Tutorial

Versata Logic Suite - Corba Edition

VERSATA, INC.
300 LAKESIDE DRIVE
SUITE 1500
OAKLAND, CA 94612-3534
PHONE: 510.238.4100
INTERNET: HTTP://WWW.VERSATA.COM
VS55C-TUT-01

Copyright

Table of Contents

Preface

Versata Logic Suite documentation

The Versata Logic Suite documentation is electronically provided in `.pdf` and `.hlp` file formats during installation of the system. Review the following sections for documentation file descriptions, installation locations, and viewing instructions.

Versata Logic Suite Library

The Versata Logic Suite Library consists of `.pdf` (portable document format) files, an .hlp file, a chm file, and a readme.txt file. These files are automatically installed in the `\Help` subfolder of the default directory during installation.

The ***Versata Logic Suite Library*** (`Library.pdf`) is the main page Provides links to all of the `.pdf` manuals, hlp file, chm file, readme, and full-text search of all of the `.pdf` manuals.

To launch `Library.pdf` after installing Versata Logic Suite and Acrobat Reader:

On the desktop, click the Start button → Programs → Versata Logic Suite 5.5 <edition_Name> → Versata Logic Suite Library.

Note: Each `.pdf` file should be viewed, searched, and printed using the 4.05 version of Adobe® Acrobat® Reader with Search to ensure that the full-text search feature functions correctly and that graphics display properly.

This software is available for installation from the main Versata installation screen, or you may download it at `www.adobe.com`.

Note: This version of the Versata Logic Suite allows integration of transaction logic and process logic in your business objects. The integration features and documentation for those features is only available if you have purchased the Process Logic Add-On.

Versata Logic Suite Library PDF Manuals

The following `.pdf` files comprise the Versata Logic Suite Library:

- ***Getting Started Guide*** (`GettingStarted.pdf`). Provides basic installation and configuration steps for the Versata Logic Studio, Versata Logic Server, and other products needed to run the Versata Logic Suite.
- ***Tutorial*** (`Tutorial.pdf`). Steps you through features of the Versata Logic Suite. It also describes Java and HTML sample applications and shows you how to create your own Java and HTML applications (with presentation design only).
- ***Architecture and Project Guide*** (`Architecture&ProjectGuide.pdf`). Introduces the system architecture, project development process, and team development functionality of the Versata Logic Suite. This guide also contains a glossary of Versata Logic Suite, Java™, and database terms.

- ***Business Object Developer Guide*** (`BusinessObjectDeveloperGuide.pdf`). Describes how to use the Versata Logic Studio to design a data model and transaction logic for applications. Sections of this manual explain data object and query object definition, business rules development, data model and transaction logic deployment, and rules testing.
- ***Application Developer Guide*** (`ApplicationDeveloperGuide.pdf`). Describes how to use the Versata Logic Studio to create the user interface for applications (with presentation design only). Sections of the manual explain HTML and Java application development, application deployment, application testing, and application delivery.
- ***Administrator Guide*** (`AdministratorGuide.pdf`). Describes how to administer deployed objects and define security in the Versata Logic Server through the Versata Logic Server Console and server code.
- ***Reference Guide*** (`ReferenceGuide.pdf`). Contains reference information, including the Versata Logic Studio user interface help, a high-level summary of system class libraries, details about repository `.xml` and `.dtd` files, and a glossary of terms.
- ***Migration Guide*** (`MigrationGuide.pdf`). Provides guidelines for upgrading to release 5.5 of Versata Logic Suite from a previous version.
- ***PDX Guide*** (`PDXGuide.pdf`). Describes how to use the user interface development features included in PDX. These features have now been integrated into the core Versata Logic Suite product.
- ***Using PDX Frameless Archetypes*** (`Using PDX Frameless Archetypes.pdf`). Describes how to use the Frameless Archetypes feature included in PDX. This feature has now been integrated into the core Versata Logic Suite product.

Versata Logic Suite User Interface Help

The Versata Logic Suite User Interface Help is provided in a Microsoft HTML help file called `vstudio.chm`. This help file provides context-sensitive help with detailed descriptions of the frames and fields in the Versata Logic Studio.

To launch `vstudio.chm`*:*

1. Focus on a window or frame in the Versata Logic Studio and press F1 to launch a context-sensitive help topic for that element.

   OR

1. Choose Help → Versata Logic Suite Library in the Versata Logic Studio.
2. In the Versata Logic Suite Library, click the Versata User Interface Help link.
3. Click yes when prompted to open the file.

OR

1. Choose Start → Programs → Versata Logic Suite 5.5 <edition_name> → Versata Logic Suite Library.
2. In the Versata Logic Suite Library, click the Versata User Interface Help link.
3. Click yes when prompted to open the file.

Versata Class Libraries Help

The Versata Class Libraries are provided in a WinHelp file called `vstudio.hlp`. This `.hlp` file describes all of the classes and methods included in the Versata Logic Suite packages.

To launch `vstudio.hlp`*:*

1. Focus on a class name, method, or a string in the Code Editor in the Versata Logic Studio and press F1 to launch a context-sensitive help for that element.

   OR

1. Choose Help → Versata Logic Suite Library in the Versata Logic Studio.
2. In the Versata Logic Suite Library, click the Versata Class Libraries Help link.
3. Click yes when prompted to open the file.

   OR

1. Choose Start → Programs → Versata Logic Suite 5.5 <edition_name> → Versata Class Libraries Help.

Versata Logic Suite Readme

The `readme.txt` file provides latebreaking release notes about the Versata Logic Suite.

To launch `readme.txt`*:*

- During installation of the Versata Logic Studio, click the `Yes` button when prompted to view the readme.
- Choose Start → Programs → Versata Logic Suite 5.5 <edition_name> → Versata Logic Suite Readme.

Conventions for documentation and user interface help

The following conventions are used in the documentation to convey special meaning.

- Code, such as folder names, file names, and example code snippets, is shown in Courier New font, `like this.`

- Brackets, < > around part of a file name, or path, indicate that the information between the brackets should be filled in as appropriate. For example, the default directory path `\Archetypes\<application_name>\<repository_name>` indicates that text between the brackets depends on the name of the currently opened application and repository. Note that these are different from the angle brackets that appear around tags and macro code.
- Toolbar menu options in procedures are shown in this format: On the desktop, click the Start button → Programs → Versata Logic Suite 5.5 → Versata Logic Server Console. The first option is the top menu in the hierarchy. Succeeding options progress to submenus.
- Menu commands and tab names are shown with their full paths:
  - **Menus**. From the File menu, choose New → Repository.
  - **Tabs**. This option is set on the Properties: Attributes tab of the Transaction Logic Designer.
- A Caution or Warning is important advice that you should read carefully.

The Versata User Interface help uses these additional conventions.

- Click F1 while focus is on a dialog, window, menu, or toolbar in the Versata Logic Studio to launch context-sensitive help.
- Contents, Index, and Find tabs are provided on the left pane of the Help window when you launch the Versata Logic Suite help file.
  - Click the Contents tab to look in the table of contents. It lists the modules and main sequences of the help.
  - Click the Index tab to search the index.
  - Click the Find tab to search the text of the entire help system.
- Browse buttons (Back and Forward) are provided in the Help toolbar, and Previous and Next hyperlinks are provided in the Help topics. Use these options to scroll through sequences of related topics.
- To print a help topic, click the Print button at the top of the Help window. To print an entire book of help topics, select the book on the Contents tab and click the Print button.

Versata Logic Suite resources

The following resources are available to help you learn more about the Versata Logic Suite.

Sample database and sample applications

The Versata Logic Studio includes a sample database with examples of business requirements, functions, and rules. Extensive sample applications are provided to illustrate features of the HTML and Java applications generated by the Versata Logic Suite (with presentation design only). These sample applications include example code for you to use to implement complex features more easily.

The samples and sample database are located in the `Samples` directory where you install the Versata Logic Suite. The `vsamples.hlp` file, located in the `Help` directory, provides detailed descriptions of the sample database, rules examples, and sample applications (with presentation design only). In addition, the most recent description of each sample application is located in the `About_*.app.rtf` file in each sample application folder in the Versata Logic Studio Explorer.

To access the Versata Logic Suite sample applications (with presentation design only):

1. Launch the Versata Logic Studio.
2. Open `sampDB1.xml` (the sample database) as your repository.
3. Expand the `Client Applications` folder in the Versata Logic Studio Explorer.
4. Select a particular sample application and run it.
5. Review the `About_*.app.rtf` file in that sample application folder (Files tab) or choose Help → Samples to launch `vsamples.hlp` for information about that sample application.

Versata Web site

Browse the Versata Web site at `www.versata.com` for the latest information about:

- Versata Logic Suite products, upgrades, and demos
- Sales
- Employment opportunities
- Training
- Professional Services

Versata Knowledge Base

The Versata Knowledge Base is available to help with your technical questions about the Versata Logic Suite. You can search through our growing library of technical articles or participate in our online Developer Discussions forum.

To access the Versata Knowledge Base:

1. Visit the Versata Web site at http://www.versata.com.
2. Click the Training and Support tab, then select Versata Knowledge Base.

Versata Developer Discussions

Access Versata Developer Discussions on the Versata Web site. Sign up to view the postings and subscribe to the mailing list to receive the latest news about the Versata Logic Suite automatically. These technical discussions provide a forum for customers, partners, distributors, and Versata internal employees to post technical and general questions, suggestions, and solutions about development, run-time, and production features of the Versata Logic Suite.

To access the Versata Knowledge Base:

1. Visit the Versata Web site at http://www.versata.com.
2. Click the Training and Support tab and select Developer Discussion.

Versata Customer Support

You may use any of the following methods to contact Versata Customer Support.

- **Internet.** At the Versata website (`www.versata.com`), click on the Training and Support tab to find information about Versata Customer Support.
- **Phone.** 510.238.4100. Between 7:00am and 5:30pm, Pacific Time, Monday-Friday
- **E-mail.**
  - For software issues: `techsupport@versata.com`
  - For documentation issues: `docs@versata.com`

LESSON 1

Introduction to the Versata Logic Suite

Tutorial overview

This tutorial allows you to explore the Versata Logic Suite development environment and selected sample applications. In addition to stepping through this tutorial, we recommend that you review the following:

- Versata Logic Suite White Papers. These are located on the Versata website at `www.versata.com`.
- Sample applications. These are included with the sample repository.

For information about other documentation available with Versata Logic Suite, see “Versata Logic Suite documentation” on page viii.

Each Versata Logic Suite application includes three tiers: a data model deployed to a database server; transaction logic deployed to an application server; and a client user interface that runs as a standalone Java application or as an HTML application or Java applet deployed to a Web server and run through a browser. You can use the Versata Logic Studio to build, modify, and deploy all of these tiers.

- For information about how the Versata Logic Suite works, see “System overview” on page 17.
- For information about products included with the Versata Logic Suite, see “Product overview” on page 19.

This tutorial introduces you to the basic tasks required to create, refine, deploy, and run Versata Logic Suite applications. As you work through the exercises, you will use the business objects already defined in the sample repository and review existing business rules to create your own applications. By the end of this guide, you should have a high-level grasp of Versata Logic Suite vocabulary, underlying concepts, and available tools.

The sections in this tutorial include:

- “Deploying the Sample Data Model” on page 25.
- “Exploring Business Rules” on page 35.
- “Deploying Business Objects to the Versata Logic Server” on page 47.
- “Creating HTML Applications” on page 53.
- “Creating Java Applications” on page 67.
- “Tracing Rule Processing in Versata Logic Suite Applications” on page 93.

System overview

The Versata Logic Suite is a complete software solution for creating, deploying, and maintaining complex, distributed business rules-based applications through a multi-tiered, enterprise application architecture. (Application development is available only if you have purchased presentation design capabilities.) The automation provided by the Versata Logic Suite is vital to creating and maintaining continually changing business rules and applications that operate in “Web time”.

In Versata’s development environment, you can design, build, deploy, and test the transactoin logic, data source connectivity, and user interface for your Web-enabled applications. This environment exposes objects graphically, making it simple to view and modify different components to meet changing business needs.

The pace of change in the current business environment does not allow for time-consuming development practices that produce inflexible, difficult-to-maintain applications. The high level of automation provided by the Versata Logic Suite eliminates much of the implementation process that is coded by hand in other development environments. At the same time, the Versata Logic Suite allows developers to write code where it is needed to optimize, customize, or extend the system, making the integration of custom code straightforward and easy to change.

With the Versata Logic Suite, software developers and business users can work together to build and maintain transaction logic-based application systems. The two groups can tailor applications to meet particular business requirements while maintaining easy updatability to meet changing business needs. Unlike typical client/server environment systems, which divide the transaction logic and repository metadata between the client application and the database server, the Versata Logic Suite multi-tier systems place transaction logic in a middle-tier—the Versata Logic Server.

Business rules contain the transaction logic for your applications. You will specify your business requirements using the Versata Logic Studio in simple, declarative terms easily understandable by business managers. Developers can work together with them in an IT/User duet to quickly translate business requirements into executable business rules that can be deployed on standard architectures. As your business needs evolve, you will use the Versata Logic Studio to redefine requirements, modify your business rules, and redeploy applications.

In the Versata Logic Studio, you can define transaction logic as business rules on repository business objects. These objects can be built into CORBA objects that enforce your business requirements at run-time. Transaction logic executes on a middle tier so it is data-source independent, providing the ability to enforce transaction logic against multiple data sources without extra development work. This architecture facilitates integration with legacy applications.

The Versata Logic Studio development environment also exposes the files it generates for these objects, providing an editor where you can modify the files generated. This editor allows easy integration of custom code, including the addition of transaction logic event-handling

code, subclassing of the system-provided Java class files, calls to methods in external classes, inclusion of external Java files and CORBA objects, connectivity to custom data sources, and presentation logic code.

Product overview

The Versata Logic Studio enables you to design, build, and deploy a data model, transaction logic, and complex Web applications. Applications generated using the Versata Logic Studio are complete internet-ready systems—Enterprise-class, multi-tier, Java and HTML applications that are highly interactive and fully transactional.

- For Java applications, the Versata Logic Server manages the logical transactions between the user interface and the database server with help from a Web server, Internet Proxy Service, and Versata Logic Server Locator (a naming service).
- For HTML applications, the Versata Logic Server manages the logical transactions between the user interface and the database server with help from the Web server, servlet engine, and Versata Logic Server Locator.

The Versata Logic Suite multi-tiered, run-time system architecture supports integration with existing systems through Versata Connectors using eXtensible Data Access (XDA) technology. This technology provides data source independence. The robust Versata Logic Studio provides business automation for a data model, transaction logic, and the user interface.

The Versata Logic Suite includes the following products, which are described on the following pages:

- Versata Logic Studio, page 19
- Versata Logic Server, page 20
- Versata Logic Server Console, page 21
- Versata Connectors, page 23
- Versata Logic Server Locator, page 22
- Internet Proxy Service, page 22
- Service Manager, page 23

Versata Logic Studio

The Versata Logic Studio is an object-oriented integrated development environment (IDE) for designing, building, and deploying a data model, business rules, and Web-enabled applications. The Versata Logic Studio includes designers, managers, wizards, and other utilities that automate many of the tasks for developing transaction logic and Java and HTML applications, including automated code generation.

In the Versata Logic Studio, you can:

- Model database objects and their relationships to each other. You can reengineer and deploy data model information from and to supported database servers.

- Input transaction logic as declarative business rules, using the Versata Logic Suite SQL-like rules language. You can build transaction logic into CORBA objects In the Versata Logic Studio and deploy them to the Versata Logic Server.
- Design application user interfaces—the forms or pages to be included in each application, the data to be displayed on each form or page, and the navigations between pages.
- Refine the appearance of forms or pages. You can use the Versata Logic Studio-supplied archetypes to promote standardization and reuse of styles. Also, you can edit each form or page as desired. For HTML application development, the Versata Logic Studio includes a Page Designer and also integrates with external HTML editors so that you can use these editors to modify your Versata Logic Studio-generated applications.
- Customize automatically generated presentation logic and automatically generated business logic with your own Java code. For HTML applications you can customize the associated JavaScript. In the Versata Logic Studio you can open generated files and review and modify code within the development environment.
- Build, compile, and test applications.

The *Application Developer Guide* provides details about how to use the Versata Logic Studio to develop and deploy Web-enabled applications.

Versata Logic Server

The Versata Logic Server is a Java-based, object-oriented application server that functions as part of the middle tier of the Versata Logic Suite multi-tier application architecture—between the user interface and the data provider. The transaction logic and most of the presentation logic for an application reside in one or more Versata Logic Servers, where processing is transparent to the end user. The Versata Logic Suite allows you to expose this logic as CORBA objects, which can be invoked remotely so that any CORBA clients can communicate with Versata Logic Studio-generated components.

The Versata Logic Server can run on a Windows NT Server or Workstation, or on a UNIX machine. On Windows NT, you can start the Versata Logic Server from the Start menu or from within the Versata Logic Studio using the Service Manager. For information about the Service Manager, see “Service Manager” on page 23.

The main features of the Versata Logic Server include hosting the presentation logic, and hosting the repository’s business objects; applying business rules as they are defined on the server; and enforcing security and rules for the business objects. The Versata Logic Suite implements every business object in a corresponding Java object, which processes the business object’s business rules.

Other features include:

- eXtensible Data Access (XDA) through Versata Connectors
- Transaction logic processing

- Presentation logic processing
- Component-based architecture using CORBA and IIOP
- Transaction management
- Cross-platform support
- Scalability and performance enhancing services, such as load balancing, connection pooling, failover, and object caching

The Versata Logic Server provides two main benefits: data source independence and processing efficiency:

- The Versata Logic Server can run anywhere that Java runs. Its Java implementation enables it to support business rules on legacy data. You can define data objects in repository metadata that describe the legacy data model and create Versata Connectors in Java that, taken together, enable the Versata Logic Server to implement business rules on the "foreign" data.
- The Versata Logic Server manages user sessions and optimizes database server resource usage by providing connection pooling among clients. Also, you can install multiple Versata Logic Servers to provide scalability services including load balancing and failover.

For information about implementing transaction logic in business objects that run on the Versata Logic Server, see the *Business Object Developer Guide*. For information about administering the Versata Logic Server and the business objects it hosts, see the *Administrator Guide*.

Versata Logic Server Console

The Versata Logic Server Console is a Java application that provides easy, enterprise-wide management of the business objects in the Versata Logic Server. The Versata Logic Server Console provides a single point for managing multiple Versata Logic Servers—both local and remote.

The Versata Logic Server Console can be run from a Windows NT Server or Workstation, from a Windows 2000 machine, or from a UNIX machine. On Windows NT, you can start the Versata Logic Server Console from the Start menu or from the Versata Logic Studio toolbar.

The major functions of the Versata Logic Server Console are:

- System administration of the Versata Logic Server, including tasks such as mapping Versata Logic Server objects to data sources, configuring the Versata Logic Server, managing Versata Logic Server replication, monitoring performance, and session management.
- Security administration of the Versata Logic Server business objects, including tasks such as adding and removing users, assigning roles to users, and assigning security permissions to roles.

The *Administrator Guide* provides details about using the Versata Logic Server Console to administer business objects in the Versata Logic Server.

Versata Logic Server Locator

The Versata Logic Server Locator is a naming service product installed automatically when you install the Versata Logic Server. The Versata Logic Server uses the Versata Logic Server Locator to receive bind calls from client machines, make connections to the database server, and return to the client machine an object reference that allows it to connect to the server.

The Versata Logic Server Locator provides transparent hosting of the Versata Logic Server and its business objects. The Versata Logic Server Locator acts as an interface repository, supporting dynamic and static discovery of the business object classes and methods on the Versata Logic Server, so that the client application thinks that they are local and in process. Actually, the business objects can be anywhere on the WAN and the Locator can find them. In addition, the Locator acts as an implementation repository, caching the business object instance(s) and state information for later reuse.

The Versata Logic Server Locator finds Versata Logic Server servers and CORBA objects by name, translating each name request into an address. If multiple Versata Logic Servers are installed, the Versata Logic Server Locator distributes processing requests in round-robin fashion. At least one Versata Logic Server Locator is required for each Local Area Network (LAN). A LAN may have multiple Locators.

On Windows NT, you can start the Versata Logic Server Locator from the Start menu or from within the Versata Logic Studio, using the Service Manager. For information about the Service Manager, see page 23.

The *Administrator Guide* provides more details about the Versata Logic Server Locator.

Internet Proxy Service

The Internet Proxy Service (IPS) is an operating system service used with Java applets. This proxy allows communication between the downloaded applet on users' local machines (usually outside of a firewall), the Web server where the application is published, and the Versata Logic Server Locator that finds the Versata Logic Server(s). The IPS is an Object Request Broker (ORB) built from a Visigenic daemon.

The IPS provides a solution in environments where a firewall allows only HTTP packets to go through. The IPS converts HTTP packets from the applets into IIOP packets that can be forwarded to the Versata Logic Server, and also converts IIOP packets from the Versata Logic Server to HTTP packets.

You can start the IPS from the Start menu or from within the Versata Logic Studio, using the Service Manager. For information about the Service Manager, see page 23.

The IPS should be installed on the same machine as the Web server. The IPS has a configuration file where you can set different properties to provide functionality for different application architectures. For details, see the *Administrator Guide*.

Service Manager

The Service Manager starts and stops the Versata Logic Server, the Internet Proxy Service, and the Versata Logic Server Locator.

You can launch the Service Manager from the Versata Logic Studio toolbar, or from Windows NT. To launch the Service Manager in Windows NT, choose Start → Programs → Versata → Service Manager. To start or stop a Versata Logic Suite service from the Service Manager, select the service in the drop-down list and click Start or Stop.

Versata Connectors

The Versata Logic Suite XDA (eXtensible Data Access) framework provides connectivity from business logic components and client applications (with presentation design only) on the Versata Logic Server to the physical data source(s). This framework provides the ability to connect with a variety of data sources, including relational, non-relational, legacy, and packaged application data.

This framework consists of well-defined, generic Java interfaces for querying, fetching, updating, and saving data on any type of data source. These interfaces are included in objects called Versata Connectors, so that this code is separated out from code for transaction logic and application presentation logic. Connectors' code integrates with JDBC driver interfaces.

Connectors to common relational databases are included with the Versata Logic Suite and installed with the default product install. Some Connectors to other data sources are available for separate purchase. If you require connectivity to another type of data source, you can write your own custom Connector code.

For more details about Versata Connectors and the XDA framework, see the *Business Object Developer Guide*.

LESSON 2

Deploying the Sample Data Model

Overview

A Versata Logic Studio application must be based upon a data model that is a logical representation of the data stored in the database. In addition, a Versata Logic Studio application must include transaction logic that describes the data processing rules when users make changes to the data.

Later sections of this tutorial explain how to declare business rules that implement data processing requirements and how to develop application user interfaces. Before you begin those tasks, we recommend that you become familiar with the setup of Versata Logic Studio applications by looking at the sample database, sample business rules, and sample applications provided with Versata Logic Suite.

This section explains how to create a Microsoft® SQL Server™ database, and use the Versata Logic Studio to copy the sample data model and test data from the Versata Logic Suite repository to the database server. After you complete these steps you can run sample applications.

Note:

- The SQL Server instructions require the software for Microsoft® SQL Server and SQL Enterprise Manager. If you do not have this software, you might want to install an evaluation copy from the Microsoft web site.
- You also can use other supported database servers. To use a different database server, consult its documentation about setting up a database, and then skip to "Setting up an ODBC System Data Source Name (DSN)" on page 30.

This section explains the following tasks:

- "Setting up a Microsoft SQL Server database" on page 27
- "Setting up an ODBC System Data Source Name (DSN)" on page 30.
- "Deploying sample data model and test data to a database server" on page 32.

Setting up a Microsoft SQL Server database

These procedures describe how to create a simple Microsoft SQL Server database for application development and testing. These procedures do not optimize for performance or other resource considerations. Refer to the product documentation for information about these considerations. In fact, if you have any problem implementing these instructions, please refer to the Microsoft SQL Server production documentation.

These procedures assume the following:

- You have Microsoft SQL Server 7.0 with SQL Enterprise Manager installed on your local machine
- You have full permissions on the Microsoft SQL Server where the database will be set up.
- Microsoft SQL Server is running.

For assistance in meeting these prerequisites, contact your system administrator. For instructions on starting Microsoft SQL Server, see the user documentation for the RDBMS.

To set up a Microsoft SQL Server database:

1. Create a database. See "Creating a Microsoft SQL Server database" on page 27.
2. Create a database login. See "Creating a Microsoft SQL Server database login" on page 28.
3. Grant permissions and options for the database. See "Granting permissions and options for the database" on page 29.

Note: If you want to use this procedure and you do not have the software for Microsoft SQL Server and SQL Enterprise Manager, you may want to install an evaluation copy from the Microsoft Web site. Remember to install the native Microsoft SQL Server JDBC™ driver as well.

Creating a Microsoft SQL Server database

In this task, you create a database to which you can, in a later step, deploy your Versata Logic Suite data model. This procedure assumes that you have already installed Microsoft SQL Server software.

To create the database:

1. Choose Start → Programs → Microsoft SQL Server 7.0 → Enterprise Manager.
2. In the Enterprise Manager Console Root window, expand the Console Root folder until you reach the Databases folder.

   A number of databases are listed here:

   The *master* database is the default database; it contains system tables that are used with every database you create.

The *model* database can be a template for new databases you create.

The *tempdb* database acts as a temporary database the server uses.

The *pubs*, *msdb*, and *Northwind* databases are sample databases.

3. Right-click on the Databases folder and from the menu, choose New Database.
4. In the Database Properties dialog box, enter a name for your database in the Name field, for example, `OrdersDB`. Notice that the file name for the database has "_Data" appended to it.

   Do not be concerned that the initial size for your database is only 1MB—you will set the database to automatically grow as new data is added to it.
5. Under File Properties, click to check the Automatically grow file check box, and select the By percent button, leaving it set at 10.

   This Create Database dialog also creates a Transaction Log file, the name of which you can determine by clicking on the Transaction Log tab. This file records all transactions you make on the database, such as dropping and recreating tables.
6. Click OK to create the database.

After you create the database, an icon appears for it, along with the other database icons. If you expand the new database folder and double-click the tables icon, you'll notice that there are a number of tables beginning with "sys" added. These are system tables that the database will use. Do not alter or drop these tables.

Creating a Microsoft SQL Server database login

In this task, a user on the new Microsoft SQL Server database is created. You will use this user later to deploy a Versata Logic Suite data model to the new database.

Note: This procedure assumes that you have already created a database in the previous step.

To create a login for a database:

1. Choose Start → Programs → Microsoft SQL Server 7.0 → Enterprise Manager.
2. Expand the Server list, and select the server where you created a new database.
3. Click on the option, Administer SQL Server, and then click on the option, create a login.
4. This runs the Create Login wizard.
5. In the wizard, select the type of authorization, the degree of access, and the database that you just created.
6. Click Finish.

Granting permissions and options for the database

In this task you set options for the new database and grant permissions to the new user on the database.

This procedure assumes that you have created a database device, created a database, and created a new user (login) for the database.

To grant the appropriate permissions and options to the database:

1. Click Start → Programs → Microsoft SQL Server 7.0 → Enterprise Manager.
2. Expand the SQL Server folder in the Explorer, the Databases folder, and select the database you just created.
3. With the database displayed, right-click to display the context menu, and select the Properties option.
4. Select the Permissions tab.
5. For the new user, click each permissions box until you have granted permissions for all options. Note that CREATE DATABASE permissions can only be set from the master database.
6. Select the Options tab.
7. Enable the ANSI NULL by Default and Truncate Log on Checkpoint options.
8. Click OK to close the dialog.

Setting up an ODBC System Data Source Name (DSN)

The DSN stores information about how to connect to your database from another source. It allows you to connect the database to Versata Logic Studio, and hence deploy your repository data model. Whenever you deploy your data model to the database, Versata Logic Studio will ask you for this DSN. A user data source is visible only to you and can be used only on the current machine. A system data source is visible to all users on the machine, including Windows NT services.

You will need to set up a DSN for the Versata Logic Studio, Versata Logic Studio applications, or the Versata Logic Server to use for connection to the database server. You may want to set up a DSN with a meaningful name, for example `SampDSN` to connect to the sample database.

To create the system DSN:

1. Choose Start → Settings → Control Panel.
2. Double-click the ODBC Data Sources icon.
3. In the ODBC Data Source Administrator, click the System DSN tab.
4. Click the Add button.
5. In the Create New Data Source dialog, choose the SQL Server driver and click Finish.
6. In the Create a New Data Source to SQL Server dialog, do the following:
   - Enter a name for the data source, for example, `OrdersDB_dsn`. Make a note of this name because Versata will prompt you for it whenever you deploy to the database.
   - Optionally enter a description for the DSN.
   - In the Server drop-down list, choose (local), or

     If you want the DSN to connect to a SQL Server on another machine, choose the machine from the Server drop-down list.

   Note: If the Create New Data Source dialog does not list a supported driver for your database server, you may need to install a new driver. For information about supported drivers, see the *Getting Started Guide*.
7. Click Next.
8. In the next dialog, for the prompt, “How should Windows verify the authenticity of the login?”, select the radio button labelled “With Windows NT authentication using the network ID”.
9. Click Next.
10. In the next dialog, set the default database to the one you created earlier for the Versata data model.
11. Accept the defaults for the remaining prompts, and click Next.

12. In the next dialog, accept the defaults and click Finish.
13. In the ODBC Microsoft SQL Server Setup dialog, click Test Data Source. If a message displays declaring a successful connection, you can exit the window. If not, review and repeat this procedure.
14. Click OK to complete the DSN creation.
15. Click OK to exit the ODBC Data Source Administrator dialog.

Deploying sample data model and test data to a database server

You should deploy the sample data model to a database server before you deploy sample business objects to the Versata Logic Server, so that the correct database connection properties can be created for the business objects. You also should transfer sample data during deployment to the database. Deployment is explained in the following instructions.

Note: The following procedure assumes that the database server is running.

To deploy the sample data model and data to a database server:

1. Choose Start → Programs → Versata Logic Suite 5.5 CORBA Edition → Versata Logic Studio.
2. Open the sample repository by selecting it as the program loads, or by choosing File → Open Repository from within the Versata Logic Studio. The sample repository is located in the `<install_directory>\Samples\SampDB1\Source` folder and is called `SampDB1.xml`. In the Open Versata Repository dialog, click the Existing tab and browse to this folder and click OK.
3. Choose Managers → Deployment Manager, then select Database Server as the deployment target and click the Next button.
4. In the Server Manager Introduction dialog, choose the type of server that you are using and click the Next button.
5. In the Connect For Auto Selection dialog, ensure that this option is disabled and click the Next button.
6. In the Select Data Objects dialog, select all of the database objects by clicking the double-chevron button (>>). Click the Next button.
7. In the Deploy to the server or the scripts dialog, choose Deploy to the Server and click the Next button.
8. In the What to deploy dialog, enable the Transfer Test Data and GRANT ALL Permissions to PUBLIC and click the Next button.
9. In the Data Model Deploy Options dialog, choose Drop and recreate the data model and click the Next button.
10. In the Configuration Options dialog, do not enable the Generate Quoted Identifiers option. Click the Next button.
11. In the Ready to Deploy dialog, review your selected options, and select Finish if everything is correct. If not, use the Back button to return to previous dialogs and make corrections, then click Finish.
12. In the Select Data Source dialog, click on the Machine Data Source tab, and select the data source that you wrote down when you created it in an earlier step. Click the OK button.

13. In the Server Deployment Preview dialog, review the messages presented. This dialog should contain informational messages only. If there are errors or warnings, you may be deploying to a schema that contains tables with the same names as those in the repository. In this case choose another schema, and try deploying again. If the deployment is in order, choose Deploy.

    Deployment may take some time, depending on your database server and network.

14. When the deployment is complete, the Server Deployment dialog appears, informing you of the success (or failure) of the deployment. Click the OK button.

Note: A lack of space for the transaction log in tempdb may cause an error to occur during the transfer of test data: “an inability to allocate a new page for tempdb.” If you receive this error, do not cancel the deployment. Perform the following steps to relieve the problem in tempdb:

a. Open the Microsoft SQL Enterprise Manager.

b. Select the current database server.

c. Expand the server in the Explorer

d. Right-click the Databases folder and choose Edit.

e. In the Edit Database dialog, click Truncate, and OK.

f. Return to the Versata Logic Studio and click Yes in the error dialog to continue with the deployment.

LESSON 3

Exploring Business Rules

Overview

In the Versata Logic Suite you define declarative business rules to drive transaction logic for computation and validation during transaction processing. You can choose to persist derived data, or not. The declarative rules generate substantial executable procedural logic you would otherwise have to manually code. Business rules consist of simple statements defining the desired result rather than lengthy code blocks describing in detail how to achieve the desired result.

Beyond the benefit of code replacement, a declarative approach to transaction logic offers other important advantages:

- **Automatic reuse.** Unlike traditional design where you code specific logic for each transaction, you state rules about data. The system then automatically reuses these rules across all relevant transactions. This reuse eliminates errors that can occur in manual design and coding. For example, when you state that the customer balance is the sum of the unpaid orders, you do not need to design event or transaction handling logic for related transactions such as inserting orders and deleting orders, in which the customer balance is calculated.
- **System-defined ordering.** The system computes the correct and most efficient order of rules processing. Furthermore, this processing order is recalculated each time you modify the rules during iterative development or maintenance.
- **Automatic optimization.** The rules compilation process automatically partitions your code to the Versata Logic Server and employs sophisticated algorithms to minimize network and I/O overhead. To maintain consistently good performance, these optimizations are recomputed each time you modify business rules.

This section includes the following :

Automating business functions with rules

As with manually coded systems, the development process in the Versata Logic Suite begins with identifying your business functions and the requirements for each function. You then define one or more declarative rules to implement each requirement.

Types of business rules

Rules are triggered when data changes. Accordingly, you create business rules that apply to a data object, or rules that apply to individual attributes in a data object. The following briefly describes the different types of rules you can define in the Versata Logic Studio's Transaction Logic Designer. These are explained in greater detail in the *Business Objects Developer Guide*.

Types of **object-level rules** are:

- **Relationship rules.** These rules specify relationships between data objects, including how you want to manage their referential integrity. For example, an order must have a customer. Deleting a parent order could delete all of its child line items. You can disallow deleting parent customers if they have existing child orders.
- **Constraint rules.** These rules contain (possibly multiple) conditions that are evaluated, and specify the action taken if the conditions evaluate to True. Any event that attempts to change an attribute referenced in the condition, causes the constraint rule to execute. For example, a customer's account balance cannot exceed its credit limit. You can specify that any transactions that would violate this rule would be rejected.
- **Action rules.** These rules contain (possibly multiple) conditions that are evaluated and specify the ***user-supplied Java method*** call that executes if the conditions evaluate to True. For example, notify the associated employee's sales manager by email whenever an order greater than $50,000 is placed.

Types of **attribute-level rules** are:

- **Derivation rules.** These attribute rules determine how an attribute's value is derived. For example, an attribute could be the result of a count, sum, formula, default value, or the value of a parent data object's attribute (referred to as parent replicates).
- **Validation rules.** These attribute rules determine how an attribute is validated, and can be formulas or coded-value lists.
- **Presentation rules.** These rules control how the data is presented, and consist of labels and formats, such as currency, and so on.

Reviewing the rules to validate customer credit

Under this requirement, if a customer attempts to order parts whose value causes the credit limit to be exceeded, the order should be rejected. Working in a top-down manner, you can examine how the constraint and related rules enforcing this requirement are declared in the Versata Logic Studio's Transaction Logic Designer.

To review the rules that implement this requirement:

1. In the Versata Logic Studio, with the SampDB1 repository opened, select the Objects tab at the base of the Explorer. Expand Business Logic by clicking the + next to it, and repeat for the CustomerOrders group folder. Double-click CUSTOMERS. The Transaction Logic Designer appears, loaded with the CUSTOMERS data object.
2. View the definition for the constraint BalanceExceeded using the directions that follow:
   - Select the Constraints tab. Constraints have five properties: name, type (Accept When/ Reject When), condition, an error message to return, and an optional error attribute (that receives focus when the constraint is violated).
   - Under the name column, select the BalanceExceeded constraint. Its components display below the grid containing the constraints. The BalanceExceeded constraint references the fields ActBalance and CreditLimit, which are defined in steps 4 and 5. Whenever ActBalance or CreditLimit are updated, this constraint validates the result. If the constraint is violated, the entire business function is rolled back to its initial state. ActBalance and CreditLimit are derived values, and they are used by the rule that validates customer balance against the credit limit.
3. While still in the Transaction Logic Designer for CUSTOMERS, select the Attributes tab.
   - Upon selecting Attributes, a second sheet of tabs appears across the middle of the form. Notice the Derivation tab. As you select attributes in the top of the form, their derivation information (if any) appears in the Derivation tab.
4. In the Attributes tab, select ActBalance to view how the value is derived. If necessary, scroll through the grid to locate it.
   - After selecting ActBalance, note in the Derivation tab that it is defined as the sum of values in the OrderTotal attribute of the data object ORDERS.
   - Sums can be qualified such that only the rows that satisfy the specified conditions are included in the sum. In this example, the condition `OrderPaid = false` must be met before the OrderTotal attribute is summed into ActBalance.
   - Derivations automate multi-table transaction processing. Whenever the value of OrderTotal changes, or an unpaid order is added or deleted for a customer, this rule fires in all relevant transactions and recalculates the customer's account balance. For example, this rule fires when the order is paid, reassigned, or otherwise changed.

5. Review the derived value CreditLimit in the CUSTOMERS data object. On the Attributes tab, select CreditLimit. Upon selection, note in the Derivation tab that CreditLimit is defined in Derivation Type as a formula, which is stated in the Formula Expression box.

   The credit limit attribute illustrates several important features available for derivations based on formulas:

   - **If logic.** Expressions can include complex conditional logic.
   - **Old/New value.** Expressions can reference old (still in cache, but not committed to the database yet) and new values for the row.
   - **Chained derivations.** Expressions can reference other derived attributes (MaxCreditLimit in this case).
   - **You can declare rules in any order.** The Versata Logic Suite determines the correct and efficient order of execution. You can change your mind about whether to derive fields, and all dependent logic is automatically redesigned by the system.

6. At any point during this walkthrough you can click the Notes tab to view an explanation of any selected rule.
7. Examine the derived attribute MaxCreditLimit in the Customers data object:
   - In the Attributes tab, select MaxCreditLimit.
   - After selecting it, note in the Derivation tab that MaxCreditLimit is defined as a parent replicate of the VALID_CREDIT.creditMaxBalance attribute. This means that when a value is entered in the VALID_CREDIT.creditMaxBalance attribute, it will be replicated (copied) into the MaxCreditLimit attribute of the CUSTOMERS data object.
   - The Maintained check box indicates that all updates to creditMaxBalance in the data object VALID_CREDIT are reflected in changed MaxCreditLimit values for all related customer rows.
   - A checked Persistent check box indicates that the derived attribute is stored in the database.

Reviewing the rules to compute the total order amount

The total order amount (ORDERS.OrderTotal) for an ORDERS data object is calculated using Tax, Freight, a boolean indicator for pre-payment, the amount of earned credit for the customer, and the summed amounts of each line item on the order.

Note that the OrderTotal attribute is derived independently of rules for other attributes that use its value, yet attributes that use OrderTotal in their derivation are automatically updated when OrderTotal changes. For example, you do not need to write code to adjust the customers' account balance (CUSTOMERS.ActBalance) when the order totals (ORDERS.OrderTotal) belonging to that customer change, even though the ORDERS.OrderTotal is involved in the calculation of CUSTOMERS.ActBalance.

This characteristic of the rules-based approach is in sharp contrast to procedural approaches, where every circumstance under which a value is set requires explicit code to propagate impacts to dependent data.

To review the rules that implement this requirement:

1. In theVersata Logic Studio Explorer, double-click the data object, ORDERS.
2. In the Transaction Logic Designer, select the Attributes tab.
3. Examine the attribute, OrderTotal:
   - In the Attributes tab, select OrderTotal. If necessary, scroll vertically through the grid to locate it. After selecting it, note in the Derivation tab that ORDERS.OrderTotal is derived from a formula. Depending on which condition is met (`If ( IsPaidByAutoBucks = true ) Then )`, either just Freight and Tax are added or Freight and Tax and the amounts of each line item (ORDERS.AmountItems) are added.
   - Note that this attribute is required in order to define the sum rule for CUSTOMERS.ActBalance (illustrating the principle that sums can only sum attributes, not expressions).
   - The derivation for Freight will be examined and modified in a later step to illustrate the process of business rule development.
4. Review the AmountItems attribute in the ORDERS data object. This attribute contains the total amount of its associated line items.
   - With the ORDERS data object open in the Transaction Logic Designer, select AmountItems on the Attributes tab. Note that it is derived by summing the Amount attributes in all of its related (child) line items in the ORDERITEM data object.

5. Examine the Amount attribute in the ORDERITEM data object:
   - In the Versata Logic Studio Explorer, double-click the ORDERITEM data object.
   - On the Attributes tab, select the Amount attribute. Note that this attribute's value is derived from a formula that multiplies the ORDERITEM.PartPrice by the quantity ordered (ORDERITEM.QtyOrdered).
   - After the referential integrity check to validate an order number, the calculation performed on Amount serves as input to calculate the total order amount. The update to the Amount field causes the attribute derivation for OrderTotal to fire.
   - QtyOrdered is entered by the end user in the course of placing an order.
6. Examine PartPrice:
   - On the Attributes tab, select PartPrice.
   - PartPrice is defined as a parent replicate of the Price attribute of the PART data object. Note that in contrast to the parent replicate for CreditMaxBalance, this replicate is not maintained (the Maintained box is not checked), so that subsequent changes to PART.PartPrice after the part was placed on the order do not affect the order amount.

Reviewing the rules to reorder parts if necessary

Under this requirement, if the total number of orders causes the existing stock to fall below the reorder point, the NeedsReorder flag attribute in the PART data object is set to initiate part replenishment.

To review the rules that implement this requirement:

1. Examine the derivation for NeedsReorder:
   - In the Versata Logic Studio Explorer, double-click the PART data object. The Transaction Logic Designer displays, with the PART data object loaded.
   - On the Attributes tab, scroll down and select the NeedsReorder attribute. After selecting it, note on the Derivation tab that this attribute is derived from a formula that sets a Boolean flag if the reorder point is higher than the quantity on hand after filling current orders: `(QtyOnHand-QtyUnshipped)<QtyReOrder`.
2. On the Attributes tab, select QtyOnHand to view its derivation:
   - Using formulas, you can distinguish between the new, uncommitted value of an attribute and its previous value (before the commit). For example, `:OLD.QtyShipped` is the quantity shipped out on an order before the current update to the order is committed, whereas QtyShipped is the quantity after the proposed commit.
   - Note that the new value for QtyOnHand includes comparisons of current and previous values for QtyOnHand, QtyShipped, and QtyReceived.
   - **Notes on reading this formula:** `Inserting` refers to the act of adding a new record; `$value` refers to the attribute itself, and `:OLD.<identifier>` refers to the old, cached value of the attribute before committing the new value.
3. On the Attributes tab, select QtyUnshipped to view its derivation:

   Note that the value of QtyUnshipped is the sum of the ordered but unshipped quantities of each part on order. The formula counts only the parts on orders that are unshipped by setting a Qualification Expression of `ORDERITEM.ShippedFlag = false`. This ensures that only orders that have not yet shipped are included in the calculation.
4. On the Attributes tab, select QtyReorder to view its derivation. This attribute contains the value entered by an end user in the QtyReorder field. If the end user does not enter a value, a default of `0` is entered. This default is provided because a non-Null value is required for other derivations that use this attribute.

5. Examine ShippedFlag:
   - In the Versata Logic Studio Explorer, double-click the data object ORDERITEM. The Transaction Logic Designer appears, with ORDERITEM displayed. On the Attributes tab, select ShippedFlag.
   - Note that ORDERITEM.ShippedFlag obtains the value of ShippedFlag from the parent data object ORDERS.
   - Note that the value is maintained, so that all changes to ShippedFlag on the parent data object ORDERS are propagated to each order item, automating this business function. In practical terms for this example, when the order ships, the order's line items reflect that status.

Reviewing the rules that automate freight charge calculations

When projects change during development or post-production, incremental changes can be made to the behavior of the repository, using the Transaction Logic Designer. These changes apply immediately to all subsequent transactions.

For example, the Order Entry application calculates Freight costs automatically, rather than requiring they be entered by the end user. The rule calculates the freight value based on the amount ordered. If the amount is above a certain threshold, the freight rate is set to a discounted rate. Otherwise, freight is set to 20% of the total cost of the amount items.

You can examine the existing rule to calculate the freight charge, then modify the rule to change the default freight rate to 15%.

To review the rules that implement this requirement:

Review the derivation for Freight:

1. In the Versata Logic Studio Explorer, double-click ORDERS. The Transaction Logic Designer displays with the data object ORDERS and its attributes loaded.
2. On the Attributes tab, select the Freight attribute.

   The formula contains a conditional (`if... then...else...endif`) that evaluates whether the amount being ordered will qualify for a discount rate. The `GetFreightBulkDiscountValue method` is used to evaluate this. This method is stored in a Java file called `CorpReuseExtRulesDataObject.java`.
3. Examine the `CorpReuseExtRulesDataObject.java.` code by selecting the Files tab at the bottom of the Versata Logic Studio Explorer. Click the + next to Versata Logic Server, then click the + next to JavaExtensions. Double-click `CorpReuseExtRulesDataObject.java` to examine the code. (You will need to scroll down in the file). Close the file when you are finished.
4. Select the Objects tab at the bottom of the Versata Logic Studio Explorer. In the Versata Logic Studio Explorer, double-click the data object, ORDERS. The Transaction Logic Designer displays, with the data object ORDERS loaded in it.

To modify the Freight rule:

1. On the Attributes tab, select the Freight attribute.
2. On the Derivation tab, to the right of the Formula Expression memo field, click the browser button. The Rule Builder appears.

   Rule expressions can reference the methods of all classes in the repository, keywords that access old values, the `current` verb, and `If-Then` conditions. These are critical in handling the requirements of sophisticated business logic. The Rule Builder provides point and click access to all of these formula components.

3. In the Rule Builder, highlight the value `0.20`. in the first instance of the string `$value = AmountItems * 0.20.` Type in the value `0.15`.

   Verify that the Rule Expression field contains the following expression.

```
If ( AmountItems <= GetFreightBulkDiscountValue() ) Then
  $value = AmountItems * 0.15
Else
  $value = AmountItems * GetFreightBulkDiscountRate()  /* eg, 10%
*/
End If
```

4. Click the OK button.
5. Select File → Save Transaction Logic to save this modification to the freight calculation rule.

LESSON 4

Deploying Business Objects to the Versata Logic Server

Overview

This section describes how to copy the sample business objects from the Versata Logic Suite repository (grouped by project, and stored as `.xml` files) to the Versata Logic Server. Business objects include the code to execute business rules. Business rules automate the computation of data, the validation of data, client application behavior, and calls to external methods when conditions are met. (See the previous chapter for information about business rules.)

The Versata Logic Server is a Java-based object-oriented application server that functions as part of the middle tier of the Versata Logic Suite multi-tier application architecture, between the thin client and the data provider. The transaction logic for an application resides on one or more Versata Logic Servers, where processing is transparent to the end user. The Versata Logic Suite allows you to expose this logic as CORBA components that can be invoked remotely, so any CORBA clients can communicate with Versata Logic Suite components.

For Java applications, the Versata Logic Server manages the logical transactions between the client and the database server with help from a Web server, Internet proxy server, and Versata Logic Server Locator. For HTML applications, the Versata Logic Server manages the logical transactions between the client and the database server with help from the Web server, servlet engine, and Versata Logic Server Locator.

This section includes the following tasks:

- "Deploying sample business objects to the Versata Logic Server" on page 50.
- "Setting up security in the Versata Logic Server Console" on page 52.

Starting the Versata Logic Server

The Versata Logic Server is installed as a manual service. Start the Versata Logic Server service in Windows NT from the Versata Service Manager.

Note: The Versata Logic Server requires that one or more Versata Logic Server Locators or other naming service is running. If the locator service stops, the Versata Logic Server will not work. Instructions for starting both of these programs follow.

To start the Versata Logic Server service in Windows NT:

1. Choose Start → Programs → Versata Logic Suite 5.5 - CORBA Edition → Service Manager.
2. Select VLS Locator from the drop-down list, and click Start. The signal button changes from red to green to indicate that the Versata Logic Server Locator is running.
3. In the same window, select Logic Server from the drop-down list, and click Start. The signal button changes from red to green to indicate that the Versata Logic Server Locator is running.
4. Close the Versata Service Manager dialog when finished.

5. At any time, you can ensure that the Versata Logic Server and the Versata Locator are still running by checking for the processes `VLS.exe` and `OSAgent.exe` (Locator) in the Task Manager, or look them up in the Services Manager.

For more information about administering the Versata Logic Server, see the *Administrator Guide*.

Deploying sample business objects to the Versata Logic Server

Deploy the sample business objects to the Versata Logic Server after deploying the data model to the database server. (For information about deploying the data model, refer to “Deploying the Sample Data Model” on page 25.) This ensures that the Versata Logic Server copies the correct database connection properties. You will use the Versata Logic Server Deployment wizard to deploy business objects’ transaction logic to the Versata Logic Server. The Versata Logic Server Deployment wizard automatically rebuilds and compiles all Java component files for business objects before copying them to the Versata Logic Server. Informational messages, warnings, and errors are listed in the bottom pane of the Versata Logic Studio window.

For more information about deploying business objects to the Versata Logic Server, refer to the chapter, “Building and Deploying Business Objects”, in the *Business Objects Developer Guide*.

To deploy the sample business objects to the Versata Logic Server:

1. Open the SampDB1 repository in the Versata Logic Studio.
2. Start the Versata Logic Server Locator and the Versata Logic Server, if they are not already started. For instructions, see “Starting the Versata Logic Server” on page 48.
3. From the Managers menu, select Deployment Manager, or press F8.
4. In the Choose Deployment Target dialog, select Deploy Transaction Logic under Versata Logic Server and click the Next button.
5. In the Deployment Options dialog, indicate the cases in which business object files should be rebuilt and recompiled before deployment and click OK.
   - Force indicates that files for all objects should always be rebuilt and/or recompiled before deployment.
   - Incremental indicates that only files for new or changed objects should be built and/or compiled before deployment.
   - None indicates that no files should be built and or compiled before deployment.

   Files are rebuilt and compiled as necessary.
6. In the Choose Versata Logic Server for Deployment dialog, enter the path of the folder where you installed the Versata Logic Server and click Next. By default, this is your Versata Logic Suite installation directory path.
7. In the Finished dialog, review the names of the host and the Versata Logic Server, and enable the check box for the wizard to stop and restart the Versata Logic Server. It must be stopped and restarted in order for deployment changes to be applied.
8. Click Finish to execute the deployment.

The wizard copies the component class files to the location used by the Versata Logic Server. One or more confirmation dialogs may display, asking you whether to overwrite previously deployed files. Generally you can click Yes to All.

When deployment is complete, a dialog appears, where you can click Yes to view the log file. Review this file for a list of deployed files.

Setting up security in the Versata Logic Server Console

After you have deployed the business objects to the Versata Logic Server, you must set up the Versata Logic Server Console before you can run applications.

The following procedure describes how to grant all privileges to a role in the Versata Logic Server Console so that users can have full access to applications and business objects. It is used to demonstrate the steps involved in preparing business objects to be accessed through Versata Logic Server Console so that users can get up and running quickly. In production environments, however, for security reasons you would not give the Public role full privileges, since Public is a predefined role conferred automatically on any valid user of the Versata Logic Server Console.

Note: The Versata Logic Server and the Versata Locator must be running before you attempt to start the Versata Logic Server Console. For instructions, see "Starting the Versata Logic Server" on page 48.

To set up business objects' security in the Versata Logic Server Console:

1. Start the Versata Logic Server Console:
   - In the Versata Logic Studio, choose Tools → Versata Logic Server Console
   - Or click the Versata Logic Server Console toolbar button from the Versata Logic Studio.
   - Or choose Start → Programs → Versata Logic Suite 5.5 - CORBA Edition → Versata Logic Server Console.
2. In the Logon to the Versata Logic Server dialog, enter the required information and click OK. You should be able to use the following defaults:
   - Admin login: `sa`
   - Admin password: (leave blank)
   - Versata Logic Server: (leave blank)
3. In the Versata Logic Server Console, expand the Versata Logic Server object, the machine, the Administration object, and the Roles folder.
4. In the Roles folder, select Public.
5. Click the Privileges:Business Objects tab. Ensure that the Both option button is selected. Click Grant All, clicking Grant Read, Grant Create, Grant Update, and Grant Delete in succession. This grants all users full access to read and modify business object data.
6. Click the Save All Changes toolbar button.
7. Close the Versata Logic Server Console.

LESSON 5

Creating HTML Applications

Overview

In the Versata Logic Studio, you can model HTML applications using the Application Designer. Wizards enable you to create application objects quickly and easily. An application diagram displays the flow of the application, and property sheets provide easy access for setting the properties for the objects. The Versata Logic Studio Explorer provides access to the application objects.

The Versata Logic Studio wizards enable basic user interface modeling with the simplicity of drag-and-drop yet with considerable control over the outcomes. Applications are defined using the wizards and refined using the Versata Page Designer or an external HTML editor.

In this section you create a simple HTML application, then add objects to the application. You can run the application, and toggle between the running application and the design environment to compare design-time and run-time application features.

This section includes the following simple tasks:

- "Creating a simple HTML application" on page 56.
- "Adding objects to an HTML application" on page 59.
- "Comparing design-time and run-time application features" on page 63.

For details about developing HTML applications in the Versata Logic Studio, including refining them in the Versata Page Designer, see the *Application Developer Guide*.

Learning more about HTML application development

The Versata Logic Studio includes a sample database with example business requirements, functions, and rules, and extensive sample applications to illustrate features of HTML and Java applications generated by the Versata Logic Studio. These sample applications also provide example code for you to use to implement complex features more easily.

The sample applicationss and sample database are part of the SampDB1.xml repository located in the `<versata_install>\Samples\SampDB1\Source` directory. You can open this repository in the Versata Logic Studio and review the description of each sample application in its `About_<Application_Name>.app.rtf` file in that sample application's folder in the Versata Logic Studio Explorer.

Highlights of some HTML sample applications

The following sample applications illustrate some commonly used HTML application development features:

- **Basic_HTML**. This application illustrates how you can build sophisticated HTML applications without programming. You can examine standard behaviors as well as easy definition of additional HTML behaviors.
- **Frames_Multiple.** This sample HTML application illustrates how you can divide your browser window into multiple regions called frames, each containing a separate, related data page.
- **ShoppingCart**. This sample application illustrates various elements of the Versata Logic Suite declarative support for page transitions, including the Initial Mode and Query properties.

To access Versata Logic Suite sample applications:

1. Launch the Versata Logic Studio.
2. Open `SampDB1.xml` (the sample database) as your repository.
3. Expand the Client Applications folder in the Versata Logic Studio Explorer.
4. Expand the HTML Applications folder in the Versata Logic Studio Explorer.
5. Select a particular sample application and double-click it to open it in the Application Designer.

To access the `About_<Application_Name>.app.rtf` *files for a Versata Logic Suite sample application:*

1. Launch the Versata Logic Studio.
2. Open `SampDB1.xml` as your repository.
3. Select the Files tab at the bottom of the left frame of the Versata Logic Studio Explorer.
4. Expand the Client Applications folder in the Explorer.
5. Expand the HTML Applications folder in the Explorer.
6. Double-click a particular sample application to open it.
7. Double-click on the `About_<Application_Name>.app.rtf` file for that application.

Creating a simple HTML application

You will be creating a simple, one page HTML application that presents customer information in scalar fields with query capability.

Designing the application

To develop a simple HTML application:

1. Choose File → New HTML Application to launch the New Application wizard.
2. In the Choose Application Style dialog, accept the System Default. This dialog displays a preview of the page and a description of the style in the Style Description box. Click Next.
3. In the Finished dialog, click Finish to create the new application.
4. In the Choose Main Source of Data for New Page dialog, make sure that the Data-driven page radio-button is selected. Click the Data Objects tab, double-click the CustomerOrders group folder, and select CUSTOMERS.
5. Click Finish to accept the remaining default options.

When you click Finish, the Versata Logic Studio creates an application named Project1, creates a node pCUSTOMERS in the Application Designer, and lists the page definition in the Versata Logic Studio under HTML Applications.

To save the HTML application:

It is recommended that you save applications with a descriptive name instead of using the default Project1.

1. Choose File → Save Application.
2. In the Application Properties dialog, enter `Simple_HTML` in the Project Name field.
3. Click the OK button.

The Simple_HTML application will now appear under HTML Applications in the Versata Logic Studio Explorer.

Previewing the application

You can preview the HTML application you created by reviewing its objects in the Versata Logic Studio Explorer and the Application Designer.

- The Versata Logic Studio Explorer displays the hierarchy of repository objects. Data and query objects reside under Business Logic at the top of the Explorer. Application objects reside under specific HTML applications in the Client Applications folder.

- In the Application Designer, nodes (boxes) represent pages, and arrows represent the flow of the application between HTML pages. The Versata Logic Suite refers to these flows as transitions from one page to another.

To preview your new HTML application:

1. On the Objects tab of the Versata Logic Studio Explorer, expand the HTML Applications folder, expand Simple_HTML to display its application objects.
2. Double-click any application object in the Explorer to display its properties sheet.
3. In the application diagram, double-click the pCUSTOMERS node to launch the Page Designer. (Save the application first if the Application Properties dialog appears.)

   An editable layout of the pCUSTOMERS page appears in the Page Designer.

Executing the application

To execute an application in the Versata Logic Studio:

1. Deploy the sample data model if you have not already done so. See "Deploying the Sample Data Model" on page 25.
2. Deploy the sample business objects if you have not already done so. See "Deploying Business Objects to the Versata Logic Server" on page 47.
3. Start the Tomcat Web Server. Click the Run Web Server button at the far right of the Versata Logic Studio main toolbar.
4. Open the HTML application in the Versata Logic Studio Explorer, if it is not already open.
5. Execute the HTML application in one of the following ways.
   - Right-click the name for the open application in the Versata Logic Studio Explorer (for example, Basic_HTML), and select Execute from the shortcut menu.
   - Press F5.
   - Choose Build → Execute.
   - Click the Execute button on the toolbar.

   If an Action Choice dialog appears asking whether to rebuild the application, click Yes.

   In a successful execution, the login page for the application opens.

Note: If your application does not execute:

- Confirm that the Versata Logic Server is running. See "Starting the Versata Logic Server" on page 48.
- Verify that Database and Schema properties are specified for the data server in the Versata Logic Server Console. See the *Administrator Guide* for more information.
- Confirm that the Tomcat Web Server is running.

6. Enter sa in the login field.
7. Leave the password blank.
8. Leave the server name blank to default to the local Versata Logic Server.
9. Click the Login button. This opens the HTML application in your Web browser.
10. To close the run-time application, choose File → Close.

Adding objects to an HTML application

When you add a new page, the Add Page wizard can generate transitions to related pages. You can think of transitions as the logic that determines the flow of the application from one page to the next. For example, a transition may occur when a button is clicked.

Transitions can be quite sophisticated, as in this example, where an ORDERS grid is generated on the existing pCUSTOMERS page. In the run-time environment this ORDERS grid displays data for the current customer. Selecting a row and clicking the Orders button displays the Order page, populated with details of that order.

Opening the previously created application

To open Simple_HTML in the Versata Logic Studio:

1. In the Versata Logic Studio Explorer, expand the Client Applications folder.
2. Expand the HTML Applications folder.
3. Double-click Simple_HTML or the icon beside it to launch the application in the Application Designer.

Adding an ORDERS page that can be opened from the CUSTOMERS page

To add an ORDERS page that can be opened from the CUSTOMERS page:

1. In the Versata Logic Studio Explorer, expand Business Logic, CustomerOrders, and Queries.
2. Drag the OrderJoinSalesRep query object into empty space in the Application Designer.
3. In the Choose Display Style for OrderJoinSalesRep Data dialog, ensure that Display is selected. Click the Next button.
4. In the Choose Page Style dialog, accept Default as the page style. Click the Next button.
5. In the Choose Additional Options dialog, de-select Create Transitions and Create Parent Picks. Click the Next button.
6. In the Create Master/Detail on New pOrderJoinSalesRep Page dialog, simply click the Next button.
7. In the Create Master/Detail on Parent Pages dialog, pCUSTOMERS should be checked. Click the Finish button.
8. Execute the application, clicking Yes when prompted to save the application.

9. Login as you did earlier: enter `sa` in the name field and leave the password and server name fields blank, then click Login.
10. Explore the changes in the application.

Design time	Run time
The pOrderJoinSalesRep node appears in the Application Designer, with an arrow representing the transition from the pCUSTOMERS node.	Note that the Customer page now contains a grid that lists orders for the current customer. Select any row, and then click the pOrderJoinSalesRep hyperlink to display order details in the Order page. Click the Back button to return to the Customer page.

11. Close the run-time application.

Note: You can execute the application as often as you wish to view run-time results of changes that you have made in design time. These instructions will not request execution again until the final application object is added.

Adding a grid of PARTs to the ORDERS page

In the prior example, we created a new page to display order information. There is another option to add a new RecordSource to an existing page. This is accomplished by dragging an object onto an existing node in the Application Designer, modifying the original page instead of creating a new page.

1. Drag the query object OrderItemJoinPart onto the pOrderJoinSalesRep node.
2. In the Choose Data Dependency dialog, click Finish to accept all defaults for the new RecordSource. Or click the Next button, review all succeeding dialogs, and accept the defaults in each one. This latter option allows you to review all of the choices available.

Adding another grid of ORDERS to the CUSTOMERS page

In this step, you will add a second instance of the RecordSource ORDERS to the pCUSTOMERS page, but use a different filter to display unpaid orders. The Versata Logic Studio automatically generates tab sheets on the page to display multiple instances of a query object.

To add a grid of unpaid orders to the CUSTOMERS page:

1. Drag the query object OrderJoinSalesRep onto the pCUSTOMERS node.
2. In the Choose Data Dependency dialog, click the Next button.

3. In the Choose Display Style for OrderJoinSalesRep Data dialog, click the Filter and Sort Records button. The Expression Builder dialog appears.
4. Double-click the object ORDERS.OrderPaid (bottom left of screen in the Attributes tab).
5. Click the operator = (from the two rows of equation buttons).
6. Expand ORDERS.OrderPaid (selected previously) to display its two Boolean children and double-click `No (False)`.

   In the Selection Condition field, the expression `OrderPaid = False` should now appear.
7. Click the OK button to apply the expression and close the Expression Builder.
8. Click the Finish button to complete and close the wizard.

Changing transition prefixes

In this step, you change the prefixes for the transitions to be more descriptive.

To change transition prefixes:

1. In the Applications folder of the Versata Logic Studio Explorer, under Simple_HTML, expand pCUSTOMERS.

   You may wish to drag the border of the Explorer to the right in order to see the full names of all of the child objects of pCUSTOMERS.
2. Double-click the transition T4OrderJoinSalesRep (identified by the green arrow icon) to display its properties sheet.
3. Change the prefix from T4 to `All`.
4. Click the transition T6OrderJoinSalesRep.
5. Change the prefix T6 to `Unpaid`.
6. Close the properties sheet.

   Note that the names of the application objects in the Explorer reflect the prefix edits.

Adding a PART page

To add a PART page:

1. Drag the data object PART onto empty space in the Application Designer.
2. In the Choose Display Style for PART Data dialog, select Grid, and click the Finish button.
3. In the post-wizard dialog that asks “Would you like to create a transition from the Start Page?”, click the Yes button.
4. Execute the application, clicking Yes if prompted to save the application.

5. Login as you did earlier: enter `sa` in the name field and leave the password and server name fields blank, then click Login.
6. Explore the changes in the application.
   - On the main page of the application, click the Customer link.
   - On the Customer page, notice there are two grids: one listing all orders and one listing unpaid orders.
   - Click the AllpOrderJoinSalesRep link to open a page displaying more information about the selected order in the grid of all orders.
   - Click the Back button to return to the Customer page.
   - Click the UnpaidpOrderJoinSalesRep link to open a page displaying more information about the selected order in the grid of unpaid orders.
   - Click the Parts hyperlink to open the Parts page with a list of parts in a grid.
7. When you are done exploring, close the run-time application and the design-time application diagram.

Note: After you have executed the application, the application diagram includes a node for the Frameset page. This is a system-supplied page used to contain all pages in the application. Feel free to adjust the position of nodes as they appear in the application diagram. so that the nodes and transition lines do not overlap.

Comparing design-time and run-time application features

The HTML application displayed in the Web browser is referred to as the run-time application. The same application displayed in the development environment is designated as the design-time application.

You can compare features and tools by toggling (ALT+TAB) between the design-time and run-time application displayed in the Versata Logic Studio development environment.

Comparing features in a sample application

To explore the HTML run-time features and design-time tools for HTML applications:

1. Open the sample repository if you have not already done so.
2. Deploy the sample data model if you have not already done so.
3. Deploy the sample business objects' transaction logic if you have not already done so.

   The previous steps are required in order to run applications from `SampDB1.xml`.
4. Open the Basic_HTML sample application in the Application Designer.
5. Execute the HTML application. If you need instructions, see "Executing the application" on page 57

   You now have run-time and design-time open for the same application, and can toggle between these two environments by pressing ALT+TAB.
6. Explore page transition properties using the suggestions that follow.
7. Explore picks used in pages using the suggestions that follow.
8. Explore editable grids using the suggestions that follow.
9. Close the application using the following:
   - In design time, click File → Close Application. Optionally, click the Yes button to save changes.
   - In run time, choose File → Close.

Exploring page transition properties

Transitions are most easily visualized by comparing the connecting lines in the Application Designer with the jumps that occur when you click hyperlinks in the Web browser. You can also edit a grid to create hyperlinks on elements within a grid.

Design time	Run time
In the Basic_HTML folder of the Versata Logic Studio Explorer, expand the StartupPage to show its children. There are two transitions: T1pCUSTOMERSQBF and T2pPART. Double-click the T1pCUSTOMERSQBF transition to open its Page Transition Properties sheet. The properties sheet indicates this is a transition to the pCUSTOMERSQBF page and that the page will open in Query mode. Close the properties sheet.	Click the `Customer` hyperlink, then the `Find` hyperlink to perform a query to get all records.
Still in the Basic_HTML folder, scroll to the pCUSTOMERSQBF page, then expand it to show its children. Double-click the transition T3pCUSTOMERMenuList to launch its Page Transition Properties sheet. The properties sheet indicates this is a zoom transition, meaning that it re-uses the current data to populate the target page. Close the properties sheet.	Click one of the hyperlinks on the left. Then, click the Order hyperlink at the bottom of the page. Notice that transitioning to the Order page provides more detail about the orders on the Customer page.
Now expand pCUSTOMER_Orders and double-click T3OrderJoinOptSalesRep to display the properties sheet. Select the Attributes tab. Click on the attribute OrderNumber. Note that in the Archetypes field the HyperlinkForGrid is selected, and T4pOrder_Items is selected for the Transition Object to Use list box. Close the properties sheet.	Click the Back button to return to the Customer page. Click the hyperlink for any Order#. This jumps you directly to the Order page for that number.

Note: Be sure to close any instances of pages, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring picks used in pages

Examine the use of picks to go to a new page in HTML applications.

Design time	Run time
Expand pOrder_Items, then expand T1OrderJoinOptSalesRep to show its children. Right-click T3EMPLOYEES and choose Properties to view the Pick Object Properties sheet. Review the settings for this pick object, then close the properties sheet.	The Order page should already be open in the browser. This page contains the pick for SalesRep ID. Click the arrow next to the text box labeled SalesRepID to go to the Choose page, and click the Select hyperlink for a different sales representative. The Choose page closes, and the Order page reappears with the newly selected SalesRep ID displayed.

- At design time, picks are automatically enabled during page creation. To disable picks, disable the Create Parent Picks radio button in the wizard dialog Choose Additional Options.
- At run time, users click the arrow button next to the pick text box to go to a new page where they can make another selection.

Note: Be sure to close any instances of pages, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring editable grids

Compare design-time changes made in Application Designer with the changes reflected at run time.

Editable grids allow users to modify attributes at run-time in their Web browsers.

Design time	Run time
Expand pOrder_Items to show its children. Double-click T4OrderItemJoinPart to open the RecordSource Properties sheet. Click the Attributes tab. Select QtyOrdered. Notice that the archetype is a GridTextBox. Now, click the HTML Object tab and see that Allow Delete, Allow Insert, and Allow Update are selected. Close the properties sheet.	On the Order page, notice that the Qty Ordered amounts appear in a text box that you can edit. Change the Qty Ordered for any order and click the OK button. The Versata Logic Suite automatically updates the Amount field.

LESSON 6

Creating Java Applications

Overview

Applications are modeled using the Application Designer. Included in the Application Designer are wizards that create the application objects, an application diagram that displays the flow of the application, and properties sheets that provide easy access for setting the properties for the objects. The Versata Logic Studio Explorer provides access to the application objects.

The Versata Logic Studio wizards enable basic user interface modeling with the simplicity of drag-and-drop, yet with considerable control over the outcomes. Applications are defined using the wizards and refined in the Form Designer and the Code Editor.

Wizards gather information about the specific data to be presented, how it is to be presented, and the flow of the application. The resulting application design is displayed in the application diagram.

Versata Logic Suite objects possess properties that correspond to those available in their Java classes. The properties and their values are listed in properties sheets.

This section includes the following tasks:

- “Creating a simple Java application” on page 69.
- “Adding objects to a Java application” on page 72.
- “Comparing design-time and run-time application features” on page 76.

The following tasks are considered advanced:

- “Implementing specialized controls and Java code” on page 82.
- “Working with transitions” on page 87.
- “Working with Java Beans” on page 90.

Creating a simple Java application

For this task, you create a simple, one-form, Java application that presents CUSTOMERS information in scalar fields with query capability.

Designing the application

To create a simple Java application:

1. Choose File → New Java Application to launch the New Application wizard.
2. In the Choose Application Style dialog, accept the System Default. This dialog displays a preview of the page and a description of the style in the Style Description box. Click the Next button.
3. In the Finished dialog, click the Finish button to create the new application.
4. In the Choose Main Source of Data for New Form dialog, make sure that the Data-driven Form is selected. Click the Data Objects tab, double-click the CustomerOrders group folder, then select CUSTOMERS.
5. At this point, you can click the Finish button to accept all of the wizard's defaults for the form. Or, you can click the Next button and review each dialog to see the choices available, then click the Next button to continue, and click the Finish button in the final dialog.

To save the Java application:

It is recommended that you save applications with a descriptive name instead of using the default Project1.

1. Choose File → Save Application.
2. In the Application Properties dialog, enter `Simple_Java` in the Project Name field.
3. Click the OK button.

The Simple_Java application will now appear under Java Applications in the Versata Logic Studio Explorer.

Previewing the application

You can preview the Java application you created by reviewing its objects in the Versata Logic Studio Explorer and the Application Designer.

- The Versata Logic Studio Explorer displays the hierarchy of repository objects. Data and query objects reside under Business Objects at the top of the Explorer. Application objects reside under specific Java applications (for example, Simple_Java) in the Client Applications folder.

- In the Application Designer, nodes (boxes) represent Java forms, and arrows represent the flow of the application between Java forms. The Versata Logic Studio refers to these flows as transitions from one form to another.

To preview your new Java application:

1. In the Java Applications folder of the Versata Logic Studio Explorer, expand Simple_Java to display its application objects.
2. Double-click any application object in the Explorer to display its properties sheet.
3. Double-click the fCUSTOMERS node to display its corresponding form. An editable layout of the form appears in the Form Designer, with access to graphical tools, property sheets, and a code editor.
4. Click any control in the fCUSTOMERS form to view its properties in the Properties dialog. If the Properties dialog is not displayed, press F4.

Note: It may be necessary to drag the corner of the properties sheet to view all of the properties.

5. Right-click a control and choose Events to display editable Java code at the control level.
6. Close the Form Designer.

Executing the application

You can run a Versata Logic Suite Java application to review its user interface, if you have deployed the data model and business rules for the application's repository.

To execute a Java application:

1. Deploy the sample data model and business objects if you have not already done so. These deployments are required to run applications in the sample repository. For instructions, see "Deploying the Sample Data Model" on page 25 and "Deploying Business Objects to the Versata Logic Server" on page 47.
2. Open the Simple_Java application in the Versata Logic Studio Explorer, if it is not already open.
3. Choose Application → Properties. In the Application Properties dialog, click the Options tab. Notice that Applet Viewer is selected for Execution Environment, then click the OK button.

4. Execute the Java application in one of the following ways.
   - Right-click the name for the open application in the Versata Logic Studio Explorer (for example, Basic_Java), and select Execute from the shortcut menu.
   - Press F5.
   - Choose Build → Execute.
   - Click the Execute button on the toolbar.

   If an Action Choice dialog appears asking whether to rebuild the application, click Yes.

   The Versata Logic Studio displays notes in the lower panel to indicate its progress in building and compiling the application. Once the build and compile is complete, the applet viewer runs and a login dialog appears.

Note: The Versata Logic Server, and the Versata Logic Server Locator must be running in order to successfully run an application. If the application does not run, verify that you have started both of these, then try again. For information, see "Starting the Versata Logic Server" on page 48.

If the application does not run, check the Database and Schema properties for the data server in the Versata Logic Server Console.

5. In the Login dialog, enter `sa` in the Login field, leave the other two fields blank, and click the OK button.
6. Maximize the application window. You may need to click the title bar of the application to bring it into the foreground. Do not close the DOS window at this time.
7. Notice the toolbar button at the top of the application's form. You can click the Search Mode button, then enter criteria to return records containing specified data.
8. To close the run-time application, click the X in the upper right corner, standard for Windows applications.

Adding objects to a Java application

In this section, you work with the application named Simple_Java that you created in the previous section of this tutorial. This application currently consists of one fCUSTOMERS form. The tasks in this section require you to modify and extend Simple_Java, to create an order entry system that includes several related forms.

These tasks illustrate how to modify a Java application in the Versata Logic Studio and how to create a Java application that includes multiple forms.

To add objects to a Java application:

1. If it is not already open, open the sample repository using File → Open Repository.
2. Open the Simple_Java application in the Versata Logic Studio. If you need instructions, see "Opening the previously created application" on page 72.
3. Add a form to display ORDERS data that can be opened from the fCUSTOMERS form. For instructions, see "Adding a form displaying ORDERS that can be opened from the fCUSTOMERS form" on page 73.
4. Add a grid of PART data to the form that displays ORDERS data. For instructions, see "Adding a grid of PART data to the fOrderJoinSalesRep form" on page 74.
5. Add another grid of ORDERS data to the fCUSTOMERS form, to display unpaid ORDERS data.
6. Add a form to display PART data. For instructions, see "Adding a form to display PART data" on page 75.
7. Compare the design-time and run-time Java application. For instructions, "Comparing design-time and run-time application features" on page 76.
8. Close the application.
   - In design time, click File → Close Application.
   - In run time, click the X in the upper right corner.

See also "Implementing specialized controls and Java code" on page 82, "Working with transitions" on page 87, and "Working with Java Beans" on page 90.

Opening the previously created application

To open Simple_Java in the Versata Logic Suite:

1. Expand the Client Applications folder in the Versata Logic Studio Explorer.
2. Expand the Java Applications folder.
3. Double-click Simple_Java or the icon beside it to launch the application in the Application Designer.

Adding a form displaying ORDERS that can be opened from the fCUSTOMERS form

When you add a new form, the Add Form wizard can generate transitions to related forms. You can think of transitions as the logic that determines the flow of the application from one form to the next. For example, a transition may occur when a button is clicked.

Transitions can be quite sophisticated, as in this example, where an ORDERS grid is generated on the existing fCUSTOMERS form. In the run-time environment this ORDERS grid displays data for the current CUSTOMERS record. Selecting a row and clicking the OrderJoinSalesRep button displays the fOrderJoinSalesRep form, populated with details of that record.

Note: Instead of the ORDERS data object, a query object called OrderJoinSalesRep that contains ORDERS and EMPLOYEES data is used as the RecordSource (source of data) for the new form. For information about query objects, see the *Business Object Developer Guide*.

To add a form displaying ORDERS data that can be opened from the fCUSTOMERS form:

1. In theVersata Logic Studio Explorer, expand Business Logic, CustomerOrders, and Queries.
2. Drag the OrderJoinSalesRep query object into empty space in the Application Designer.
3. In the Choose Display Style dialog, you can click the Finish button to accept all defaults for the new form. Or you can click the Next button, review all succeeding dialogs, and accept the defaults in each one. This latter option allows you to review all of the choices available.
4. Execute the application, clicking Yes if prompted to save the application. If you need instructions, see “Executing the application” on page 70.
5. Explore the changes in the application.

Design time	Run time
The fOrderJoinSalesRep node appears in the Application Designer, with an arrow representing the transition from the fCUSTOMERS node.	Note that the Customer form now contains a grid that lists orders for the current customer. Select any row, and then click the fOrderJoinSalesRep button to display order details on the fOrderJoinSalesRep form. Close the application.

Note: You can execute the application as often as you wish to view run-time results of changes that you have made in design time. These instructions will not request execution again until the final application object is added.

Adding a grid of PART data to the fOrderJoinSalesRep form

When an object is dragged onto a node in the Application Designer, a new form is not created. Instead, a new RecordSource is added to the existing form.

Note: Instead of the PART data object, a query object, OrderItemJoinPart that contains PART and ORDERITEM data is used as the RecordSource for the new grid. For information about query objects, see the *Business Object Developer Guide*.

To add a grid of PART data to the fOrderJoinSalesRep form:

1. Drag the query object OrderItemJoinPart onto the fOrderJoinSalesRep form that you created in the prior step.
2. In the Choose Data Dependency dialog, click the Finish button to accept all defaults for the new RecordSource. Or click the Next button, review all succeeding dialogs, and accept the defaults in each one. This latter option allows you to review all of the choices available.
3. Click the Save button.

Adding another grid of ORDERS data to the fCUSTOMERS form

In this case, a second instance of the RecordSource ORDERS (OrderJoinSalesRep) is added to the fCUSTOMERS form, but with a different filtering to display unpaid orders. The Versata Logic Studio automatically generates tab sheets on the form to display multiple instances of a RecordSource.

To add a grid of unpaid ORDERS data to the fCUSTOMERS form:

1. Drag the query object OrderJoinSalesRep onto the fCUSTOMERS node.
2. In the Choose Data Dependency dialog, click the Next button.
3. In the Choose Display Style dialog, click Filter and Sort Records. The Expression Builder appears.
4. Define a Where clause in the Expression Builder by performing the following steps:
   - Double-click the object OrderPaid (bottom left of screen in the Attributes tab).
   - Click the operator = (from the two rows of equation buttons).
   - Click the + control adjacent to OrderPaid (selected previously) to display its two Boolean children and double-click `No(False)`.
   - In the Selection Condition field, the expression `OrderPaid = False` should now appear.
   - Click the OK button to apply the expression and close the Expression Builder.
5. Click the Finish button to complete and close the wizard.

6. Click the Save button to preserve application changes before moving to the next step.
7. Edit the prefixes for the fCUSTOMERS form transitions by performing the following steps:
   - In the Java Applications folder of the Versata Logic Studio Explorer, under Simple_Java, expand fCUSTOMERS.
   - You may wish to drag the border of the Explorer to the right in order to see the full names of all objects on the fCUSTOMERS form.
   - Double-click the transition T4fOrderJoinSalesRep to display its properties sheet. Change the Prefix for Controls from T4 to `All.`
   - Double-click the transition T6fOrderJoinSalesRep. Change the Prefix for Controls from T6 to `Unpaid.`
8. Close the Form Transition Properties sheet.
9. Click the Save button.

   Note that the names of the application objects in the Explorer reflect the prefix edits.

Adding a form to display PART data

To add a form to display PART data:

1. Drag the data object PART onto empty space in the Application Designer.
2. In the Choose Display Style dialog, select Grid, and click the Finish button.
3. In the post-wizard dialog Would you like to create a transition from the Start Form?, click Yes.
4. Click the Save button.
5. Execute the application. If you need instructions, see “Executing the application” on page 70. Now you can compare design-time and run-time application features. For more information, see “Comparing design-time and run-time application features” on page 76.

Comparing design-time and run-time application features

In this section you review run-time features in your newly created Simple_Java application, toggling to the design-time environment to see how these features are represented there.

Then, you review more advanced application features in design time and run time, using the _Demo application and other samples included in the sample repository.

Comparing features in the newly created application

To compare the design-time and run-time application:

1. Execute the Simple_Java application. If you need instructions, see "Executing the application" on page 70.

   After execution, you can toggle between the design-time and run-time applications by pressing ALT+TAB.
2. Note the display of the StartupForm in both the run-time and design-time environments.

Design time	Run time
The Application Designer displays a StartupForm node, even if there is just one target form from the start.	The StartupForm appears only when there are multiple paths into an application. In the current case, where the user can enter the application through the Customer or Parts forms, buttons for both are provided on the StartupForm when you run the application. Otherwise, as in the single-form application in the first exercise, no StartupForm appears.

3. Note how business rules are implemented.

   You do not need to write code for transaction logic, as it is stored in the Versata Logic Server as business rules. Because business rules reside on the Versata Logic Server, they apply across applications, and are automatically applied to all relevant transactions. These rules are centrally enforced in the Versata Logic Server, avoiding the performance and integrity issues associated with fat-client applications.

4. Examine business rules in the repository.
   - In the run-time application, display the Parts form, and then click any cell in the Type attribute. The cell becomes a combo box because the business rules for this attribute identify a set of valid values.
   - To see the business rules underlying this behavior, toggle to the design-time application, then double-click the PART data object (in the Explorer under Business Objects, Data Objects) to display the Transaction Logic Designer for this object.
   - Under the Attributes tab, click the grid row Type, and then click the Validation/Data Type tab. Upon selection, the Validation/DataType tab (lower half of the screen) opens and shows that Validation Type is a Coded Values List, with a finite list of valid values. The Versata Logic Studio generates fields that access coded values lists as combo boxes.
5. Close the run-time application.

Note: You can browse the Transaction Logic Designer Designer at this time to review some of the business rules acting on the repository. Business rules are discussed specifically in "Exploring Business Rules" on page 35.

Once you deploy business objects to the Versata Logic Server, their transaction logic is enforced whenever an end user adds a new order. For example, business rules would automatically reorder a PART if its inventory is at the reorder point, adjust the ORDER amount, adjust the CUSTOMER's balance, and check the CUSTOMER's balance against the credit limit.

Exploring more advanced application features

In this section, you review features of the _Demo application in the sample repository.

Before you proceed through the tasks in this section, you need to execute the _Demo sample application. Follow the steps below.

To execute the _Demo sample application:

1. In the Versata Logic Studio, close the Simple_Java application, then open _Demo.
2. Then, execute the application. If you need instructions, see "Executing the application" on page 70.

   After execution, you can toggle between the design-time and run-time applications by pressing ALT+TAB.

Exploring tab sets

Task: Notice how the Versata Logic Suite handles multiple instances of a RecordSource in the same form.

- RecordSources (data or query objects that are the source of data appearing in a form) can have properties that function as filters.
- Because RecordSources can be filtered to display multiple sets of information, the same RecordSource can appear more than once in the same form; for example, on the CUSTOMER form, where the Paid Orders and Open Orders tabs represent different filters of data from the same query object, OrderJoinOptSalesRep.

Design time	Run time
In the _Demo folder of the Versata Logic Studio Explorer, expand the CUSTOMER form. Under T1CUSTOMERS, right-click T2OrderJoinOptSalesRep and choose Properties. Under the Query tab, note the Additional Where Clause (on the Query tab) of `OrderPaid = False.` Repeat for T4OrderJoinOptSalesRep and notice the Additional Where Clause of `OrderPaid = True.`	Select a customer on the Customers form and click the Details hyperlink A Customer form that includes Open Orders and Paid Orders information is displayed. Click each tab to review its contents. A design note: The Order Detail hyperlink and the Orders button used to zoom to scalar order detail were placed inside the Open Orders and Paid Orders tab sheets, in order to avoid user confusion.

Note: Be sure to close any instances of forms, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring picks used on forms

Task: Examine the use of picks in forms to populate a foreign key for Java applications.

- Users pick a foreign key from a combo box. The selected foreign key populates the combo box. Related fields are populated with values from the row of the data object where the foreign key resides as a primary key.

- Picks are automatically enabled during form creation. To disable picks, disable the Create Parent Picks radio button in the wizard dialog Choose Additional Options.
- The Versata Logic Studio represents picks as a text box with an arrow next to it.

Design time	Run time
In the _Demo folder of the Versata Logic Studio Explorer, click the + beside the OrderJoinSalesRep form to display the objects it contains. Then right-click the T2EMPLOYEES pick object and choose Properties. On the Query tab, note the Additional Where Clause of `EmpType = 'C'`, which specifies that only Sales Reps are valid candidates for the pick.	On the Open Orders grid on the Customer form, pick the first order and click the Order Detail hyperlink. The Order form appears. On the Order form, note that the Sales Rep field provides pick functionality. (Even though it is not the foreign key, it is joined from the EMPLOYEES data object.) From the combo box, pick another Sales Rep by double-clicking its row in the popup Choose grid, and note that the name and corresponding Sales Rep ID fields are populated.

Note: Be sure to close any instances of forms, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring an event

Task: Compare the bar chart in the run-time Customers form with its editable event code in the design-time Code Editor.

Visual elements in Versata Logic Suite applications are implemented using Java events. The bar chart uses `drawGridCell`.

Design time	Run time
Select the CUSTOMERS_Grid node in the application diagram and double-click to open the form. Right-click the grid and choose Events. If it is not already displayed, select `grdT1CUSTOMERS` from the Controls drop-down list. Then choose `drawGridCell` from the Events drop-down list. The Java code for the `drawGridCell` event is displayed in the editing pane of the Code Editor.	Click the Customer image button to launch the Customers form. In the Balance attribute, note the bar chart. Click the column heading once to sort in increasing order; click again to sort in decreasing order.

Note: Be sure to close any instances of forms, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring form transition properties

Task: Notice how the behavior and filtering of transitions are controlled by transition properties.

Transitions are most easily visualized by comparing the connecting arrows in the Application Designer with the jumps that occur when form buttons are clicked.

Design time	Run time
In the _Demo folder of the Versata Logic Studio Explorer, expand the StartupForm to show its children. Double-click the transition T6fDepartment to launch its properties sheet. The Additional Where Clause (on the Query tab) limits choices to those departments with no head department. Still in _Demo, expand the fDEPARTMENT form to display the objects it contains. Double-click the T8fDepartment transition to open the transition properties sheet. The properties sheet indicates that this is a zoom transition, meaning that it re-uses the current data to populate the target form.	In the run-time application, click the Departments image button to launch the Department form. Note that the scalar field Name contains Executive. The grid below lists immediate sub-departments of the Executive department. In the grid, click the Sales row, and then click the Departments button. This hyperlink launches—or transitions to—a new instance of the Department form, showing Sales at the top, with Sales sub-departments (such as NE Sales and SE Sales) in the grid.

Note: Be sure to close any instances of forms, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Exploring form-level coding

Task: Compare design-time methods and run-time behavior of type hierarchies.

In addition to one-to-many and many-to-many relationships, Versata Logic Suite data models also support Super/Sub relationships, where a supertype of an object contains subtypes that inherit the behavior of the supertype and extend it. A Super/Sub relationship, commonly referred to as a type hierarchy, is not a relationship by the traditional relational definition, because it does not join two data objects.

The following is an example of a type hierarchy: the supertype is Employee and the subtypes are Salaried, Hourly, and Commissioned. These types share certain fields in a form. Some types also have type-specific fields. Type hierarchies in Versata Logic Suite applications are implemented using form-level methods that are accessible to all controls on the form.

Design time	Run time
With the _Demo application open in the Application Designer, double-click the node EmployeeList to display it in the Form Designer. Right-click the Employee data control and choose Events. In the Code Editor, choose the event `AfterReposition` to display its event-handling code. Next, choose the event `AfterColUpdate` to display its event-handling code. Note that the code for both events invokes the method `ShowEmpTypePanel`. This method is defined at the form level, and is invoked by both controls associated with type hierarchies.	In the run-time, click the Employees image button to display the Employees form. Click the Next button (right arrow) twice, to scroll through the list of employees to Mary Joe Bucks. When you access Ms. Bucks, additional type-specific scalar fields are displayed below the shared fields in the form.

Note: Be sure to close any instances of forms, properties sheets, or the Code Editor in the design-time environment before proceeding to the next step.

Implementing specialized controls and Java code

In this section, you explore the Versata Logic Suite's Tree control, as well as event handlers for query behavior and control of focus in a Java application. Throughout this section, you toggle between the design-time and run-time elements of the sample application Client_Tree_Customers to review several key customizations of controls and code.

To explore specialized Versata Logic Suite controls and Java code:

1. Open the Client_Tree_Customers application in the Application Designer.
2. Execute the application. If you need instructions, see "Executing the application" on page 70.

   You now can toggle between the compiled Client_Tree_Customers and its development environment by pressing ALT+TAB.
3. Explore query behavior coding. For instructions, see "Exploring code for query behavior" on page 83.
4. Explore Tree control implementation. For instructions, see "Exploring Tree control implementation" on page 84.
5. Explore coding behavior for the Tree control. For instructions, "Exploring code for VSTree behavior" on page 85.
6. Explore coding focus. For instructions, see "Exploring coding focus" on page 86.
7. Close the application.

Exploring code for query behavior

In Java applications, source forms are generally moved to the background when a query is executed. In this case, the query form is closed when the query specification is complete.

The Find Customers (CustomersSearch) form provides services to select the desired customer to view in the Customer and Order Details (CustOrdersinTree) form. End users can define selections with one or more criteria, including a series of ranges and OR terms. (For example, a user could select customers from Massachusetts or California with balances greater than $1000.) All of this query functionality is completely automated, so no code is required.

One element of the form behavior did require coding. This application implements closure of the query form using the `actionPerformed` event.

Compare query form behavior against the code for the query.

Design time	Run time
In the Application Designer, double-click the CustomersSearch node to display the query form. Right-click the FindCustomers button and choose Events to display the `actionPerformed` event for the VSImageButton1 control. This event code executes a query, launches the target form with the result set, and then closes the query form. Close the Code Editor.	In the run-time, click the Customers image button to launch the FindCustomers form. A query form for CUSTOMERS data appears. In the query form, click the Find Customers button. Note: Since you did not specify any criteria for the query, the query returns all customer orders in Tree view form.The Customer and Order Details form appears. Note that the query form is not active after the query is complete.

Exploring Tree control implementation

The tree view of data provided is generated declaratively through property sheets; no code is required. Tree nodes can be mapped to labels or to data objects. The labels are useful for grouping similar data objects; for example, Open Orders and Paid Orders.

Compare displayed values in the run-time `Customer Orders` tree with design-time node properties for the Tree control.

Design time	Run time
In the Application Designer, double-click the CustOrdersInTree node to display its form. Right-click the Tree control, then choose Nodes. The Tree Nodes properties sheet appears. Expand the displayed nodes by clicking the + sign. In turn, click each displayed node to select it. When selecting the constant text nodes (Open Orders, Paid Orders), note that they have no associated Data Source / Data Field values on the right side of the Nodes properties sheet. When selecting the nodes that display data (VSTreeNode3, VSTreeNode5), note that each is mapped to a Data Source, and that each is associated with a specified Data Field (header) whose value is placed into the tree at run time.	Still on the Customer and Order Details form, expand a customer record to show its children. Repeat to show Open Orders and Paid Orders children. Specific order numbers from the repository are displayed. Select an order number to view details about it.

Exploring code for VSTree behavior

Multiple display options for a specific location on a form can be stacked in the Form Designer and displayed under varying conditions using the nodeSelected event.

Notice how the VSTree control choices affect the data presented on the right side of the associated form, and how this is coded.

Design time	Run time
In the Application Designer, double-click the CustOrdersInTree node to display the form, then right-click the VSTree control and choose Events to display the `nodeSelected` event for the VSTree1 control. In the Code Editor, note that the `nodeSelected` event shows/hides the appropriate panel (`custPanel, openOrdersPanel, paidOrdersPanel`) depending on the DataSourceName for the selected node. Close the Code Editor. In the Form Designer, on the right side of the form, note that all three display panels are defined on the same form and placed in the same area. Cycle between the overlapping panels by selecting the panel area, and then right-clicking to choose Send To Back.	Still on the Customer and Order Details form, expand Bill's AutoBody, then expand Bill's Autobody's Open Orders and Paid Orders. Click the order numbers in the Open Orders and Paid Orders nodes. As you do so, note that selection-specific displays appear on the right side of the form. Different displays are provided for selected Customers, for Open Orders, and for Paid Orders.

Exploring coding focus

Toolbar buttons typically operate on data that has the current mouse focus. Buttons that operate on the currently open RecordSource—for example, an open order—also can be coded.

Notice how the Versata Logic Studio locates focus and how this action is coded.

Design time	Run time
In the Application Designer, double-click the CustOrdersInTree node to display the form. As needed, right-click the display panel on the right side of the form, and choose Send to Back until the Open Orders Detail panel is displayed. In the Open Orders Detail panel, right-click the Image button with the Save icon on it and choose Properties. Note that the DataSource property specifies the data control `datT4OrderItemJoinPart,` which specifies the RecordSource focus. Note also that the DataSource property is a combo box. Upon clicking the arrow on the combo box, a number of options are available, including current, which is used to define focus-specific database operations.	Still on the Customer and Order Details form, select an open order. Note that the selection-specific area on the right includes several Image Buttons. These buttons are enabled only when a RecordSource is activated and act only on that currently active RecordSource.

Working with transitions

In this section, you explore a number of customization features that involve variations of transition properties, using the _Transitions sample application. This sample application illustrates various elements of Versata Logic Studio declarative support for form transitions, including the Initial Mode and Query properties. Initial Mode enables you to control how the target form operates when it is started, and whether it allows you to add, query or browse data. Query properties enable you to filter the rows on the target form. All of the features in this application are declarative; they require no code.

To explore transition properties:

1. Open the _Transitions sample application in the Application Designer.
2. Execute the application. If you need instructions, see "Executing the application" on page 70.

   You now can toggle between the _Transitions application and the development environment by pressing ALT+TAB.
3. Review transition types. For instructions, see "Reviewing transition types" on page 87.
4. Review zoom transition directions. For instructions, see "Reviewing zoom transition directions" on page 88.
5. Review initial form behavior properties. For instructions, see "Reviewing initial form behavior properties" on page 88.
6. Review grid versus query behavior. For instructions, see "Reviewing grid versus query behavior" on page 89.

Reviewing transition types

In this section, you compare two different types of transitions for the same data in the same form, consider the advantages of each, and learn how to change the type of transition. The Versata Logic Studio provides two types of transitions:

- Show Related, which initiates a query whose results are displayed on the target form. This type is appropriate when data is not already displayed on the source form. For example, it could be used to display details about a part for a line item in an order.
- Zoom, which typically launches a detail form that tells more about a selected row in a grid. No query is required, as the target form shares the current query results from the source form. In practical terms, performance is faster, as no query is required. Changes on the target form (including insertions and deletions) are reflected in the source grid.

Reviewing zoom transition directions

Notice that Zoom transitions can move in either direction between Display and Grid style RecordSources.

These transitions are implemented in design time and run time as follows:

Design time	Run time
In the application diagram, double-click the CUSTOMERS_Grid node to open it in Form Designer. Right-click the Customer Detail button (located at the bottom of the page), and then click Properties. Note the value automatically chosen for the Form Transition property.	From the Hello Employee form, click Find a Specific Customer button. The Customers form opens in Query Mode. Enter some criteria (California, for example), and click the Get Data button. The form now displays customers that match the query criteria, with their orders. To see all the customer records that match the query, press the Customers button.

Reviewing initial form behavior properties

Notice the different initial behaviors of several forms and how properties determine this behavior.

Transition properties determine initial form behavior. A form may launch in Query, Browse, or Add mode. The launch node is specified in the field Initial Mode on Transition on the Form Transition Properties sheet.

Design time	Run time
In the Explorer, click the + next to the _Transitions application. Note that since there is only one transition from the StartUp node, this form will be bypassed when the application runs.	The first form displayed is fEmpHello.
In the Explorer, click the + next to fEmpHello. Right-click FindSpecificCustomerCUSTOMER and select Properties. The field Initial Mode on Transition shows Query mode is selected.	Click Find a Specific Customer to launch a Customer Query form. Use the drop down list to select a state. Click the Get Data button and view the customers from the selected state. Close this form to return to the Hello Employee form.

Design time	Run time
There are several more transitions in fEmpHello. Right-click ListCustomersImmediateCUSTOMERS_Grid and select Properties. The field Initial Mode on Transition shows Browse mode is selected. Note the caption List Customers (immediately) is also displayed on this form.	On the Hello Employee form, click List Customers (immediately) to view all customers. Then close this form to return to the Hello Employee form.
Also under fEmpHello, right-click AddOrderORDER and select Properties. The field Initial Mode on Transition shows Add mode is selected. Note the caption Add a New Order is also displayed on this form.	Click Add a New Order to display the Order form. Note that no orders are displayed and you are able to enter a new order. Then close this form to return to the Hello Employee form.

Reviewing grid versus query behavior

Fields in a form can display RecordSource data or the current query specification. Data fields always show current data for the field, while query fields always shows the current selection criteria (if any).

This behavior is specified in control-level properties.

Compare two otherwise identical fields with different display modes, in run time and in the Control Properties of the Form Designer.

Design time	Run time
In the application diagram, double-click the CUSTOMERS_Grid node to display it in Form Designer. Right-click the Select State field (which is a VSChoice control), and then choose Properties. The DisplayMode property is Query. By contrast, display Properties for the State field below the grid. The DisplayMode is Data.	In the run-time application, click Find a Specific Customer to launch the Customers form in query mode. Click Get Data to display the first customer in the list.

Working with Java Beans

Java Beans, or Beans, are interchangeable, reusable components that add functionality to Java applications. Their architecture is modeled on the same reusable component architecture that is used in other reusable software components, such as OCXs and widgets. Like these other components, Beans have properties and methods that may be addressed through property sheets and APIs. Beans are fully implemented as of JDK 1.1.8.

Beans are design-time components, unlike classes, which are run-time components. That is, Beans are used by the Versata Logic Suite in design time. Classes and instances of classes are used by the Java VM at run time.

To use a Bean in a form:

1. Register the Bean in the Versata Logic Studio. For instructions, see "Registering a Bean in the Versata Logic Studio" on page 90.
2. Add the Bean to the form. For instructions, see "Adding the Bean to a form" on page 91.
3. Assign properties to the Bean. For instructions, see "Assigning properties to the Bean" on page 91.

For an example of a Versata Logic Suite application that incorporates a Bean, see the Client_JavaBean sample application in the sample repository.

Registering a Bean in the Versata Logic Studio

To register a Bean:

1. From the Versata Logic Studio menu, choose Tools → Add Object to System Registry.
2. In the Add Object to System Registry dialog, select the JavaBean option, then click Add.
3. In the Choose Java Bean dialog, browse to the directory where the `.jar` or `.class` file for the Bean is located.

   A sample bean is located in
   `<versata_install>\Samples\SampDB1\Source\ClientApps\JavaApps\Client_JavaBean.app\molecule.jar`
4. Select the Bean file and click the Open button.
5. Click the OK button when the message box confirms that the Java Bean has been added.

Adding the Bean to a form

Before adding a Bean to a form, register the Bean in the Versata Logic Studio.

To add a JavaBean to a Versata Logic Studio form:

1. Choose File→New Java Application. In the first wizard dialog that appears, click Finish to accept all defaults. In the second wizard dialog, choose Empty form and click Finish.
2. In the application diagram, double-click the fNoData node to open it in Form Designer.
3. Choose View → ToolBars → Beans to open the Beans toolbar.
4. In the toolbar, click the desired Bean button.
5. In the form, click the desired location for the Bean.
6. To make the Bean larger or smaller, click on one of the corners of the Bean control and drag it.
7. To move the Bean, position the cursor over the Bean until the crossed arrows appear, then click and drag the Bean to a new position.

Assigning properties to the Bean

After registering a Bean and adding it to a form, assign general Bean properties and Bean-specific properties to the Bean.

To assign properties to a Bean:

1. Select the Bean on the form.
2. Press F4 or right click and choose Properties to open the properties sheet for the Bean.
3. Enter values for the properties into the properties sheet. The properties available to set depend on the Bean.
4. Click the Save button.

LESSON 7

Tracing Rule Processing in Versata Logic Suite Applications

Overview

To define transaction logic for Versata Logic Suite applications, you declare business rules. To implement transaction logic, Versata Logic Suite creates a Java implementation file for each business object and copies the files to the Versata Logic Server . The Java implementation files instantiate objects that have attributes corresponding to rules declared in the Transaction Logic Designer. The instantiated objects also support the `save` method, which, depending upon row state, executes an `insert`, `update`, or `delete`, enforcing the restrictions defined in the business rules. As users view and modify data, Java components execute on the Versata Logic Server to enforce the transaction logic.

The Versata Logic Server Console is a Versata Logic Studio-generated Java application that provides seamless, enterprise-wide management of the Versata Logic Server system components and of the business objects hosted in the Versata Logic Server. You can use the Versata Logic Server Console to manage the following major system components: servers, connections, sessions, and transactions. You can manage different aspects of the different types of component.

The Versata Logic Server Console allows you to monitor rule processing to ensure that the business logic executes as you intended. In this section, you use the Versata Logic Server Console to trace system activity and rule execution on the _Demo sample application. For more information about the Versata Logic Server Console, see the *Administrator Guide*.

This section includes the following tasks:

Enabling tracing in the Versata Logic Server Console

You can trace end user activity for different users in the Versata Logic Server Console Tracing Monitor. The Tracing Monitor displays events on a selected user. This option is only available when a run-time application is currently running on the Versata Logic Server.

Note: Before starting this task, make sure that the Versata Logic Server Locator and the Versata Logic Server are running. For instructions on starting the Versata Logic Server Locator and the Versata Logic Server, see "Starting the Versata Logic Server" on page 48.

To trace end user activity:

1. From the Versata Logic Studio, execute the _Demo sample application. If you need instructions, see "Executing the application" on page 70.
2. Start the Versata Logic Server Console, if it is not already running.
   - In the Versata Logic Studio, choose Tools → Versata Logic Server Console, or click the toolbar button.
   - In the Logon to the Versata Logic Server dialog, enter the required information and click OK. You should be able to use the following defaults:
     - Admin login: `sa`
     - Admin password: (leave blank)
     - Versata Logic Server: (leave blank)
3. In the Versata Logic Server Console, expand the Versata Logic Server object, the machine, and the Monitor object.
4. Expand the User Sessions folder and select a user session to trace. (Do not select the Console session, as this represents the Versata Logic Server Console rather than an application.)
5. Enable the Trace user activity check box.
6. (Optional) Click the Filter button and select the types of events that you want to monitor in the trace. The following types are available:
   - Error events occur when an error message displays. An error message prevents the current action from being completed.
   - Information events include events such as transactions beginning.
   - Warning events occur when a warning message displays. A warning message allows the current action to complete.
   - Database events occur when data is retrieved from or saved to the data source.
   - Component events occur when business object code, including rules execution code, is processed on the Versata Logic Server.
7. Now you can observe system activity in the upper window.

Observing system activity

In this task, you observe system activity in the Versata Logic Server Console upper tracing window as you open forms in the _Demo application.

Note: If you need more detail about a particular event, select that event in the upper window and more information will display in the Details window.

At any time, click the Clear button to remove the trace text from the Details window.

At any time, click the Save event log button to save the trace data to a log file. Click the OK button to the confirm message. Note that the file will only contain the trace that has occurred since the last Clear, or since the last Save.

To observe system activity:

1. In the _Demo application, click the Customer image button. The Customers form opens.
2. Review the text in the upper trace window of the Versata Logic Server Console.
   - Notice the symbols associated with different types of events. You can click the Filter button to review the type of event indicated by each symbol.
   - For some types of events, you can select the event in the upper trace window to view details about the event in the lower trace window labeled Details.
3. On the Customers form, click the Details hyperlink to open the Customer form.
4. On the Customer form, click the Orders button on the PaidOrders tab or the Order Detail hyperlink on the Open Orders tab to open the Order form to display unpaid orders.
5. Again, review the text in the trace windows of the Versata Logic Server Console.

Observing rule execution

In this task, you observe rule execution in the Versata Logic Server Console as you update data in the _Demo application.

To observe rule execution:

1. On the Order form in the _Demo application, modify the values for Quantity for the first two line items and click the Save button.
2. Review the entries in the upper trace window of the Versata Logic Server Console. Note that a number of entries are written as the transaction is processed, and that most recent entries are at the top.
3. Scroll down through the trace window so you can see the first entries at the bottom, then review the entries from the bottom up.
   - Notice the symbols associated with different types of events. You can click the Filter button to review the type of event indicated by each symbol.
   - For some types of events, you can select the event in the upper trace window to view details about the event in the lower trace window labeled Details.

Observing server event execution in a Java application

In this section, you review an example of procedural code added to the EMPLOYEES data object in the sample database to extend the logic of declarative rules. This code creates an event that is executed when the specified data object is updated. Next, you use the Versata Logic Server Console trace facility to observe rule execution by the component that has been extended with procedural event code, while running the Server_EventAction_CreateChildren sample application.

You can add your own event code to Versata Logic Server components. Your code can call other services such as email and message-oriented middleware. Your custom code is preserved when you make changes to rules and rebuild components.

Note: Before starting this task, be sure to close the _Demo application. Also, close its application in the Versata Logic Studio.

Reviewing event code in the EMPLOYEES data object

To review server event code:

1. In the Versata Logic Studio Explorer, click the Files tab.
2. Expand the Versata Logic Server and HR folders.
3. Double-click the `EMPLOYEESImpl.java` file. The Code Editor opens.
4. In the Code Editor, click the event mode button (rightmost of the two small buttons in the upper left of code editor screen). In the Events drop-down list, select `afterUpdate`. Review the code.
5. When you are done reviewing the code, close the Code Editor.

Note: This custom code and its related component were already deployed to the Versata Logic Server when you completed a previous section of this tutorial.

Running the Server_EventAction_CreateChildren sample application

To observe server event execution:

1. In the Versata Logic Studio, open the Server_EventAction_CreateChildren application.
2. Execute the Server_EventAction_CreateChildren application. If you need instructions, see “Executing the application” on page 70.

3. In the Versata Logic Server Console, select the user session for the Server_EventAction_CreateChildren application.
4. Enable the Trace user activity check box.
5. (Optional) Click the Filter button and select the events that you want to monitor in the trace.
6. In the Server_EventAction_CreateChildren application, on the Employee form, change the salary of the first listed employee by a small amount, and click the Save button.
7. Review the text in the Versata Logic Server Console trace window.
   - Notice the symbols associated with different types of events. You can click the Filter button to review the type of event indicated by each symbol.
   - For some types of events, you can select the event in the upper trace window to view details about the event in the lower trace window labeled Details.
8. To see the effects of the salary audit event in the application, click the Next button to display data about the second employee, then click the Previous button to redisplay the first employee's data. The repositioning refreshes the list of salary audit rows so you can see the audit entry related to your update.
9. When you are done reviewing trace code, close the Server_EventAction_CreateChildren application. Also, close its application diagram in the Versata Logic Studio. In the Versata Logic Server Console, right-click the user session for Server_EventAction_CreateChildren and click Remove Session From List.