

in2j

Moving from Oracle Forms to Java the easy way



Quintessence Systems Limited

www.in2j.com

1. Introduction

Oracle Forms was developed over 25 years ago as a tool for the rapid development of user interface applications accessing the Oracle database. It was hugely popular and was adopted by the majority of Oracle's 160,000 customers. The sophisticated way in which forms can be created with multiple block and with items spread over multiple canvases gives an elegant solution to the style of complex transactions it is designed to handle.

Most Forms applications are used by skilled administrators and consist of hundreds or thousands of forms each of which perform a particular type of complex database transaction. A typical form will have ten to thirty blocks and a similar number of canvases. A typical canvas will have 20 items (fields) half of which will repeat.

Forms applications are rarely designed and as a whole, but are instead built over many years based on detailed requests by users. They may be documented but in general they are not. Forms has no inheritance structures so new forms tend to be modified copies of existing forms, this tends to make systems huge. But as each form is freestanding, the system is relatively easy to maintain.

In early 2008 Forms 6 became unsupported. The status of Forms 10g is somewhat ambiguous. On the one hand, Oracle have no plans to drop support for Forms 10g (see <http://www.oracle.com/technology/products/forms/pdf/10g/ToolsSOD.pdf>). On the other, Oracle now recommends that all new systems are written in JDeveloper using its Java based Oracle Application Development Framework (ADF).

However it is rare that a 'new system' is built. Instead old systems are extended as a customer's business needs evolve. Many Forms systems are over twenty years old, and a team of ten people may have been enhancing that system over the period and seen it go from 1 form to 500 forms, a huge investment in human resources and money.

Forms users also report some considerable difficulties in upgrading their applications between different Forms versions. Even after a considerable investment in upgrading, for example from 6i to 10g, users may find that their forms still do not operate consistently with the original version.

Given the above Forms users may understandably feel uncertain as to the future of Forms support, and the long term security of their Forms investment.

Application Server does offer interoperability so Forms applications can call J2EE applications and vice versa. However with Forms 10g running in an applet and J2EE as a separate web page, it is hard to see how forms can be individually replaced with Java versions over time.

The Forms user is therefore left with a choice: to stay with Forms; or move to Java. If they stay with Forms they have the stress of watching their platform become obsolete while their best staff leave to work with newer technology. If they move to Java thousands of forms must all be re-written at once, a daunting task indeed.

2. The in2j Approach

The in2j approach is to move the system automatically from Forms to Java and XML. No knowledge of the system is required other than to:

- locate the source files for the forms to be migrated
- arrange acceptance testing by the users.

The migrated forms can be maintained with tools provided by Quintessence Systems. Any form can then be replaced with a form using Oracle's toolset, but still call or be called by a migrated form. Some priority forms will be rewritten immediately, but other rarely used forms need not be rewritten for years.

3. The Migration Process

Forms are stored as .fmb files. These are converted by Oracle tools to equivalent XML files. The in2j migration tool takes the XML file and converts the PL/SQL into Java which it then places in a single .JAVA file. This Java file can then be compiled and executed with a Java IDE such as JDeveloper (recommended) or Eclipse.

Migration is fast and will create the Java for a large form in about 5 seconds.

Only two external libraries are required. The Oracle JDBC driver (ojdbc14.jar) connects the application to the database and our in2j.jar.

Menus and objects libraries are also converted to XML and migrated with the forms. PL/SQL libraries are converted to text (PLD) and also converted to Java. Both the XML file and the Java file are used at runtime. The XML file contains the information used to create the presentation layer. The Java file contains the business logic.

Oracle reports are also converted to XML and Java.

4. The Runtime Framework

The migration process extracts the PL/SQL from the XML file and converts it to Java. The in2j framework contains a complete reimplement of PL/SQL, Forms and Reports in Java. For instance an ON-CHECK-DELETE-MASTER trigger may contain:

```
DECLARE
    Dummy_Define CHAR(1);
    CURSOR EMP_cur IS
        SELECT null FROM EMP
        WHERE DEPTNO = :DEPT.DEPTNO;
BEGIN
    OPEN EMP_cur;
    FETCH EMP_cur INTO Dummy_Define;
    IF ( EMP_cur%found ) THEN
        Message('Cannot delete master record when matching detail records
        exist.');
```

```

        RAISE Form_Trigger_Failure;
    END IF;
    CLOSE EMP_cur;
END;

```

this becomes:

```

final Variable dummyDefine = newVariable();
final Cursor empCur = newCursor();
try
{
    dummyDefine.setType( newCharType("1") );
    empCur.setOracleSql( "SELECT null " +
        "FROM emp " +
        "WHERE deptno = :1" );
    empCur.setArgument( 1, newItemVariable("dept","deptno") );
    empCur.open();
    empCur.fetch();
    empCur.getResult( 1, dummyDefine );
    if (( empCur.isFound() ).isTrue())
    {
        message( newStringVariable("Cannot delete master record when
matching detail records exist.") );
        empCur.close();
        throw getFormTriggerFailure();
    }
    empCur.close();
}
catch (CodeException e1$)
{
    throw e1$;
}

```

The in2j framework library contains the classes Variable and Cursor, these behave exactly as PL/SQL variables and cursors do. For instance the class Cursor has methods: setArgument(Variable), open(), fetch(), getResult(int,Variable), close(). Each of these methods is implemented using JDBC but the effect is a cursor which acts identically to an Oracle Cursor.

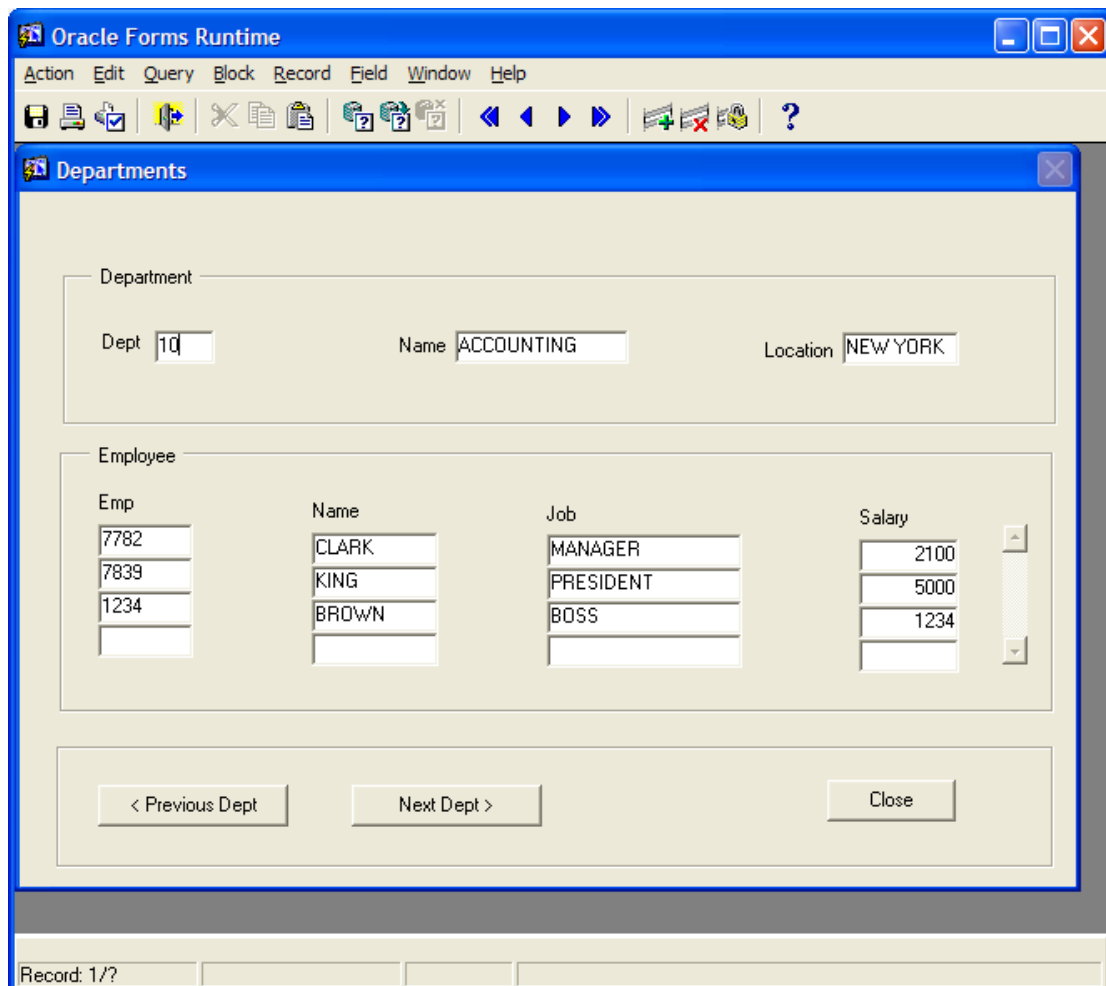
As well as classes to implement PL/SQL the in2j class library also implements the entire functionality of Oracle Forms and Reports. These are the 178 classes.

Alert	Debug	Menu	RowIdType
AlertType	DecimalType	MenuItem	RunForm
AnyCharacterType	DecType	MenuItemType	RunFormFrame
AnyNumericType	Dictionary	MenuModule	ScalarType
Argument	DoublePrecisionType	MessageAlert	Script
ArgumentManager	ExecSql	MessageBox	SearchReplace
BFileType	Executable	Migrated	SignType
BinaryIntegerType	FloatType	MisLabelType	SmallIntType
BLOBType	Form	Module	Standard
Block	FormModule	ModuleObject	StandardExtensions
BlockMenu	FormModuleType	ModuleObjectType	StringType
BlockRecord	Forms40	NaturalNType	SubType
BlockRecordItem	Forms4w	NaturalType	TableType
BlockType	Ftree	NCharType	TabPage
BooleanType	FtreeNode	NCLobType	TabPageType
Canvas	FtreeNodeRenderer	NumberType	TextIo
CanvasType	FtreeNodeType	NumericType	TextItemEditor
Catalog	Function	NVarchar2Type	Timer
CharacterType	GotoException	ObjectLibrary	TimerType

CharType	GroupColumn	Ole2	TimestampType
CLobType	GroupColumnType	OleObjType	ToolEnv
CodeException	Help	OraFfi	ToolErr
CodeLibrary	Htf	OraJava	ToolRes
CollectionType	Htp	OraNls	Trigger
CompositeType	In2Object	OraProf	TriggerAbortedException
Connect	IntegerType	PackageBody	TriggerThread
Cursor	IntType	ParameterList	Type
DatabaseObjectType	Item	ParameterListType	UtilFile
DatabaseProcedure	ItemType	Pecs	Varchar2Type
DatabaseTableType	Jni	PlsIntegerType	VarcharType
DatabaseType	Keys	PlsqlTable	Variable
DatabaseVArrayType	Lib	PositiveNType	VArrayType
DateLov	LibException	PositiveType	Wbx
DateTimeType	List	Procedure	Viewport
DateType	LobType	QueryWhere	ViewportType
Db2Lib	LocalFunction	RawType	VisualAttribute
DbmsAlert	LocalProcedure	RealType	Vt
DbmsLob	LongRawType	RecordGroup	VtType
DbmsOutput	LongType	RecordGroupType	Web
DbmsPipe	Lov	RecordType	Window
DbmsSql	LovColumnMapping	ReferenceCursorType	WindowType
DbmsStandard	LovDialog	Relation	Wrappers
Dde	LovListCellRenderer	RelationType	XmlElement
	LovType	Report	XmlStore
		ReportType	XYConstraints
			XYLayout

Every 'object' which makes up PL/SQL and Forms has been represented as a Java class and executes just as the original would. Notice that the naming is consistent with the original Oracle naming, as is the naming of all the methods.

So:



Becomes:

The screenshot shows the 'in2j Forms Runtime' application window. The title bar includes standard window controls. The menu bar contains: Action, Edit, Query, Block, Record, Field, Window, Help. The toolbar contains various icons for file operations and navigation. The main content area is titled 'Departments' and is divided into two sections:

Department Section:

Dept	10	Name	ACCOUNTING	Location	NEW YORK
------	----	------	------------	----------	----------

Employee Section:

Emp	Name	Job	Salary
7782	CLARK	MANAGER	2100
7839	KING	PRESIDENT	5000
1234	BROWN	BOSS	1234

At the bottom of the form are three buttons: '< Previous Dept', 'Next Dept >', and 'Close'. The status bar at the bottom left shows 'Record: 1/?'.

The resulting form is as near to pixel accurate as is possible.

5. Completeness of Migration

It would have been nice to have a complete specification of how Forms / PL/SQL executes. Alas there is no such thing. The runtime has had to be developed from the available manuals. These are excellent in parts, however some areas – free, for example – are almost undocumented.

Almost all Forms systems use undocumented features. Although in2j now recognises a good many of these, there certainly remain features that in2j has yet to encounter, and will not recognise. All new migrations will contain some inconsistencies between the original Forms and the migrated Java version. All such inconsistencies are fixed as bugs in in2j, leading to a continuous improvement over time.

6. Forms Versions

in2j runs equally well with Forms 6i as Forms 10g. It also migrates Forms 4.5, but the character mode interface looks dated. The in2j approach should work for Forms

2/3 but has never been attempted. Anecdotal evidence indicates that Forms 3.0 is still widely used, though as it is unsupported no data is available.

7. SWING or HTML

in2j uses Swing as its presentation layer. Of course we get many requests for migrations into Java Server Pages (JSPs) but despite great advances, HTML-based technologies are still not powerful enough to represent the rich interface provided by Forms. Oracle is well-aware of the issues involved here, see:

<http://www.oracle.com/technology/pub/articles/nimphius-mills-swing-jsf.html>

Forms has a complex user interface consisting of layers of canvases which can be moved and hidden by triggers. Furthermore, fields are hidden, shown and validated by complex triggers. The above paper warns against using J2EE for Forms-like applications:

“To summarize the above, **you use Swing** today if your application requires immediate response to user inputs or events that change the user display. This also is true for complex User Interfaces that have multiple master-detail dependencies displayed on one screen. Applications that need to perform immediate item validation, which could be based on complex logic, also find better support in Swing than on the Web. “

The HTTP protocol was designed for presenting documents on the screen and not implementing user interfaces. Although satisfactory for systems which, say, book air tickets it does not provide sufficient support for the very complex database transactions which make up Forms applications. The above paper states:

“The Web model is based on the HTTP protocol, which is not designed for transactional and dynamic business applications. JavaServer Faces automatically handles a lot of the problems Web developers were facing in the past, but still, with handling multi row updates, there is an area that requires the developer to have his hands on. In contrast, In Swing multi row operations are no-brainers for application developers e.g. working with ADF Swing. “

Multi-row updates are very simple Forms operations compared to the complex trigger code which provides the usual business logic of most Forms applications.

Using Swing does not mean that the migrated application need to run as a desktop like Forms 6. Many technologies, such as UltraLightClient (ULC) provided by Canoo (<http://www.canoo.com>) allow a Swing application to run on any server supporting J2EE, such as Application Server, but the user interface can run on a browser and appear identical to Forms 10g.

8. Java, ADF and SOA

10 years ago when in2j was first conceived Oracle proudly declared itself 300% Java: <http://www.wired.com/science/discoveries/news/1998/04/11720>. That was a long time ago and Oracle no longer recommends migrating Forms to Java, instead offers its new ADF Faces technology based around JDeveloper, business objects and Java Server Faces within a Service-Oriented Architecture (SOA).

Quintessence Systems believes moving to Java is a step forward whilst maintaining a similar architecture to Forms. A migrated application using ULC has a very similar architecture to 10g, except it is Java from end to end. Similarly, a migrated application using straight Swing has an almost identical architecture to Forms 6.

Migrated code cannot exploit ADF as business objects cannot be identified without human intervention. However, the in2j framework provides an interface so any form can be rewritten and still be called by the menu or another form. This allows a form to be wholly rewritten using ADF so that over the long term the whole application will use business objects. Similarly the forms structure could be simplified to be 'JSF-ready'.

Service-oriented Architecture (SOA) allows applications to share code across systems. Unfortunately, even though a Forms application is running on middleware none of its business logic can be shared. Once migrated to Java any of the triggers, packages, procedures and functions that make up the Forms application will be available to a SOA.

9. The Migration Process

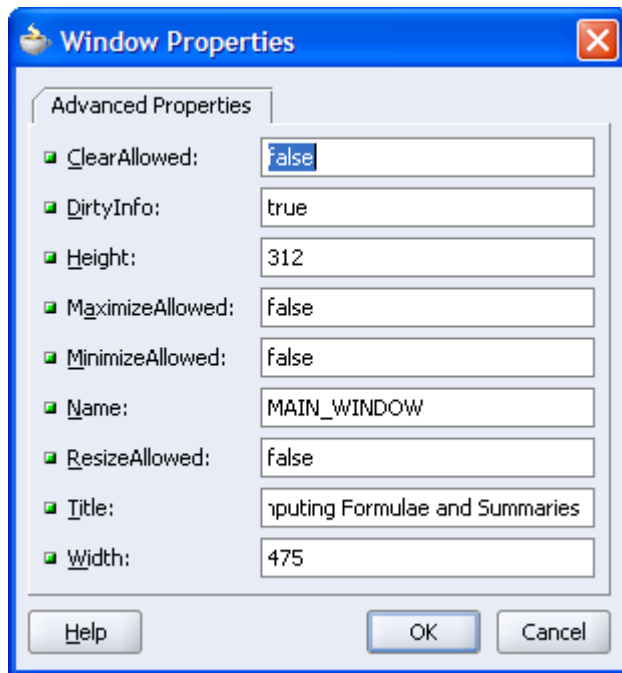
- 1) You send us a Proof of Concept (POC) sample of your code, complete with libraries (if necessary), a database (with data) and a script (with screenshots). We will migrate it and send it back as Java.
- 2) You tell us how much code you have in megabytes and we quote you a price.
- 3) You send us all your code, we migrate it and send it back as Java.
- 4) You test it and accept the system.
- 5) You pay us
- 6) You deploy the system.

Although the migration process is wholly automated, for reasons mentioned above in2j does not presently offer an 'out-of-the-box' solution. Of course, if your code is sensitive we can perform the whole process on-site.

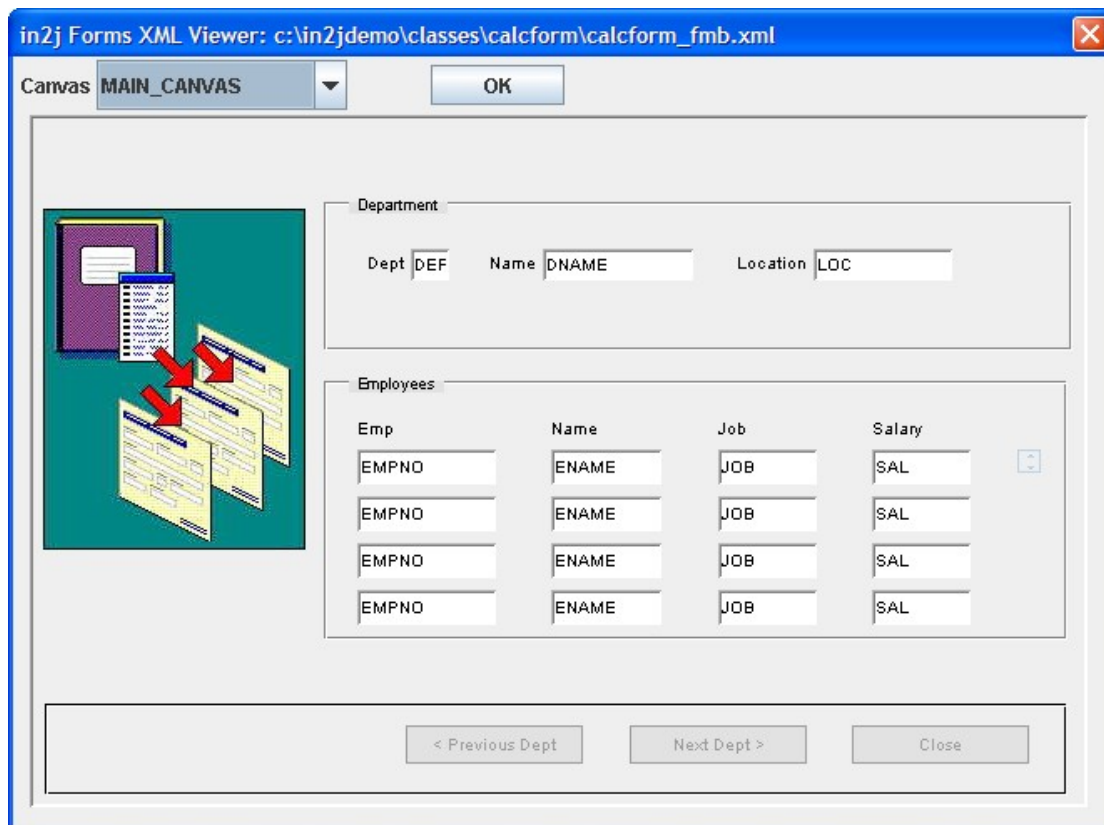
Other than on-site expenses, you are not invoiced until the system has passed acceptance testing.

10. Maintaining the Migrated System

The XML files contain the presentation layer. Before a form is properly rewritten it may be necessary to make small changes to the XML file to maintain the system. To change, say, block information this can be done directly using the XML tools in JDeveloper.



However for positioning items on canvases this could be very tedious and the application would have to be run to see if the changes were correct. The JDeveloper addin in2jView allows a canvas to be previewed without running the application.



Similarly the tool allows reports to be previewed.

The combination of the addin and JDeveloper's Java and XML editing tools gives a tool markedly similar to Forms Developer interface. It therefore provides a familiar development environment for a Forms developer moving to Java.

11. In conclusion

in2j provides a relatively quick and simple way for users of Oracle Forms and Reports to safeguard and upgrade the huge investment of time and resources they have made in their applications over many years. By migrating these entire applications into Java, organisations can in effect future-proof this investment and need no longer worry about upgrading to new Forms versions, or other long term support worries.

Once the Forms application is migrated to Java, Forms developers have access to JDeveloper and other tools with which to maintain and enhance the system, in a familiar environment similar to Forms Developer. The migrated Forms also offer a stepping stone for conversion to an ADF platform as a longer term destination.

END