



NXJ Developer

Delivering Rich Internet Applications With NXJ Developer

**A Unify Corporation
Technical White Paper**



TABLE OF CONTENTS

INTRODUCTION.....	4
NXJ DEVELOPER.....	5
VISUAL USER INTERFACE DESIGN.....	6
<i>Visual Layout Construction</i>	7
<i>Rich Set of Controls</i>	8
<i>Data-aware AJAX Controls</i>	8
<i>Custom Controls</i>	8
<i>Information-dense Forms with Tab Controls</i>	8
<i>Property-based Behaviors</i>	9
<i>Comprehensive Display Formatting</i>	10
<i>Advanced Microsoft Office/Document Integration</i>	10
<i>Dynamic Image Management</i>	10
<i>Automated File Transfer</i>	11
<i>Text/Binary Data Type Support</i>	11
<i>Automated Query/Insert/Update/Delete and Navigation Handling</i>	11
<i>Cascading Menu Navigation</i>	12
<i>Form and Field Navigation</i>	12
<i>Customizable Look-and-Feel</i>	12
<i>Automatic Client-side Data Validation</i>	12
<i>Automated Master-Detail Relationships</i>	12
<i>Data Set Caching</i>	13
<i>Flexible Transaction Management</i>	13
<i>Multiple Integrated Data Sources</i>	13
<i>Look Up/Zoom Forms</i>	14
<i>Portal Integration</i>	15
BUSINESS RULE PROGRAMMING	15
<i>Custom Business Rules</i>	15
<i>Event/Trigger-based Programming Model</i>	15
<i>Application-specific Commands</i>	16
<i>Embedded SQL</i>	16
<i>Usage of Stored Procedures and Packages</i>	17
<i>Named Access to Client-side Objects</i>	17
<i>Dynamic List Box Population</i>	18
<i>Dynamic Auto-Stippling</i>	18
<i>Dynamic Look-and-Feel and Behaviors</i>	18
<i>Programmable Server-side Data Validation</i>	18
<i>Programmable Field Auto-Fill</i>	19
<i>Computed Fields</i>	20
<i>Programmable Transaction Management</i>	20
<i>Programmable Application Security</i>	20
<i>Null Value Handling</i>	20
<i>Usage of External Java Classes</i>	20
AUTOMATED DATA SOURCE INTEGRATION.....	20
<i>Wizard-driven</i>	20
<i>All Major Database Systems</i>	21
<i>Importing Web Services</i>	21
WEB SERVICES.....	22
<i>Web Services Development</i>	22
<i>Web Services Consumption</i>	23
CROSS-PLATFORM PORTABILITY	24
<i>Single-Click Compilation/Package/Deploy/Run</i>	24
<i>Source-level Debugging</i>	24
<i>Database Independence</i>	25
<i>Application Server Independence</i>	25
<i>Browser Independence</i>	25
<i>Operating System/Hardware Independence</i>	25
REPOSITORY-BASED DEVELOPMENT AND RE-USE	26

.....

<i>Repository Libraries</i>	26
<i>Reusable Components</i>	26
VERSION CONTROL INTEGRATION	26
<i>Open Source Control Integration</i>	26
<i>Tight Integration with CVS</i>	26
INTERNATIONALIZATION, LOCALIZATION, AND MULTI-LINGUAL SUPPORT	26
<i>Single and Multi-byte Support</i>	26
<i>Localizable</i>	26
<i>Multi-lingual Applications</i>	27
ABOUT UNIFY	28

Introduction

Companies today have external and internal pressures that are placing ever increasing demands on IT and ISV/VAR organizations. These forces seem to be creating a rising corporate appetite for faster, more productive ways of building business applications, in particular Rich Internet Applications (RIA). Rich Internet applications are web applications that have the features and functionality of traditional desktop applications. Corporate mandates are also requiring these new applications to be Web services-based, using a Service-Oriented Architecture (SOA), and reusing existing IT software and information assets. How are developers and development teams supposed to keep up with the demand for Rich Internet Applications that meet industry standards, provide a competitive advantage and need to be delivered “yesterday?”

Industry analysts and the media are writing today about Web 2.0, the next generation of Internet-based services, including social networking sites, wikis and communication tools that let people collaborate and share information online in ways previously unavailable. This phenomenon is happening in consumer website today such as Yahoo mail, myspace and Google maps. It has not crossed over into enterprise business applications. Building collaborative, Rich Internet Applications that leverage the power of SOA, integrate data from multiple sources and deliver a rich user interface that meets the demands of the user and corporate mandates remains arduous, particularly for organizations without an army of developers.

At Unify, we saw this emergence of Rich Internet Applications and have combined it with our heritage in building productive and reliable development tools and databases specifically designed for business application developers. The result is a rapid application development solution that:

- Meets corporate standards for SOA, Web services-based applications
- Delivers collaborative, intuitive, process and services-based applications
- Dramatically reduces total coding and testing time and effort
- Contains pre-built functionality to accelerate any coding that may be required
- Provides a rich User Interface design environment
- Enables business unit developers to start and complete an enterprise application development project
- Enables complex applications to be built in months with low maintenance requirements post deployment

Unify delivers a solution which meets these requirements – NXJ Developer.

NXJ Developer

What sets NXJ Developer apart is its combination of an intuitive visual design environment, flexible SOA programming and system orchestration for production deployment of high performance Web Application that embraces the power and openness of Java. The combination of AJAX-rich client technology, Web Services, workflow, reporting and portal integration make NXJ Developer the ideal solution for Rich Internet Applications.

NXJ Developer is the key to leveraging the power of production-ready AJAX-based Rich Internet Applications. NXJ Developer enables business application developers to quickly design, code, deploy, and debug AJAX and Service Oriented Architecture (SOA)-based applications in a single unified environment. It uses an intuitive drag-and-drop interface to create views and forms, and to populate them with data in real time.

NXJ Developer isolates the complexity of Rich Internet Applications and brings together all the capabilities necessary for traditional business application developers to quickly ramp up the skills ladder and deliver more business value and lower transaction costs through web services-based applications. Design is straight-forward that developers familiar with other traditional development tools can easily use the NXJ Developer right out of the box.

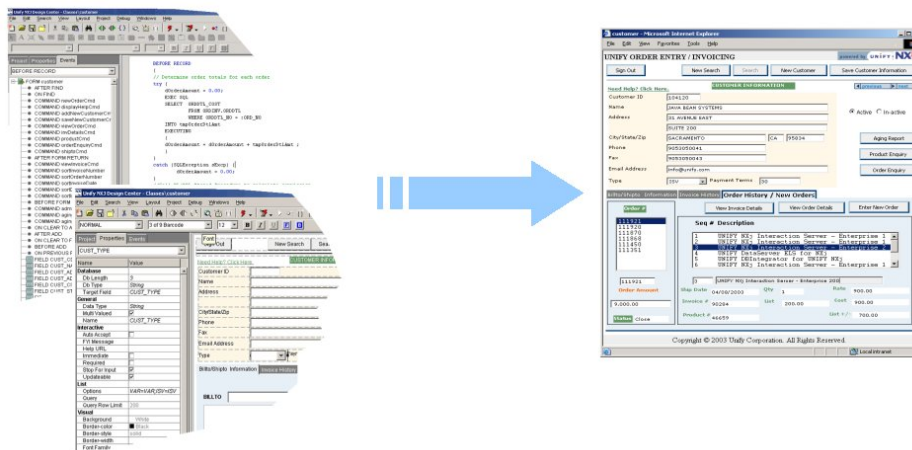


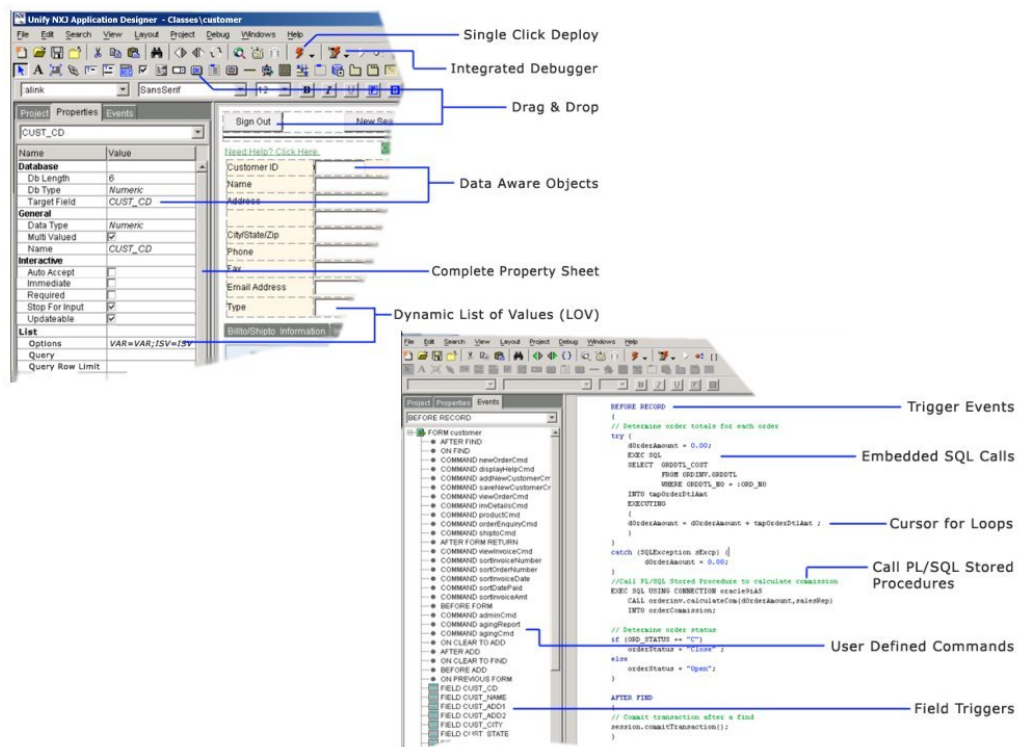
Figure 1 – NXJ Developer visual design enables rapid completion of Rich Internet Applications

Visual User Interface Design

Most traditional IDEs are general purpose in nature and have no inherent knowledge of the target application that is being built. In addition, they have no knowledge of the services and components that the developer is using. At best, they can render graphical classes within a visual editor, and display generic lists of properties and methods.

NXJ Developer works differently. It is fundamentally designed to develop Rich Internet Applications that rely on transactional and non-transactional data derived or “mashed-up” from various services, legacy systems and databases. Based on this application domain focus and specialized knowledge, NXJ Developer provides a variety of application accelerators that dramatically increase productivity and lowers on-going application maintenance costs.

Figure 2 - NXJ Developer Capabilities



NXJ Developer includes the following capabilities:

- Visual AJAX User Interface Design
- Web Services provider, consumer and governance
- Single Sign On (SSO) LDAP, Active Directory or custom database
- Repository-based team development
- Reusable form components and inheritance
- Visual Business Rules Programming
- Automated Data Source Integration

- Single-click compile, deploy, run, and debug
- Cross-platform portability
- Source code control integration
- Localization, internationalization, and multi-lingual applications
- Web based reporting (optional module)
- Workflow / Business Process Management (BPM)
- JSR 168 Portlet creation
- Automatic deployment of forms, portlets, web services, reports and business processes

Visual Layout Construction

NXJ Developer includes a visual layout tool for specifying the look and feel for AJAX user interfaces. Unlike tag-based editors, NXJ Developer allows developers to perform visual construction without the need to know tag-based programming. NXJ Developer provided AJAX components are automatically incorporated into the application design and commercial third party AJAX components can also easily be added and extended within the development environment.

Figure 3 - NXJ Developer provides AJAX visual construction of interfaces with a rich set of controls

The screenshot shows a web browser window titled 'customer - Microsoft Internet Explorer' with the address 'http://localhost:8089/IBossCenter/packages/ordinv/login'. The page is titled 'UNIFY ORDER ENTRY / INVOICING' and is powered by UNIFY NXJ. It features a navigation bar with buttons for 'Sign Out', 'New Search', 'Search', 'New Customer', and 'Save Customer Information'. Below this is a 'CUSTOMER INFORMATION' section with a form containing fields for Customer ID (104120), Name (JAVA BEAN SYSTEMS), Address (31 AVENUE EAST, SUITE 200), City/State/Zip (SACRAMENTO, CA, 95834), Phone (9053050041), Fax (9053050043), Email Address (info@unify.com), and Type (ISV). There are also radio buttons for 'Active' and 'In-active', and buttons for 'Aging Report', 'Product Enquiry', and 'Order Enquiry'. Below the customer information is an 'Invoice History' section with a table showing a list of invoices. The table has columns for Invoice #, Order #, Invoice Date, Invoice Amount, Date Paid, and Amount Paid. The first row is highlighted in blue. Below the table is a summary row for the selected invoice (90014) and a status bar showing 'Status Paid', 'Amount Outstanding 0.00', 'Due Date 02/28/2003', and 'Overdue'. The footer of the page reads 'Copyright © 2003 Unify Corporation. All Rights Reserved.' and 'Local intranet'.

Invoice #	Order #	Invoice Date	Invoice Amount	Date Paid	Amount Paid
90014	111450	01/29/2003	75,000.00	03/03/2003	75,000.00
90068	111351	02/13/2003	13,250.00	04/11/2003	13,250.00
90254	111868	03/31/2003	30,000.00		0.00
90255	111870	03/31/2003	3,000.00		0.00
90283	111920	04/08/2003	25,000.00		0.00
90284	111921	04/08/2003	9,000.00		0.00

Rich Set of Controls

NXJ Developer includes a rich set of pre-built AJAX controls for use in web interfaces. The controls provided include:

- Static and dynamic tree and menus
- Dynamic spreadsheet grids
- Date picker, label, styled text box, link, text field, text area, dynamic text field, check box, list box, dropdown list box, button
- Radio group, image button, line, image, table, repeating area, group box, data view, tab set, inline frame and form

Virtually all controls can be programmatically changed dynamically in deployment.

Data-aware AJAX Controls

NXJ Developer AJAX Controls are a set of advanced, data-aware controls that provide automatic connectivity to data source elements such as a server-side database column, SOA Services or BPM process activity operand without the need for additional programming.

Custom Controls

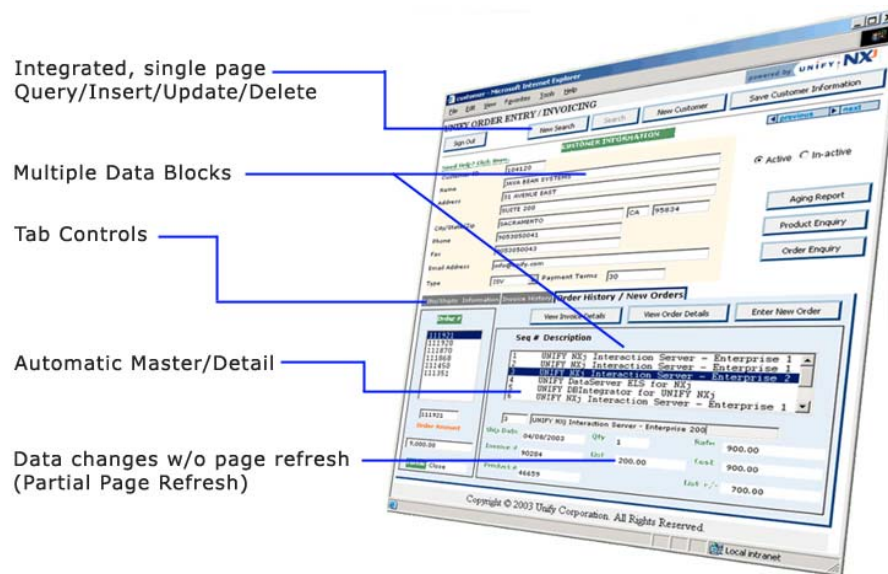
Developers can extend the NXJ control set through the creation of NXJ Custom Controls. These controls are a combination of Java Swing (Visual representation) and JavaScript for functional operation.

Information-dense Forms with Tab Controls

Within its control set, NXJ Developer provides a native AJAX Tab Set control. By using the Tab Set control, developers can produce applications which are both information-dense and easy to navigate as well as highly responsive.

Unlike typical image-based tab structures which are actually multiple HTML pages, NXJ tab controls look and operate like native tab controls, where all pages of a tab set are contained on the same web page. Clicking on different pages of a tab set does not require additional interactions with the web server, thus providing a greatly enhanced end-user experience. Performance of the Tab Control is enhanced with AJAX streamed data so that non-visible tabs are populated after control is returned to the user.

Figure 4 - The information-rich presentation of a typical NXJ application



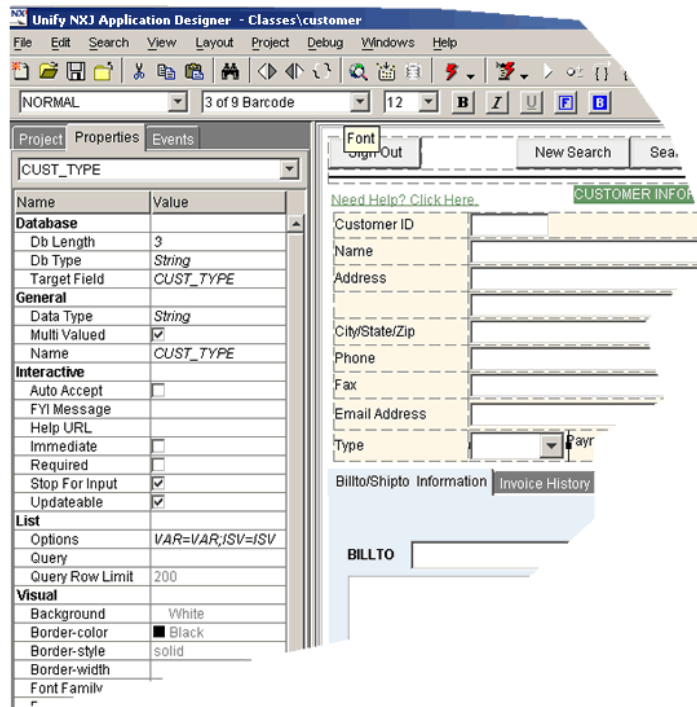
Property-based Behaviors

NXJ Developer provides a rich set of property sheets for each AJAX control that is available. Through this powerful mechanism, a large amount of application behavior can be specified, thus reducing custom coding and maintenance costs, increasing productivity, and increasing overall application consistency and quality.

Examples of control properties include:

- User navigation: updatable, read-only, tab stops, case conversion, formatting, and password
- Display formats
- Data element connectivity
- Visual attributes (color, style, font, label etc.)
- Component visibility
- Custom commands (click, double-click focus etc.)
- Security (visible, update, find, delete, add, focus etc.)
- Custom validation rules
- Bubble help

Figure 5 - Property-based programming within the Designer



Comprehensive Display Formatting

NXJ Developer provides a wide range of built-in display formatting options for data types including support for various locale-specific settings.

Advanced Microsoft Office/Document Integration

NXJ Developer applications provide the ability to automatically store and retrieve documents such as Microsoft Word and Excel as well as other document types such as Adobe PDF from the application server. This allows the user to retrieve, update, and save documents such as a spreadsheet directly from within their web application.

Documents are server-side resident and can be stored in either a file system or within a database as BINARY objects.

Dynamic Image Management

NXJ Developer applications provide the ability to automatically and dynamically utilize any Image type residing as binary data types within a database.

For example an application may have an image of each employee stored in the database. With the NXJ Developer, a user can simply create an Image object on a form and associate it with the "employee picture" image column in the database or web service. At runtime, the images will automatically be retrieved and displayed.

Dynamic inserting and updating of image data is also provided with no coding required by the developer.

Automated File Transfer

Applications built with NXJ Developer provide the ability to perform bi-directional file transfers between the browser and the application server. NXJ provides a file control which automates the management of file upload and download, with advanced features available through dynamic server-side programming.

Text/Binary Data Type Support

NXJ Developer provides direct support for TEXT and BINARY (CLOB/BLOB) data types. This includes retrieving TEXT or BINARY values from a database or Web services and associating them with objects on an NXJ form. For example, an application can retrieve TEXT data from a database table and present it directly in a TextArea on a form. An application can also select a Binary image from any data source and assign it to an image object on an NXJ form.

Automated Query/Insert/Update/Delete and Navigation Handling

NXJ Developer provides both development and deployment handling of standard data maintenance operations: Query-by-Form, Insert, Update, and Delete.

Operations are transactional. Query-by-Forms includes support for wild cards, ranges, sets of values, and exact value matches.

Data set caching and navigation is automatically provided. Controls include built-in navigation bars which can be replaced or customized as the need arises.

Figure 6 - NXJ applications provide full query-by-form capabilities automatically

**Show all
Invoices > \$20,000
Paid between 1/1/03 and 5/31/03**

Invoice #	Order #	Invoice Date	Invoice Amount	Date Paid	Amount Paid
90014	111450	01/29/2003	75,000.00	03/03/2003	75,000.00
90068	111351	02/13/2003	12,250.00	04/11/2003	12,250.00
90254	111868	03/31/2003	30,000.00		0.00
90255	111870	03/31/2003	3,000.00		0.00
90283	111920	04/08/2003	25,000.00		0.00
90284	111921	04/08/2003	9,000.00		0.00

Invoice #	Order #	Invoice Date	Invoice Amount	Date Paid	Amount Paid
90014	111450	01/29/2003	75,000.00	03/03/2003	75,000.00

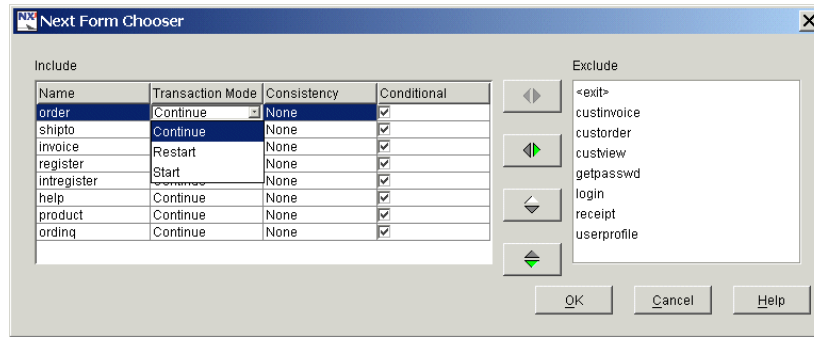
Cascading Menu Navigation

NXJ Developer provides a simple and intuitive mechanism for designing cascading menus. Developers can add system- or user-defined commands to any node of a menu, or they can construct the menus dynamically at runtime based on business or security logic. These menus can utilize the built-in skins or developers can customize look-and-feel via standard Cascading Style Sheets.

Form and Field Navigation

NXJ Developer includes control properties for both field to field and form to form navigation. Navigation can also be changed programmatically on the fly based on particular user inputs or application state.

Figure 7 - NXJ Developer provides both field and form navigation specifications



Customizable Look-and-Feel

NXJ Developer provides the ability to associate multiple look-and-feels defined with a project. Look-and-feels provide a broad set of customization including different images, colors, fonts, styles (any Cascading Style Sheet "CSS" settings). This capability allows a single application to be "branded" differently for depending on customer type. This branding (changing the colors, images, fonts etc.) can be done at compile-time or with changes as the application is deployed. NXJ Developer provides this ability based on standard Cascading Style Sheets (CSS) from files as well as utilizing URL-based style sheets.

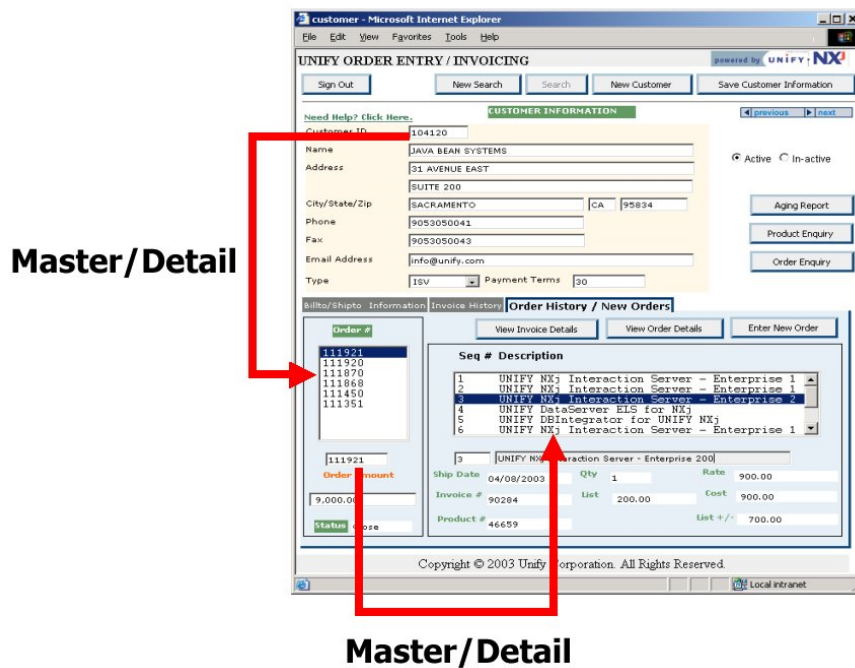
Automatic Client-side Data Validation

Since client-facing controls all have an associated data type, the controls provide automatic data input validation, thus eliminating tedious JavaScript coding.

Automated Master-Detail Relationships

Master-detail relationships are a common occurrence with many applications. NXJ Developer provides automatic AJAX driven master-detail specification as well as runtime implementation of the relationships. Multiple master-detail relationships can be specified on a single form and to any level of depth.

Figure 8 - NXJ Developer provides automated master/detail relationships



Data Set Caching

NXJ Developer provides for both client-side and server-side caching of data sets thereby optimizing network traffic, response time, and load on critical server resources.

Client-side caching is automatically performed depending on the type of control used. Server-side caching is performed either completely in-memory or optionally cached on disk to allow for the handling of large data sets.

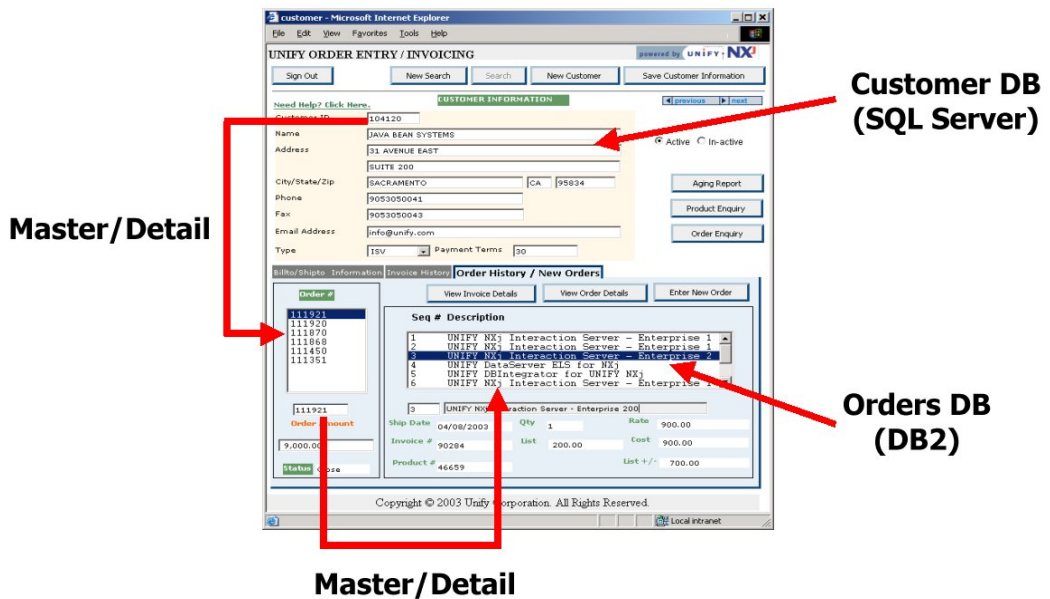
Flexible Transaction Management

NXJ Developer applications often are transactional. Transaction control can be specified both through control properties as well as programmatically through the easy API. In addition to auto-commit, NXJ Developer allows for developer-specified control over the length of a transaction, thus allowing a transaction to be committed or rolled back at the exact time it should be.

Multiple Integrated Data Sources

A single page of an NXJ Developer application can include data from any number of data sources. NXJ provides a sophisticated data-aware control called a Data View. A Data View provides a presentation and data management model for a single data source, where a data source can be a relational or non-relational source. A single NXJ form can include one or more data views, formed in any type of hierarchical relationship within the form. For example, a single page can display customer information from an Oracle database with invoice numbers from a SQL Server database and open customer inquiries from a web service or CICS transaction.

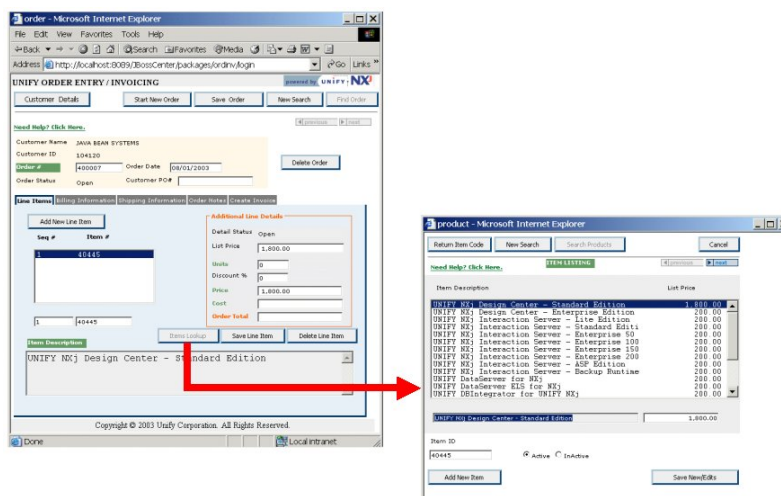
Figure 9 - NXJ applications can reference multiple data sources, even in a single page



Look Up/Zoom Forms

NXJ Developer provides a unique and powerful data-lookup feature called a Zoom form (sometimes referred to as a Lookup Form). Zoom form functionality provides an alternative to list boxes and other controls where the number of choices is too cumbersome for the end-user to use. Zoom forms allow the end user to dynamically search (Query-by-Forms), select a particular data record, and then 'return' that data back into the current form, automatically filling in the associated input fields in the receiving form. It's like lookup-copy-and-paste between pages of an application.

Figure 10 - NXJ applications provide a powerful dynamic look-up capability called a Zoom form.



Portal Integration

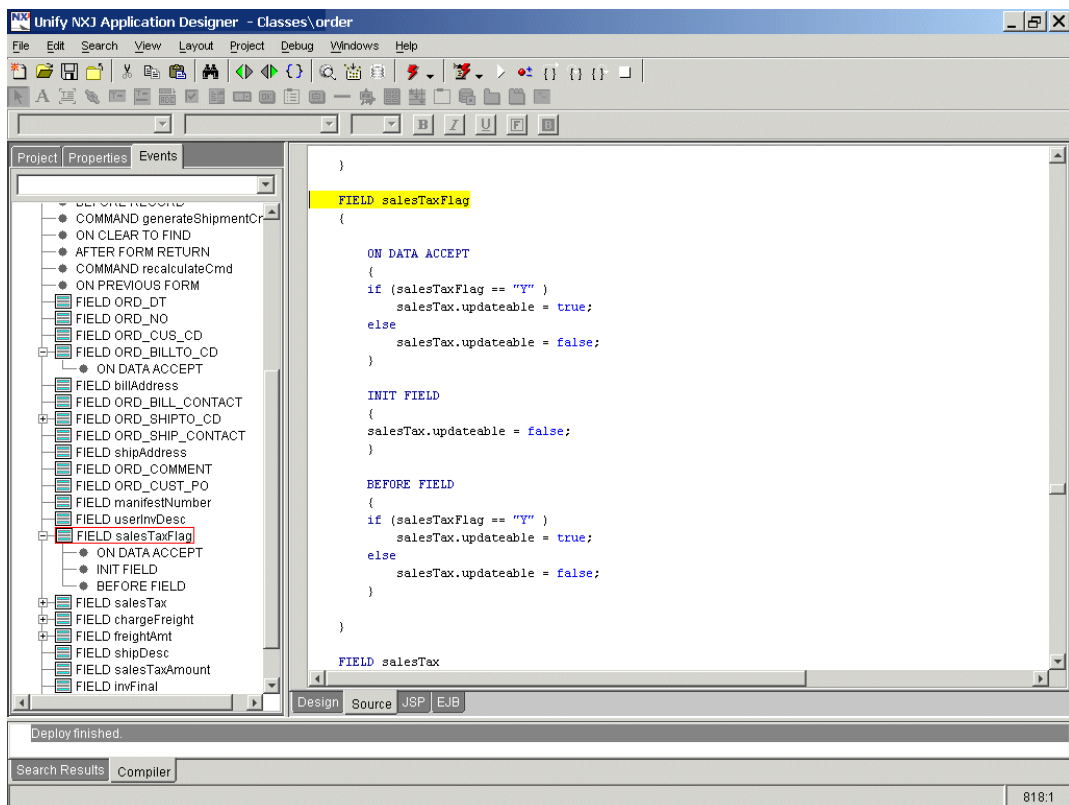
NXJ Developer provides for the automatic creation of JSR 168 portlet interfaces to leading portal servers. NXJ applications can be automatically deployed through portals including IBM WebSphere Portal, Oracle Portal and WebLogic Portal.

Business Rule Programming

Custom Business Rules

NXJ Developer provides a feature rich and easy to use environment that enables business developers to quickly extend the Rich Internet Applications. Business logic is extended and enhanced with auto-complete, color coding and programming accelerators. To simplify and improve productivity, NXJ Developer provides accelerators that provide a familiar development paradigm and enable breakthrough productivity in a Java/J2EE application development environment.

Figure 11 - Programming NXJ Developer applications is performed using Java



Event/Trigger-based Programming Model

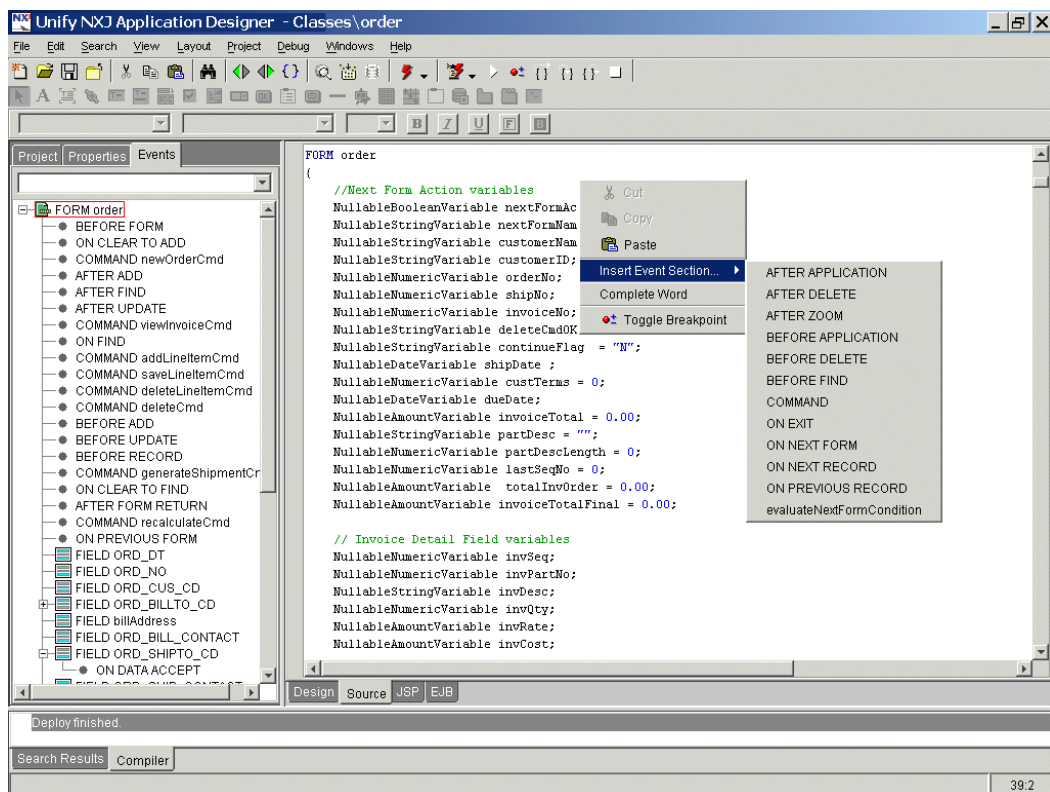
NXJ Developer provides an event or trigger-based programming model, which is similar to the programming style provided by many popular client/server development tools such as Lotus Notes, Team Developer, Oracle Forms, PowerBuilder, and Visual Basic.

Triggers are available for nearly all user interface controls as well as data source operations, and provide a convenient mechanism for specifying server-side actions with their associated client-side and server-side events. The code specified for these events ultimately resides within the application server.

Examples of trigger types include:

- BEFORE FORM, ON NEXT FORM, BEFORE UPDATE, AFTER UPDATE
- BEFORE FIELD, AFTER FIELD, ON DATA ACCEPT
- BEFORE UPDATE OPERANDS, ON CANCEL ACTIVITY, AFTER COMPLETE ACTIVITY

Figure 12 - Unify NXJ Developer provides a trigger-based programming model



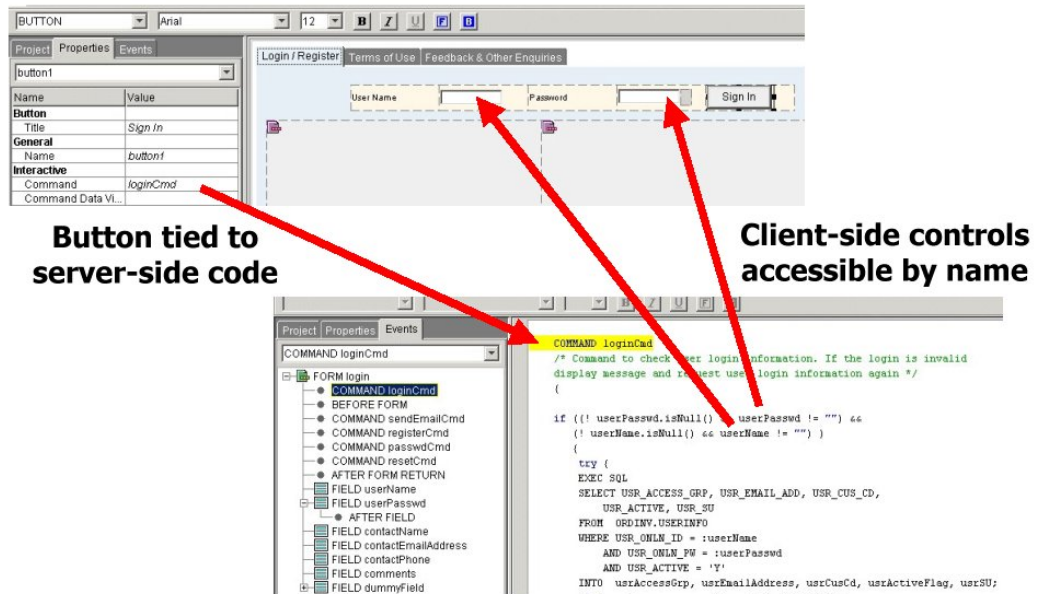
Application-specific Commands

NXJ Developer provides a programming feature called a Command which ties together application-specific operations with user-initiated events. Similar to events, a command is server-based code which can be tied to interface controls such as a button or function key. NXJ provides several built-in commands, but also allows for developer-defined commands.

Embedded SQL

As an alternative to coding several hundred lines of JDBC code for database operations, NXJ Developer provides a facility to embed SQL-compliant statements directly within Java. Statements such as SELECT, INSERT, UPDATE, and

Figure 14 - Server-side access to client-side controls is easily performed by simply referring to a control by name



Dynamic List Box Population

NXJ Developer provides the ability to specify automatic population of list box (list-of-value) controls. A developer can simply specify a SQL statement to retrieve the values for a list box control and AJAX will automatically populate the control dynamically at runtime.

Dynamic Auto-Stippling

NXJ Developer through AJAX provides the ability to automatically disable and visually stipple button and image controls. This ability can be directly tied to security constraints and available application-specific commands, thereby controlling and directing application access based on the user's privileges and application context.

Dynamic Look-and-Feel and Behaviors

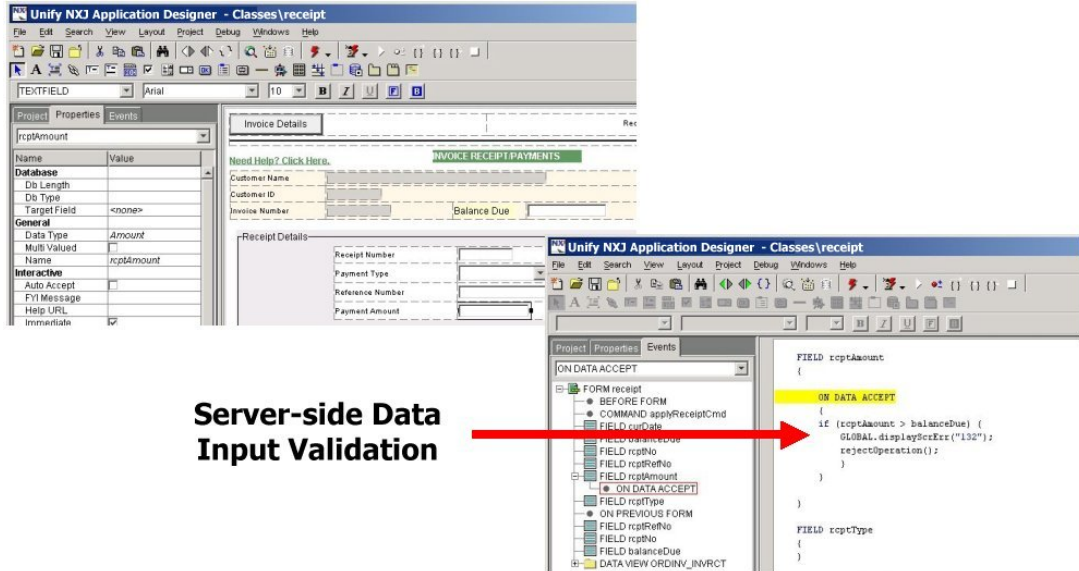
The properties of labels, buttons, fields and other objects can be programmatically modified dynamically at runtime. For example, the color of currency can dynamically be changed to red when it is a negative amount or a field can be made invisible based on the access rights of the current user. This functionality is propagated throughout all of the spreadsheet grid and repeating area AJAX components.

Programmable Server-side Data Validation

With the combination of event-driven real-time events and AJAX technology, NXJ Developer provides the developer full server-side data validation. Validation operations can be either executed immediately via AJAX or batched together into a

single operation. Server-side validation code can perform operations such as real-time lookup of data within a DBMS, web service, or legacy transaction.

Figure 15 - Even the most sophisticated data validation tests can be performed through server-side real-time validation

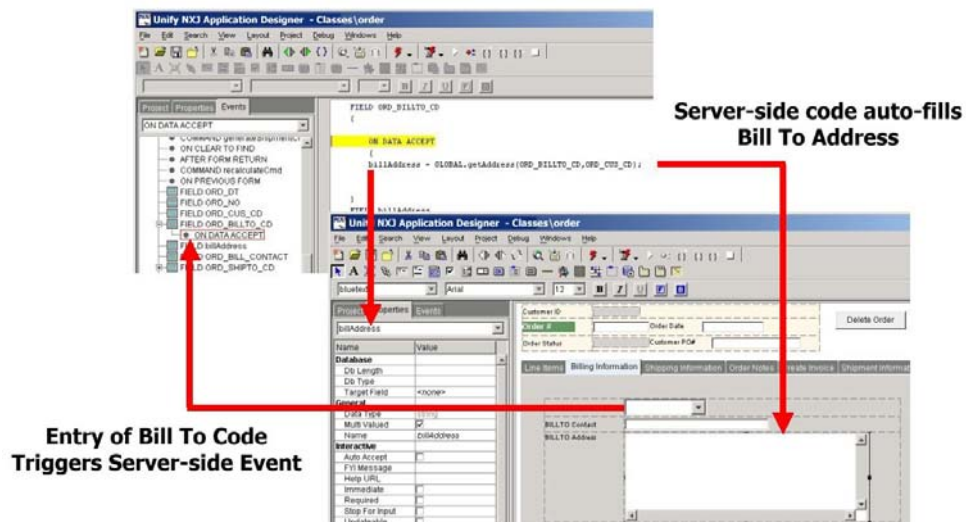


Server-side Data Input Validation

Programmable Field Auto-Fill

By utilizing event-driven server-side rules with AJAX, a developer can provide sophisticated data entry operations such as automatically filling in data input fields based on the current data entry field or other application-specific states. For example, by entering a customer name, related customer information can automatically be populated in the associated fields on the form (i.e. bill to, ship to, customer id, etc.).

Figure 16 - Programmable auto-fill enriches the end-user experience and increases data input speed



Entry of Bill To Code Triggers Server-side Event

Server-side code auto-fills Bill To Address

Computed Fields

In many cases, the value for a particular field is the result of a computation based on values in other displayed and non-displayed fields/data elements. NXJ Developer allows a developer to define and control computed fields through server-side code. For example, an extended price for an order line item can be defined simply by assigning the “extended price field” the value of “SKU unit price * SKU quantity”.

Using AJAX technology, the computed value is automatically propagated to the client, thereby ensuring that the user sees the correct, up-to-date value. AJAX handles all of the communication of the data values between the browser client and the server using standard HTTP/HTTPS protocols.

Programmable Transaction Management

The NXJ Developer provides complete and flexible transaction action, both at the property level as well as programmatically through business rules. Access to the current transaction as well as data source connection information is readily available via a rich API.

Programmable Application Security

NXJ Developer provides for the ability to build simple to advanced application-level security. Built-in operations limit specific users and groups to standard operations (select, insert, update, delete, user-defined commands). Further limitations can be handled programmatically, thus controlling specific access and operations on client-side data elements as well as application operations. Client-side fields can even be made invisible on the fly to prevent viewing by unauthorized users, but without the necessity to develop multiple forms for different classes of user authorizations. Full support is provided for LDAP, Active Directory, NTLN and custom database tables.

Null Value Handling

Null values are a way of life with relational database systems. However, Java does not make it easy on developers to support the null value notion. NXJ Developer makes null value handling easy by providing extended data types which can take on a value of NULL. This eliminates the need to handle null values individually for each possible data element, thus decreasing coding efforts and increasing quality.

Usage of External Java Classes

Business rules within NXJ Developer applications are specified in Java and are thus free to utilize all features provided by Java and the J2EE platform. Externally developed or third party classes can simply be imported through standard Java import statements.

Automated Data Source Integration

Wizard-driven

NXJ Developer includes wizard-driven definitions for all data sources with data source-specific information provided as required.

All Major Database Systems

NXJ Developer supports all major database systems including Oracle, IBM DB2, Microsoft SQL Server, Sybase, SQLBase, MySQL, Informix, and the Unify DataServer family. Rich Internet Applications can have multiple data source connections that are managed by the application server connection pooling system. This multiple connection support enables applications to act and react to data from multiple database systems from the same AJAX client.

Figure 17 - Data source specifications for NXJ data source

The screenshot shows a dialog box titled "Database Connection insord". It contains the following fields and options:

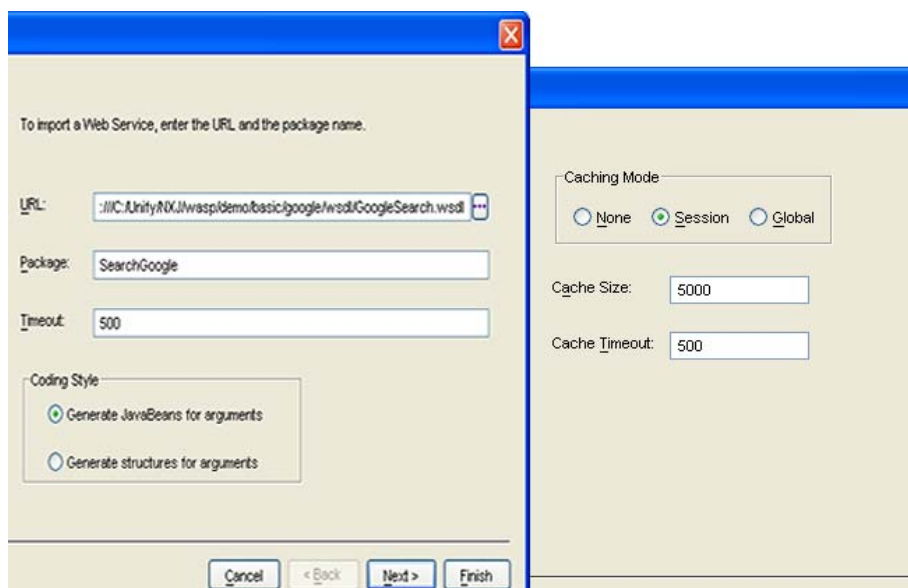
- Type: Oracle
- Jar/Zip File: C:\Unify\Spring\lib\jdbcDrivers\classes12.zip
- User Name: ins
- Password: ***
- Runtime DataSource Mapping: insorc
- Make this the default connection?
- Oracle Properties:
 - Host: dmglat
 - Port: 1521
 - SID: NXJORA

Buttons at the bottom: OK, Test, Cancel.

Importing Web Services

NXJ Developer enables external Web services to be incorporated into the AJAX client as a data source. Discovering and creating the required client connectivity is as simple as entering the URL for the published WSDL. NXJ Developer will then automatically generate the client helper classes for instantiating the services to enable any AJAX client to connect to these services. The imported services can be easily cached with a configurable refresh schedule. This comprehensive support allows for easy “mashup” application to be constructed of multiple Web services.

Figure 18 - NXJ Developer importing Web Services



Web Services

NXJ Developer provides a powerful and intuitive Web services development environment that enables departmental developers to create, publish, consume and orchestrate services.

Web Services Development

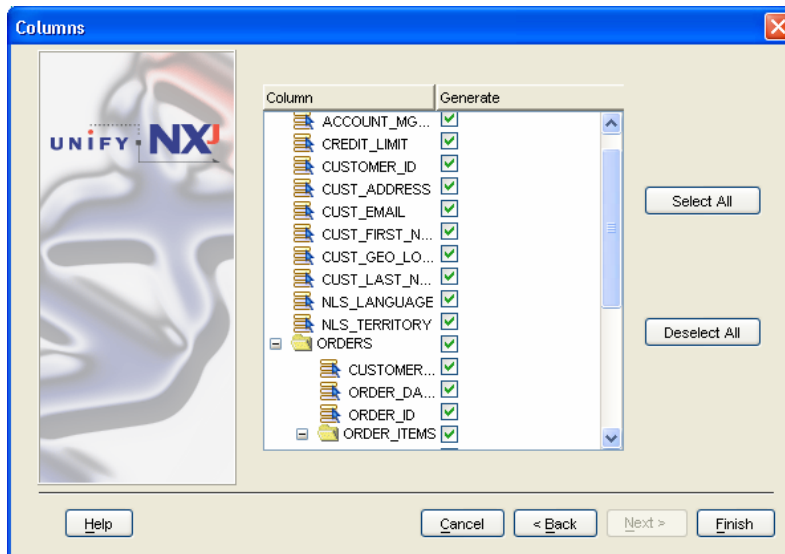
NXJ Developer provides automated wizards to create Business Web services or Data Web services. Business Web services are a convenient method to create and encapsulate services that combines business processes and multiple data sources and augments the aggregated data with business logic. A common example of this type of Business Service is a service that connects to a CRM system to lookup customer data then calls to an SAP system to gather manufacturing data then calls a database to lookup specific order data. All of the accumulated data is then packaged into a cohesive data structure with the appropriate methods for finding, updating, inserting and deleting records. These powerful Business Services include the ability to utilize embedded SQL and standard third party classes. These services are automatically deployed as local Java classes and as Web services.

Data Services produced in NXJ are patterned after JSR-235. A Data Service is a special type of service which conforms to a well-known interface. These Data Services can represent a hierarchical data structure with the appropriate methods for finding, updating, inserting and deleting data. For example, you can point to a relational database table of customers and the resulting NXJ Data Service will include the hierarchical view of the customer with the related address, orders and items (see Figure 1). The Data Service interface can be used by NXJ and other environments (Java, .Net, or otherwise) to rapidly build automated user interfaces

that interact with the underlying data service. Data Services are automatically deployed as local classes and as Web services.

All Web services created with NXJ Developer take advantage of the powerful features of the NXJ environment; including Embedded SQL, nullable variables, transaction semantics, SSO security (Governance) and automated WSDL generation. When the NXJ Developer deployment package is constructed, the Web services are packaged and automatically deployed into the Web service container.

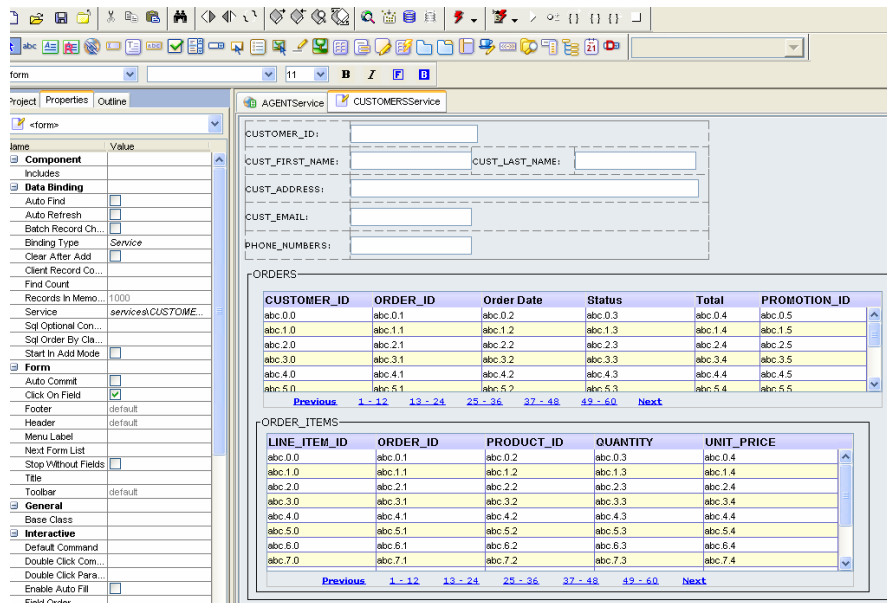
Figure 19 - Data Service with hierarchical data



Web Services Consumption

NXJ Developer provides very intuitive and innovative ways to consume and otherwise orchestrate Web services. One of the most powerful features of NXJ Developer is its ability to automatically generate and bind AJAX clients to Web services. The AJAX clients can be modified and extended to provide the desired interaction with the Web service. The underlying “plumbing” is all handled by NXJ leaving the developer free to focus on the business solution.

Figure 20 - NXJ Developer generated AJAX Client for a Web Service



Cross-platform Portability

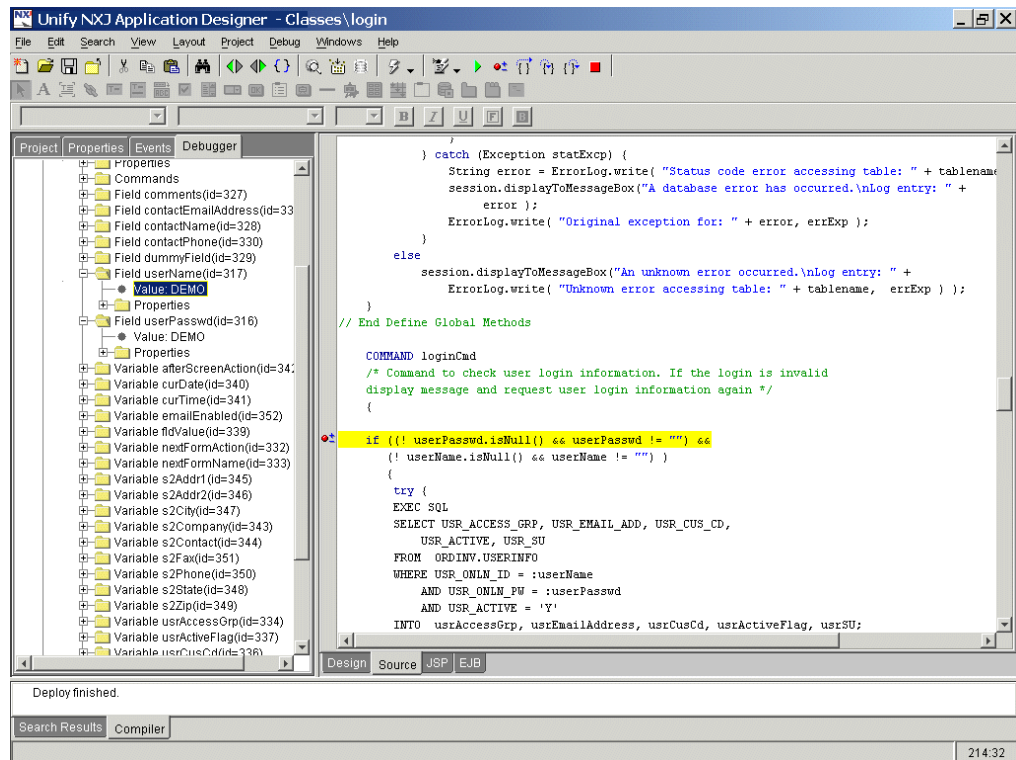
Single-Click Compilation/Package/Deploy/Run

NXJ Developer provides a convenient single action command to compile, package, and run an application. NXJ Developer automatically packages the various application components into universally deployable Web Application. This Web Application, along with application-server specific deployment descriptors and files, are then deployed to the target Application Server.

Source-level Debugging

For application testing and debugging, the NXJ Developer provides an integrated source-level debugger. Debugging follows a typical debugger paradigm and provides business-rule source level debugging – the developer debugs at the specification level, not the code generation level.

Figure 21 - NXJ Developer provides source-level debugging.



Database Independence

NXJ Developer-built Rich Internet Applications are database independent with the exception of any developer-specified vendor-dependent operations. Thus, applications are easily configured for any DBMS vendor type.

Application Server Independence

NXJ Developer-built Rich Internet Applications are application server independent. NXJ goes beyond nearly every application development tool, automatically generating all necessary deployment options and files required across each server type. Switching between Oracle10GAS, JBoss, IBM WebSphere or BEA WebLogic is as simple as re-deploying the application to a different server.

Browser Independence

NXJ Developer takes care of all browser and browser-version dependencies, thus freeing the developer from browser-specific programming. Support for Microsoft Internet Explorer 6.x and above as well as Firefox is provided.

Operating System/Hardware Independence

Since NXJ Developer Rich Internet Applications are Java-based, all NXJ applications are completely portable across popular operating systems and hardware platforms. All that is required is the appropriate Java Virtual Machine and J2EE server, and NXJ Developer Rich Internet Applications can be deployed immediately.

Repository-based Development and Re-Use

Repository Libraries

NXJ application components can be stored in shared repository libraries thereby enabling component reuse across applications, projects, teams, and organizations. Any distinct component or resource within a library can be easily added and configured into any NXJ form or container. Repository libraries can provide standardized look-and-feel property files, presentation components, service components, process components, and data source connections. The use of repository libraries provides an easy to manage and use mechanism for implementing general reuse principals across NXJ projects.

Reusable Components

Reusable components enable developers to easily construct new applications composed of new and existing forms and form components. NXJ's modular design environment enables solutions for critical challenges while maximizing the use of existing assets while at the same time minimizing the cost of future enhancements and maintenance. NXJ provides the capability to define both forms and form components that can be reused in multiple applications throughout multiple projects. These reusable components can be sub-classed or extended to provide a quicker and more standard development and deployment environment.

Version Control Integration

Open Source Control Integration

NXJ Developer fully supports team development and provides an easy to use, open integration with all check-in/check-out based source control systems.

Tight Integration with CVS

NXJ Developer also provides tight integration with CVS, the popular open source version control system. Full feature support and flexibility is provided through sophisticated dialogs and functionality.

Internationalization, Localization, and Multi-Lingual Support

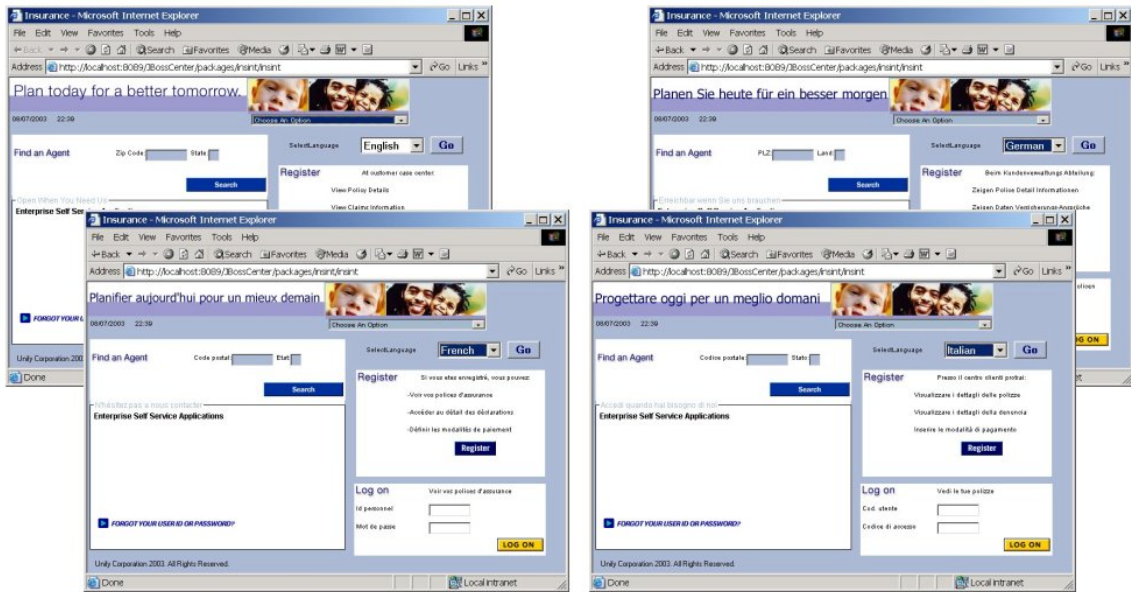
Single and Multi-byte Support

NXJ Developer applications fully support US, European, and Asian languages including multi-byte character sets.

Localizable

NXJ Developer applications are localizable including all application-specific and system-wide labels and text. NXJ Developer provides a powerful tag-based feature which allows application specification to be done in a single language and then subsequently translated to other languages based on an external tag map file. Through this feature, localization can be performed independent of the application development team and the application itself.

Figure 22 - Unify NXJ Forms applications can be multi-lingual and are easily localizable.



Multi-lingual Applications

NXJ Developer goes one step beyond localization by letting a single application source be run in different languages concurrently. Thus an NXJ application deployed on a single server can support English, French, Italian, Japanese and German all simultaneously. A single application instance can even switch between languages dynamically.

About Unify

Unify provides software development and database products that deliver a broad set of capabilities for rapidly developing business applications that automate business processes, deliver collaborative information and integration existing information systems. Through its expertise and market leading technologies, Unify helps customers and partners deliver solutions that drive efficiency, apply governance, lower IT costs and increase customer service.

ASI, Cast & Crew Entertainment, Inc., EMC Documentum, Fox Racing, Pinnacol Assurance, Pioneer Natural Resources and TravelCenters of America are among Unify's more than 100 NXJ customers.

To download an evaluation copy of NXJ Developer, email Unify at info@unify.com or visit us at www.unify.com.

Unify Corporation
Worldwide Headquarters:
2101 Arena Blvd.
Sacramento, CA 95834
USA

Phone: 1.916.928.6400
Toll Free: 1.800.468.6439
Fax: 1.916.928.6404
info@unify.com

United Kingdom:
Phone: + 44.178.4487940
E-Mail: infouk@unify.com

France:
Phone: +33.1.413.82299
E-Mail: infofr@unify.com

www.unify.com

COPYRIGHT © 2006. UNIFY CORPORATION. All rights reserved.

Unify, Unify NXJ and the Unify logo are registered trademarks of Unify Corporation. Java and J2EE are the trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All other company or product names are trademarks of their respective owners.

