understanding RowSets

A RowSet component represents rows fetched from the database. You can use these components to configure data models for several Swing components.

RowSet *Background*

A RowSet object contains a set of rows from a JDBC result set or another source of tabular data, such as a file or spreadsheet.

Depending on how you implement them in your code, RowSets can be serializable or extensible to non-tabular sources of data.

Because a RowSet object follows the JavaBeans model for properties and event notification, it is a JavaBeans component that can be combined with other components in an application.

RowSets can be either connected or disconnected, depending on their implementation. A disconnected RowSet obtains a connection to a data source to fill itself with data or to propagate changes in data back to the data source, but most of the time it does not have a connection open.

Even when it is disconnected, a RowSet does not require the use of a JDBC driver or the full JDBC API, so its size is small. A disconnected RowSet is an ideal format for sending data over a network to a thin client.

Types of RowSets:

The JDBC Tab makes three different types of row sets available:

- NB Cached RowSet

  The NBCachedRowSet is a disconnected RowSet that caches its data in memory. This special type of RowSet is suitable for smaller sets of data. You can use it to create JDBC applications that provide code to operate on thin Java clients, such as Personal Digital Assistants (or PDAs).

  When a RowSet is disconnected from its data source, any updates that application writes on the RowSet are propagated to the underlying database.

- NB JDBC RowSet

  The NBJDBCRowSet represents a JavaBeans™ wrapping of a connected ResultSet object to be used in models of Swing components. It can be used to read extremely long tables more efficiently than a cached RowSet, which stores all data in an internal cache.

- NB Web RowSet

  The NBWebRowSet represents a set of fetched rows in a cache to be used in models of Swing components. It provides all cached RowSet functionality, and enables the rows to be imported and exported in XML format. The file can then be sent over the internet using HTTP/XML protocols.

You can customize a JDBC RowSet by setting the following properties under the properties tab in the Properties Editor:

TABLE 1 RowSet Properties

Property	Definition
Command	SQL query to populate this RowSet. The query can be any syntactically-correct SQL Select Query.
Connection provider	The configured connection source; a drop-down list provides choices.
Read-only	If True, this RowSet is read-only. Data from the RowSet cannot be written out to the database.
Rowcount	The number of rows.
Status	Status of a read against a RowSet
Transaction isolation	determines how the RowSet handles data under transactions. For detail, see Java documentation for java.sql.Connection.
XML output directory (WebRowSet only)	Identifies the directory where data from the WebRowSet will be sent.
XML Output File (WebRowSet only)	Determines the name of the file that will contain the XML output from a WebRowSet.

Other Properties, Event, and Code Generation Tabs for a RowSet

The Other Properties Tab for a RowSet enables you to inspect and modify additional properties.

TABLE 2 RowSet Other Properties Tab Properties

Property	Definition
Database URL	The location of the database where records will be updated. In most cases, it is the same URL as listed in the Database URL property of Connection Source.
Default Column Values	The values to be inserted into a new row. You can press Fetch Columns to retrieve a list of columns in the RowSet.
Execute on load	If true, the NB RowSet can be executed on load. You can specify a parameter with the Execute on Load from a Form Connection, and you can generate initialization code.

TABLE 2 `RowSet` Other Properties Tab Properties *(Continued)*

Property	Definition
Password	A password the user must supply to gain access to the table that contains this `NB RowSet`.
Table Name	The name of a database table where records will be updated.
User Name	The name of a user updating records.

The Event Tab for a RowSet enables you to inspect and modify events associated with `RowSet`s.

TABLE 3 `RowSet` Event Tab Properties

Property	Definition
`cursorMoved`	Specifies event handlers for the `cursorMoved` event. This method is called when an `NBCachedRowSet`'s cursor is moved.
`rowChanged`	Specifies event handlers for the `rowChanged` event. This method is called when a row in a `RowSet` is changed.
`rowInserted`	Specifies event handlers for the `rowInserted` event. This method is called when a row in a `RowSet` is inserted.
`rowSetChanged`	Specifies event handlers for the `rowSetChanged` event. This method is called when an `RowSet` is changed.
`rowCompleted`	Specifies event handlers for the `rowCompleted` event. This method is called after an inserted row is committed to the database.

The Code Generation Tab enables you to specify pre- and post-processing code related to a rowset.

TABLE 4 Code Generation Tab Properties

Property	Definition
Code Generation	Choose between generating standard or serialization code for the component.
Custom Creation Code	Enter your own creation code for the component, not including the variable name and equal sign (=). This creation code is called in the `initComponents()` method. If this property is left blank, the IDE generates a default creation code for the component.
Post-Creation Code, Post-Init Code, Pre-Creation Code, and Pre-Init Code	Write custom code that you want the IDE to place before and after a component's creation code and before and after its initialization code. The IDE always places creation code before initialization code in `initComponents()`.

TABLE 4 Code Generation Tab Properties *(Continued)*

Property	Definition
Serialize To	Set the name of the file for the component to be serialized to, if it is serialized.
Use Default Modifiers	Set to `True` if you want the component's variable modifiers (public, private, and so on) to be generated using the default modifiers. The default modifiers are specified in the Variables Modifier property of the Form Objects node in the Options window. (Choose Tools → Options to view the window.) Set to `False` if you want the Variables Modifier property to appear on the component's property sheet, enabling you to override the default modifiers.
Variable Name	Modify the component's variable name.

Data Navigator

The JDBC module provides a visual component that provides direct navigation of a `RowSet` with a pre-built GUI. This component is useful when you need to create prototypical applications and when you want to create data entry applications.

You can customize a Data Navigator by setting the following properties under the properties tab in the Properties Editor of a Data Navigator.

TABLE 5 Data Navigator Properties

Property	Definition
AutoAccept	Automatically accept changes in the database. When you specify this property, changes you make through the Navigator are either immediately propagated to the database, or added to the RowSet and propagated to the database when you request it.
Bound RowSet	The `RowSet` to be controlled by the Data Navigator.
Layout of buttons	Determines whether buttons are displayed in one or two rows.
Modification buttons	Enables or disables the display of buttons for modification.

Stored Procedures

Stored procedures are a group of SQL statements that form a logical unit and perform a specific task. Stored procedures encapsulate operations or queries that execute on a database server. Such procedures, of course, vary in their nature according to the database management system (DBMS) on whose server they execute.

Within the Sun ONE Studio 5 IDE, a stored procedure is a non-visual component that represents a database stored procedure in your JDBC application. You can call a stored procedure in response to an event initiated by a user within an application GUI (such as a button click).

The syntax for a stored procedure is different for each database management system that Sun ONE Studio 5 supports. For example, one database management system might use `begin`, `end`, or additional keywords to indicate the beginning and ending of the procedure definition, while a second DBMS might use other keywords to indicate the same parts of the procedure definition.

The *JDBC Tutorial* provides information on some of the stored procedures you can create for different databases, in addition to information on calling a stored procedure from your JDBC application.

You can customize a stored procedure by setting the following properties under the properties tab in the Properties Editor of a stored procedure. Once you have specified these properties in the property sheet, you can connect stored procedures to any user action.

TABLE 6 Stored Procedure Properties

Property	Definition
Arguments	Represents database data that you want used by the stored procedure when called from the application.
Bound RowSet	Enables you to select a RowSet from a drop-down list that is refreshed from the database after the stored procedure is called.
Call format	Format in which your stored procedure is called. For example, it might include Name and Arguments that are substitution codes for the properties with those names on this property sheet.
Connection provider	A configured connection source in whose context the stored procedure is to be called from the application.
Name	The name of your called stored procedure.

Programming With JDBC Components

Use the visual and non-visual components provided in the JDBC module in conjunction with Swing components to create forms that you use to retrieve and manipulate database data.

For example, a number of Swing components (`JList`, `JTable`, `JComboBox`, `JButton`, `JToggleButton`, `JRadioButton`, `and JCheckbox`) are associated with data models for the data they display. Within the IDE, you use Property Editors and the Component Inspector to customize the data model for these Swing components by specifying the JDBC components with which they interact to access a database. After you have completed specifying the JDBC components, Sun ONE Studio 5 generates the corresponding JDBC code.

Setting Data Models for Components

The following Swing components have associated data models.:

- `JList`
- `JTable`
- `JComboBox`
- `JButton`
- `JToggleButton`
- `JRadioButton`
- `JCheckbox`

You can configure these data models to use data from the database.

The most common component to display database tables is `JTable`. The model can be configured in the property sheet of each Swing component (under the model property).

Selecting Database Columns

Components that can display multiple rows, such as `JTable` or `JList`, also have the `selectionModel` property.

`JList` and `JComboBox` also have a special kind of model. This model consists of using one column from one RowSet to work with another column from another RowSet to display data, using a SQL join. See below for details.

Text components which have the document property (such as `JTextField`, `JTextArea`, `JPasswordField`, `JTextPane`, and `JEditorPane`) can set up this property to use data from the database.

▼ To Configure the Data Model for `JTable`

1. **For the model property in the `JTable`'s property sheet, open the custom property editor by clicking on the value of the property and then clicking the ellipsis (...) button that appears.**

2. **Choose the TableEditor mode.**

3. **In the RowSet field, choose the RowSet to be displayed in the table.**

4. **Use Fetch columns to load column names into the list.**

5. **Use the Add, Remove, Edit, Move Up, and Move Down buttons to set the names and order of the columns in the table.**

6. **Click OK to preserve the changes and close the custom property editor.**

▼ To Configure the Selection Model for `JTable` and `JList`

1. **For the selectionModel property in the component's property sheet, open the custom property editor by clicking on the value of the property and then clicking the ellipsis button (...) that appears.**

2. **In the RowSet field, choose the RowSet to be displayed in the table or list.**

3. **Click OK to preserve the changes and close the custom property editor.**

▼ To Configure the Data Model for `JList` and `JComboBox`

1. **For the model property in the component's property sheet, open the custom property editor (by clicking on the value of the property and then clicking the ellipsis button (...) that appears).**

2. **For the Primary RowSet fields, choose the RowSet for the data model to retrieve rows from, and then select one column from the Column drop-down list.**

3. **If you want, in the Secondary RowSet field, choose the RowSet to display data from (according to a SQL join). Corresponding columns from the primary and secondary RowSet must have the same data type.**

4. **If the Join check box is checked, a corresponding component displays the result of a database join. If it is unchecked, a corresponding component is used as a code map to set values in the primary rowset.**

5. **Choose a Data column (join column) and Display column (visible data). Click OK to preserve the changes and close the custom property editor.**

▼ To Configure the Data Model for `JCheckbox`, `JRadioButton`, and `JToggleButton`

1. **For the model property in the component's property sheet, open the custom property editor (by clicking on the value of the property and then clicking the ellipsis (...) button that appears).**

2. **Choose the RowSet from which the data is to be fetched.**

3. **Choose a column; data from this column will be used to decide if the component should be selected.**

4. **Enter the database value corresponding to a selected component into the Select field and the value of an unselected component into the Unselect field.**

5. **Click OK to preserve the changes and close the custom property editor.**

▼ To Configure the Document Model for Text Components

1. **For the document property in the component's property sheet, open the custom property editor by clicking on the value of the property and then clicking the ellipsis button (...) that appears.**
2. **Choose the RowSet from which the data is to be fetched.**
3. **Choose a column in which to display the text component.**
4. **Click OK to preserve the changes and close the custom property editor.**

Creating a Visual Form

After you have used the Property Editor to customize Swing components in your application, Sun ONE Studio 5 enables you to create a visual form associated with the Swing components that interacts with the database.

▼ To Create a Visual Form With Swing Components That Interact With a Database

1. **Create a Swing component form using a template provided in the Sun ONE Studio 5 IDE.**
2. **Add any needed** `Connection Source` **(or** `Pooled Connection Source`**),** `RowSet`**, or** `Stored Procedure` **nonvisual components to your form from the Component Palettes.**
3. **Using the corresponding Property Editor, customize these components for the database entities they represent.**
4. **Add any visual components you need, including the Data Navigator.**
5. **Use the corresponding Property Editor to customize the visual components appropriately, referencing the** `RowSet` **components you need.**

   As you specify the Swing components to use with your JDBC application, Sun ONE Studio 5 automatically creates the correct Swing classes to use in your application.
6. **Use the Properties Editor for the specified form to indicate exceptions that should be caught during runtime and run the form.**

Using the Component Inspector With JDBC Components

You can use the Sun ONE Studio 5 Component Inspector to modify properties for components you use in your JDBC application. The following components can be found under Non-visual Components in the Component Inspector:

- `NB Cached RowSet`
- `NB JDBC RowSet`
- `NB Web RowSet`
- `Connection Source`
- `Pooled Connection Source`
- `Stored Procedure`

The `Data Navigator` component and other Swing components are shown according to their position in the container hierarchy.

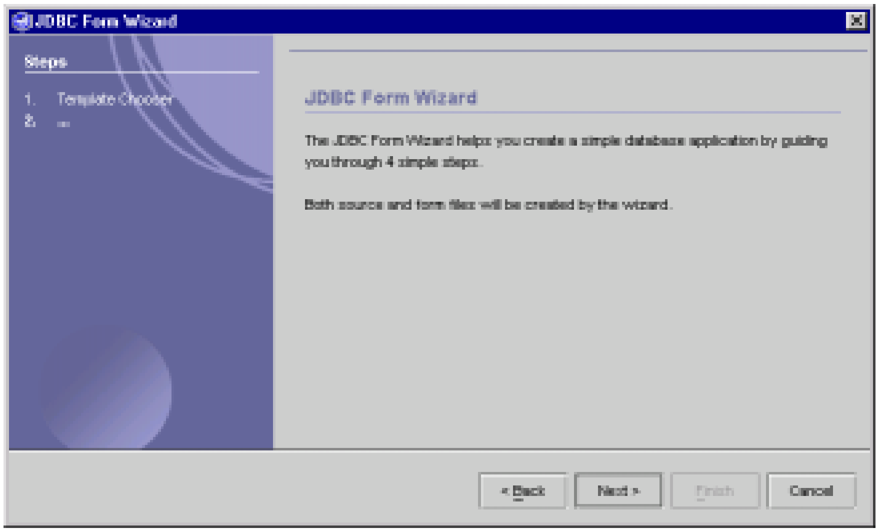
Using the JDBC Form Wizard

The JDBC Form Wizard guides you through the creation of a form that can interact with database tables. It provides a substitute for the explicit editing of properties that you would otherwise perform if you used the approach outlined in “Using JDBC Components” on page 18. When you finish running the wizard, you will have a generated application, a file name for the application, and a package.

The following sections illustrate the JDBC Form Wizard, using the sample PointBase Server Database that comes included with the Sun ONE Studio 5 IDE.

▼ To Open the JDBC Wizard

- **Select Tools → JDBC Form Wizard**



JDBC Form Wizard

The JDBC Form Wizard helps you create a simple database application by guiding you through 4 simple steps.

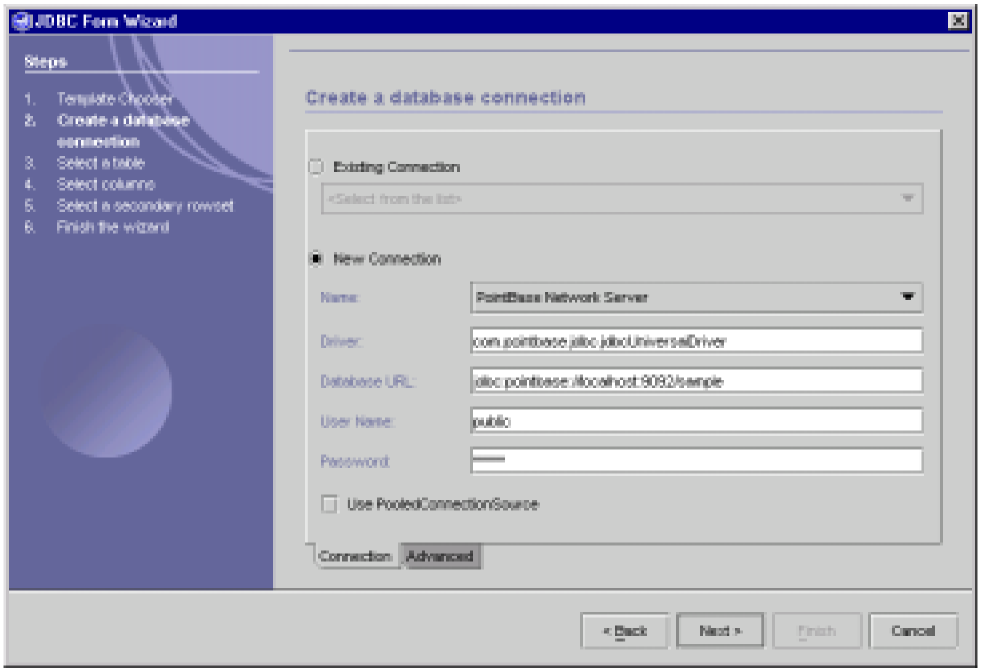
Both source and form files will be created by the wizard.

FIGURE 1 JDBC Form Wizard, Opening

Establishing a Connection

When you use the JDBC Form Wizard or when you use the JDBC tab to create a JDBC client application, one of the first tasks you must perform is to establish a connection with the database management system that you want to use.

Typically, the JDBC Form Wizard or Sun ONE Studio 5 connection generates the code that you can use in your JDBC application when you use the Visual Form Editor or the JDBC Form Wizard to create a form. The application uses the form to populate information that it obtains from a database management system.



Name: PointBase Network Server
Driver: com.pointbase.jdbc.jdbcUniversalDriver
Database URL: jdbc:pointbase://localhost:9092/sample
User Name: public
Password: ******

FIGURE 2 JDBC Form Wizard, Database Connection

The second panel of the JDBC Form Wizard lets you establish a connection with a database. You can specify the use of a pooled connection for a DataSource in this panel.

When you need a new connection, you must supply:

- The name of your database. For example, PointBase Network Server.
- The JDBC driver name for the database. For example, `com.pointbase.jdbc.jdbcUniversalDriver`.
- The Database URL where the database is located. For example, `jdbc.pointbase://localhost:9092/sample`.
- User Name
- Password
- Select the Use Pooled Connection Source check box to specify an optional pooled connection.
- Optionally select the Advanced tab to specify a schema to get tables.

Sun ONE Studio 5 provides these parameters to the JDBC application code that it generates.

You can select an existing connection by clicking the Use Existing Connection radio button, and selecting the connection from the drop-down list.

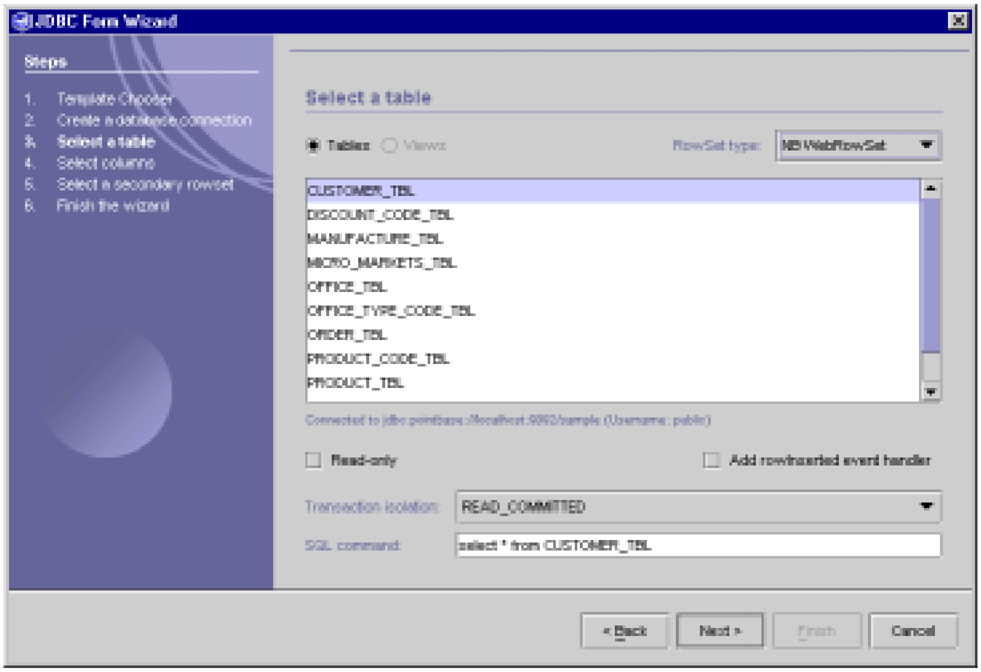
When you select the Next button, Sun ONE Studio 5 calls a method that creates a database connection based on parameters you enter. You use this connection to the database in the same way that you use the wizard to write JDBC application code.

Selecting Database Tables or Views

The third panel of the JDBC Form Wizard lets you:

- Select a table or view in the database to which you are connected.
- Specify that you want only read access to a specific table for your generated JDBC application. This means that the application cannot alter data in the database.
- Add a `rowInserted` event handler to a table. This event handler handles the listening for events associated with the application's insertion of rows into the tables you select.
- Set the Transaction Isolation level for a table. See "Transaction Isolation Levels" on page 32.
- Provide a SQL command to run against the tables you specify.

The JDBC Form Wizard lets you execute SQL statements against tables you specify in the Wizard. You use the data from the SQL output to populate visual forms. You can specify SQL statements which, when applied to a specific form, generate the appropriate SQL code. In FIGURE 3, Sun ONE Studio 5 provides a default SQL command to use with the table you have selected.



[✓ Tables] Views

RowSet type: NBWebRowSet

CUSTOMER_TBL
DISCOUNT_CODE_TBL
MANUFACTURE_TBL
MICRO_MARKETS_TBL
OFFICE_TBL
OFFICE_TYPE_CODE_TBL
ORDER_TBL
PRODUCT_CODE_TBL
PRODUCT_TBL

Connected to jdbc:pointbase://localhost:9092/sample (Username: public)

Read-only

Add rowInserted event handler

Transaction isolation: READ_COMMITTED

SQL command: select * from CUSTOMER_TBL

FIGURE 3 JDBC Form Wizard, Select a Table

Transaction Isolation Levels

To avoid conflicts during a transaction, a database management system uses *locks*. Locks are operative until the application commits the transaction or rolls it back from the database.

Locks are set according to a transaction isolation level. Locks apply to the entire `ResultSet` that is returned to the application or committed from the application to the database.

Each database management system provides its own default transaction isolation level. Sun ONE Studio 5 lets you choose between the transaction isolation levels within the second panel of the JDBC Form Wizard.

Note – The driver and the data base management system must support the transaction isolation level you use.

TABLE 7 Transaction Isolation Levels

Property	Definition
`TRANSACTION_READ_COMMITTED`	Prohibits a transaction from reading a row that has uncommitted changes in it.
`SERIALIZABLE`	Includes the prohibitions in `TRANSACTION_REPEATABLE_READ`. It prohibits the situation where one transaction reads all rows that satisfy a `WHERE` condition, a second transaction inserts a row that satisfies that `WHERE` condition, and the first transaction rereads for the same condition, retrieving the additional "phantom" row in the second read.
`TRANSACTION_NONE`	Transactions are not supported.
`TRANSACTION_REPEATABLE_READ`	Prohibits a transaction from reading a row with uncommitted changes in it. It also prohibits the situation where one transaction reads a row, a second transaction alters the row, and the first transaction rereads the row, getting different values the second time (that is, a non-repeatable read).
`TRANSACTION_READ_UNCOMMITTED`	A row changed by one transaction can be read by another transaction before changes in that row are committed to the database. If changes are subsequently rolled back, the second transaction retrieves an invalid row.

Selecting Columns to Display

The fourth panel of the JDBC Form Wizard lets you select columns from the database tables to include in the form that is displayed. In this panel, you can specify:

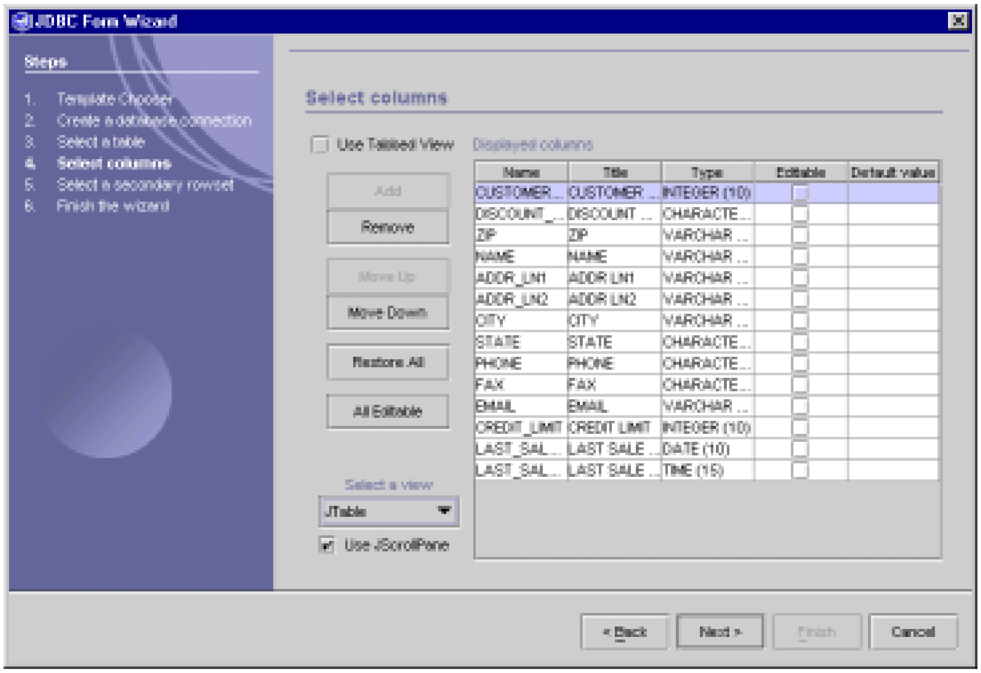
- Columns you want displayed in the application you generate
- The order of the columns you want displayed
- Column parameters:
  - Column title
  - Column editability
  - Default column value
  - A Swing component to display the table in the application

In the example provided, `JTable` (the most common Swing form) is used. The `JTable` form displays more than one column of data in the application.

Other Swing component choices include:

- `Jlist`: displays a column in a list
- `JComboBox`: displays one column in a combo box
- `JTextField`: displays one or more columns in a text field

In FIGURE 4, the first Column is selected. It can be removed or moved in position.



Name	Title	Type	Editable	Default value
CUSTOMER...	CUSTOMER ...	INTEGER (10)	☐	
DISCOUNT_...	DISCOUNT ...	CHARACTE...	☐	
ZIP	ZIP	VARCHAR ...	☐	
NAME	NAME	VARCHAR ...	☐	
ADDR_LN1	ADDR LN1	VARCHAR ...	☐	
ADDR_LN2	ADDR LN2	VARCHAR ...	☐	
CITY	CITY	VARCHAR ...	☐	
STATE	STATE	CHARACTE...	☐	
PHONE	PHONE	CHARACTE...	☐	
FAX	FAX	CHARACTE...	☐	
EMAIL	EMAIL	VARCHAR ...	☐	
CREDIT_LIMIT	CREDIT LIMIT	INTEGER (10)	☐	
LAST_SAL...	LAST SALE ...	DATE (10)	☐	
LAST_SAL...	LAST SALE ...	TIME (15)	☐	

FIGURE 4 JDBC Form Wizard, Select Columns

If you choose `JList` or `JComboBox`, only one column can be displayed, and you can choose a column to display from the Name property:

1. **Select a value in the Name column.**

2. **Select a column name from the built-in combo box.**

▼ To Edit Column Titles

1. **Click on the Title field you want to edit. An edit window appears with two tabs.**

2. **Select the String Value tab to enter the new name as a simple string value.**

3. **Select Resource Bundle to enter the name using a resource bundle. Enter the name of the bundle into Bundle Field, and select any related keys from the Keys combo box.**

4. **Select OK to close the edit window.**

Selecting a Secondary `RowSet`

This panel displays a list of all available tables according to the database connection created on the Connection panel and is enabled only if a view supporting two `RowSets` (`JList` of `JCheckbox`) is selected.

You can use this panel to populate the secondary `RowSet` of the generated application.

▼ To Select a Secondary `RowSet`

1. **Check Use Secondary Rowset.**

   If you check this rowset, the secondary rowset is used in the generated application.

2. **Select either the Tables or Views radio button.**

3. **Select a type of rowset from the RowSet type combo box.**

4. **Select a table or view from the list.**

5. **Check Read-only if you want the corresponding rowset to be read-only.**

6. **Check Add rowInserted event handler to add a rowInserted event handler to the source code of the generated application.**

   The handler is called when a new row is inserted and enables the creation of default column values dynamically.

7. **Choose a transaction isolation level for the rowset using one of the values in the Transaction isolation combo box.**

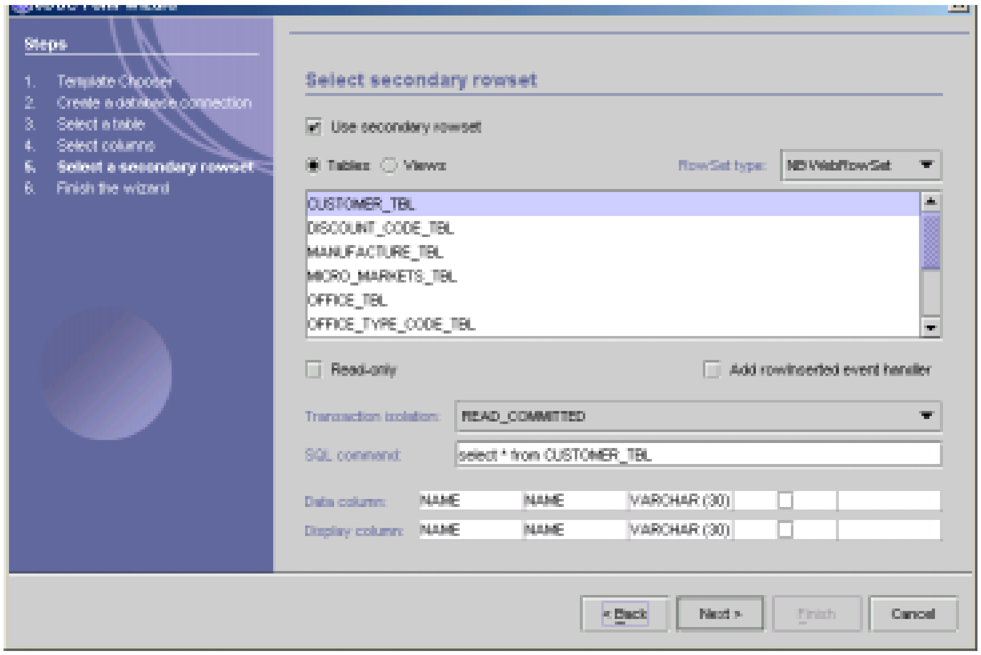
   The default transaction level is `READ_COMMITTED`.

8. **Use the SQL_command text field to prepare SQL to populate the rowset.**

   By default, Sun ONE Studio 5 generates the text `select * from table-name`.

9. **Select a data column to use with a database join.**

   Selecting this column will display a different field other than the primary column retrieved; however, it must be of the same data type as the primary column.



Select secondary rowset

[✓] Use secondary rowset

(●) Tables ( ) Views   RowSet type: NBWebRowSet

CUSTOMER_TBL
DISCOUNT_CODE_TBL
MANUFACTURE_TBL
MICRO_MARKETS_TBL
OFFICE_TBL
OFFICE_TYPE_CODE_TBL

[ ] Read-only   [ ] Add rowInserted event handler

Transaction isolation: READ_COMMITTED

SQL command: select * from CUSTOMER_TBL

Data column:	NAME	NAME	VARCHAR (30)	☐		
Display column:	NAME	NAME	VARCHAR (30)	☐		

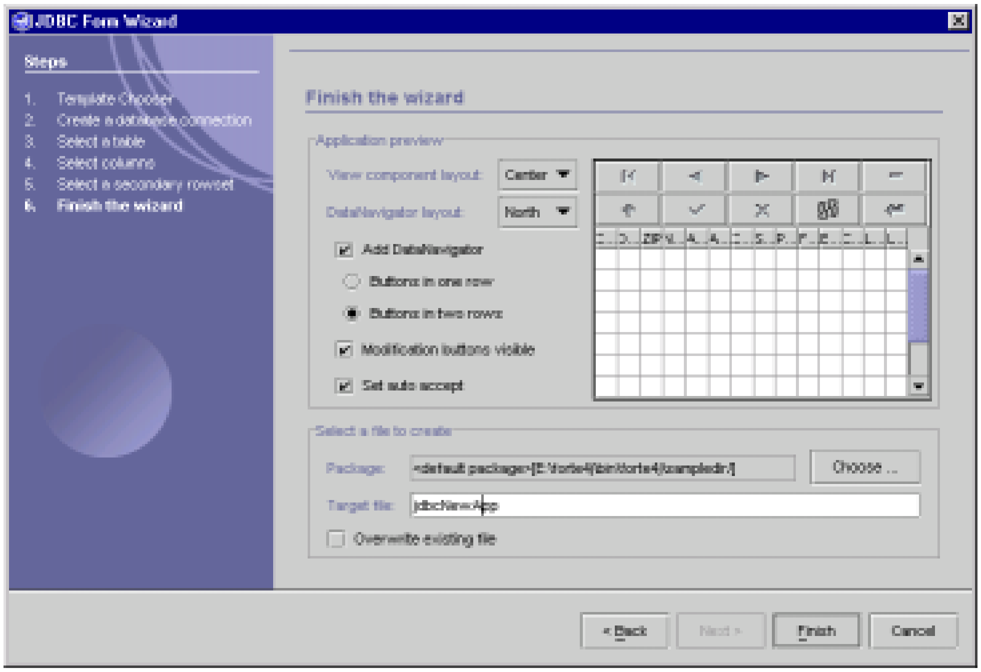
FIGURE 5 JDBC Form Wizard, Select Secondary RowSet

Previewing and Generating an Application

The last panel shows a preview of a generated application. Use this panel to complete your generated application. In addition, you can select a package and a file name to create a completed application.

Provide the name of the package under Package and the target file under Target.

You can view the component layout and the layout from the view of the Data Navigator. What you view depends on the Swing form you have chosen to contain the data that is manipulated in your application.



View component layout: Center
DataNavigator layout: North
[✓] Add DataNavigator
( ) Buttons in one row
(●) Buttons in two rows
[✓] Modification buttons visible
[✓] Set auto accept
Package: <default package>[E:\forte4j\bin\forte4j\sampledir\]
Target file: jdbcNewApp
[ ] Overwrite existing file

FIGURE 6 JDBC Form Wizard, Finish the Wizard

Running Your JDBC Application

You can compile, run, and debug JDBC applications as if they were any other form. If you need special JDBC drivers, ensure they are in the Sun ONE Studio 5 CLASSPATH, so they will, by default, be available for external compiling, executing, and debugging of JDBC-based forms.

You can run your application external to the IDE by adding paths to these packages into your CLASSPATH:

- `modules/ext/sql.jar`
- `modules/ext/rowset.jar`
- `lib/ext/jdbc20x.zip`
- A corresponding JDBC driver. JDBC drivers are typically stored in `lib/ext`.

If a `WebRowSet` is used in your JDBC application, two more JAR files are required:

- `lib/ext/parser.jar`
- `lib/ext/xerces.jar`

Index