



# Lycia

The rebirth of Informix 4GL

# Legal Notice

- The information contained within this presentation is subject to change without notice
- Please submit your feedback to [info@querix.com](mailto:info@querix.com)
- All rights reserved. No part of this document may be reproduced or transmitted, in any form means, without written consent of Querix (UK) Ltd.
- Lycia, LyciaDesktop, LyciaWeb are trademarks of Querix (UK) Ltd.
- All other products and company names used within this document are trademarks of their respective owners

# Agenda

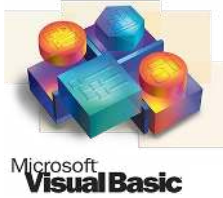
1. Evolution of the development software
2. Current challenges
3. The solution you have been waiting for

# 1. Evolution of the development software

# 80's: The success of Informix 4GL

- 2nd most used language at the time on UNIX platforms, after C
- Simple, fast, and powerful way of developing business applications:
  - Easy to learn
  - Native interface with relational databases
  - Use of forms and report writer
  - Support for calling C functions and vice versa
  - Platform independency
  - High performance without requiring huge hardware infrastructures
  - Native 4GL instruction-based debugger

# 90's: Domination of the OO languages



# 00's: Nothing but WEB



# Real added value?

- Advanced graphic rendering
  - It became possible to implement functionality that were impossible or too complex to implement before
  - Clouds allow for consistent infrastructure savings
- ? Productivity
  - ? Application maintenance time
  - ? Cost savings
  - ? Complexity of developments
  - ? Do code generators help with the real-life applications?
  - ? Savings in IT/IS departments

## 2. Current challenges

# Integrate with other applications

- Cloud, no Cloud ?
- Consume and provide web services
- Streams/asynchronous data input (Internet of Things)
- BOTS, CTI, IBM Bluemix™ ...

# Use rich and modern GUI

- Wide choice of the form controls
- Highly customizable
- Optimized for touch devices
- Responsive

# Get more benefit from the data

- Powerful report design tools, easily integrated into IDE
- Cognitive computing (ex: IBM Watson™ ...)

# Cut the costs

- Spend less money on projects
- Deliver on time and on budget
- Minimize the maintenance cost

# Mitigate the risks

Replacing a working application is always a risk on

- Budget
- Timeline
- User adoption
- Business continuity

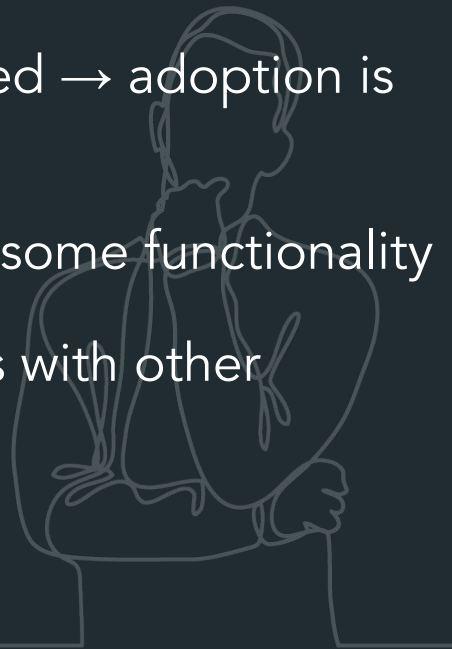
# Your 4GL application is an asset

- It perfectly fits your company's activity
- It is bug free
- Users drive it just as they drive their own car
- No tier-company dependency



# The dark side

- Young developers hate 4GL because it is text-based → adoption is getting worse
- Legacy 4GL does not technically allow to develop some functionality
- Your applications need to communicate both ways with other applications inside/outside of your company



Locked out in the legacy 4GL,  
you are thinking of rewriting the application

# What vendors want to sell you

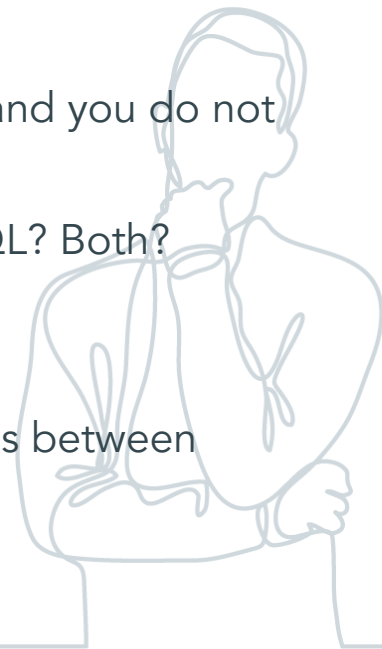
- Cloud
  - Infrastructure implementation trends are changing
  - Servers, network infrastructure, and databases become huge
  - Extensibility and scaleout are now requirements
- Big Data
  - You can do everything with Hadoop
  - SQL is dead, NoSQL is what you need
- Development tools
  - Java is easy; every developer knows Java
  - New IDEs and languages allow for the short development cycles
  - Do not develop, integrate with the 3rd party applications
  - Open source development systems are free
- ERPs
  - You have nothing to do after it is installed
  - It handles your activity better than your application does
  - It will rationalize your costs



Where it leaves you

# What technology to use?

- New OO languages and Web Dev systems:
  - generate higher costs of maintenance than I4GL
  - require powerful hardware infrastructure to run applications
- You do not really know which new language will be sustainable and you do not want to bet on the wrong horse
- The future of DBMS technology is uncertain for you: SQL? NoSQL? Both?
- Open source software introduces new challenges:
  - no regular updates
  - upgrading often results in dealing with the compatibility issues between versions
  - GUI is clumsy

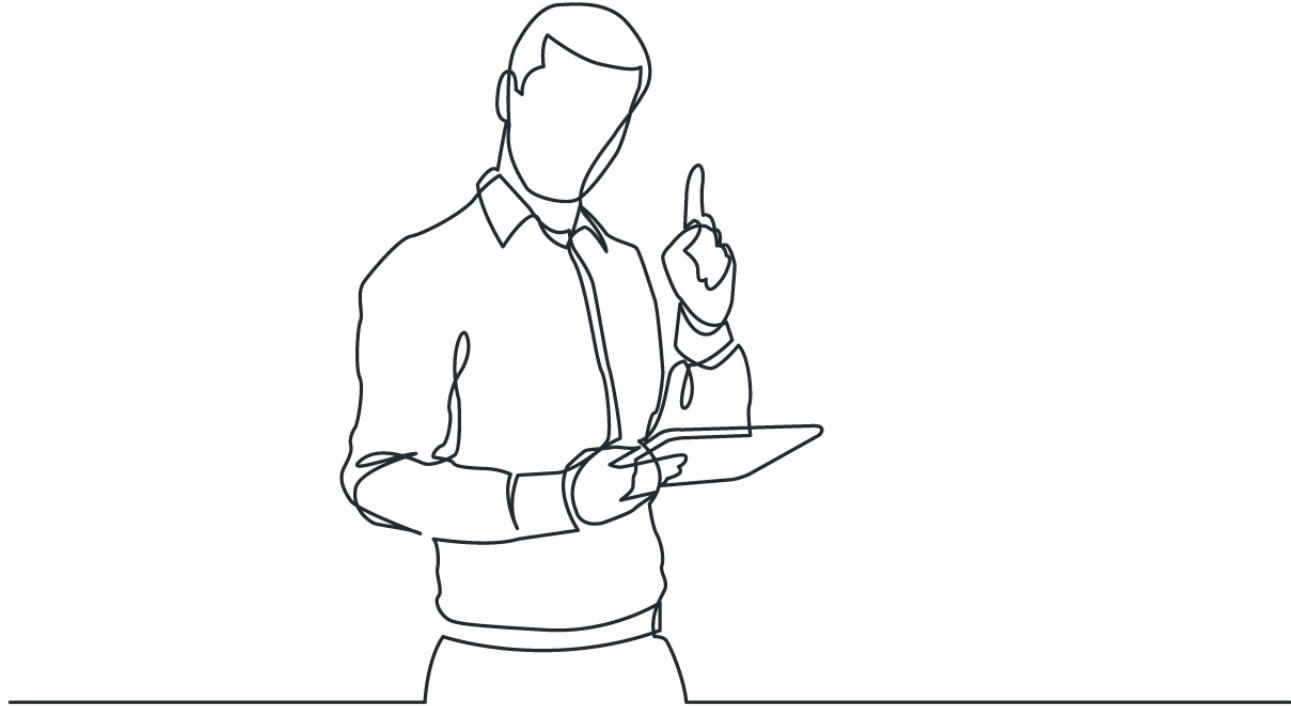


# How to involve the new expertise?

- Hire the required expertise:
  - hire developers & QAs in house
  - outsource the work
- Purchase new hardware to support the infrastructure changes
- Organize the knowledge transfer between developers:
  - enormous number of communications
  - do you have the sufficient product documentation?
  - do you have the sufficient technical documentation?
- Migrate all the data from the old system to the new one



# Time to ask the right questions.



Is it a good idea  
to waste decades  
of specific  
application  
development?



What if the conversion of my  
application to a new  
languages takes six years, or  
even ten, instead of one as  
initially promised?



Am I willing to entrust  
the development of  
my new app to  
people or companies  
who do not know my  
business?



To what extent is my  
company able to bear  
a project budget  
overrun?



Won't the adoption of an  
ERP make me dependent  
on the integrator and  
introduce unsuspected  
constraints?



Shouldn't I look for  
information about  
4GL application  
replacement  
projects failure rate,  
before it is too late?



It takes **years** to train a Java developer.

It only takes **a few weeks** to train a good 4GL developer!

3. The solution you have been waiting for

# Querix Lycia



# Full stack for the I4GL application development

Maximize the technical capacity of your classic apps or set off on the new development.

Run your I4GL apps in a modern and technologically open context.

# Why Lycia?

- Full I4GL compatibility → Low migration effort
- Rich and modern UI
- New functional challenges that Lycia makes possible
- Total control on your solution remains in your hands
- Regular platform updates that enhance your experience with Lycia
- Lycia is cross-platform and database independent
- Effortless installation and update
- Highly configurable and customizable
- Adaptable to various infrastructures

FRONT END

# Rich and modern GUI

- Thin graphical clients: web and desktop
- Both clients use the same application server and deployed only once
- Rich set of form controls
- Material UI, optimized for touch screens [coming soon](#)
- Responsive [coming soon](#)
- Highly customizable with Lycia Theme Designer and CSS

# This was your application

Vendor:      Edit    **New**    Delete    Search    Quit  
Add new vendor

Vendor Code	[   ]
Vendor Name	[   ]
+	
Address	[   ]
City	[   ]
Country	[ UNITED KINGDOM ]      State [   ]      ZIP [   ]
+	
Currency	[ GBP ]
Vendor Type	[   ]
Term Code	[   ]
Tax Claim	[   ]
ABN	[   ]      Tax Inclusive Pricing [ N ]

F1=Apply    ESC=Cancel

# This will be your application

The image shows a web application interface for 'Querix Lycin'. On the left is a dark sidebar with a menu. The main area is titled 'Demo Application' and contains a horizontal navigation bar with tabs: VENDORS (active), CUSTOMERS, MIDDLEMEN, EMPLOYEES, INVOICES, CHECKS, and EXPENSES. A modal window titled 'Add New Vendor' is open, displaying a form with three sections: Vendor Info, Address Info, and Account Info. The Vendor Info section includes fields for Code, Name, and a label 'Vendor name or Company name'. The Address Info section includes fields for Address Line 1, City, State, and Postal Code, with a label 'State or province' under the State field. The Account Info section includes a Currency dropdown (set to 'United Kingdom Pound Sterling (GBP)'), a Vendor Type dropdown, and a Term Code dropdown. A 'Taxer' icon is visible in the sidebar menu.

Querix Lycin

DEMO APPLICATION

- Transactions
- Reports
- Roles
- Vendors**
- Customers
- Middlemen
- Employees
- Taxer
- Income Tracker
- Planner
- Inventory

Demo Application

- VENDORS
- CUSTOMERS
- MIDDLEMEN
- EMPLOYEES
- INVOICES
- CHECKS
- EXPENSES

### Add New Vendor

#### Vendor Info

Code

Name

Vendor name or Company name

#### Address Info

Address Line 1

Address Line 1

City

State

State or province

Postal Code

Country

United Kingdom (GB)

#### Account Info

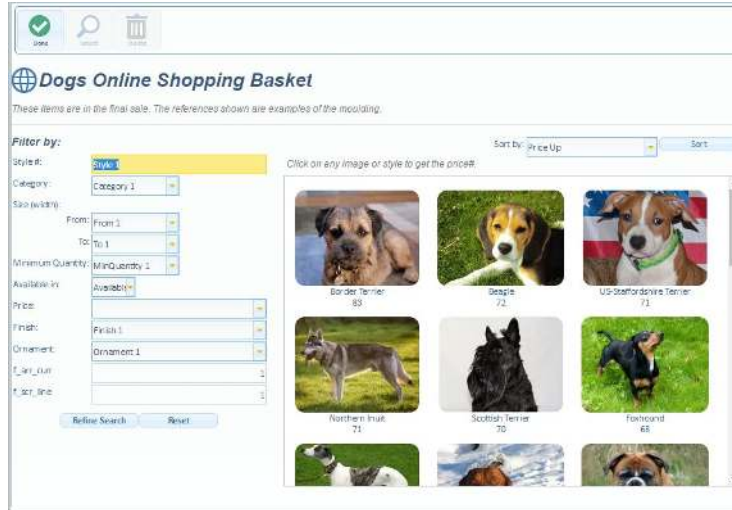
Currency

United Kingdom Pound Sterling (GBP)

Vendor Type

Term Code

Convert the most sceptical people.  
They won't believe this is a 4GL application:



Check [Lycia online demo](#) at [official Querix website](#)

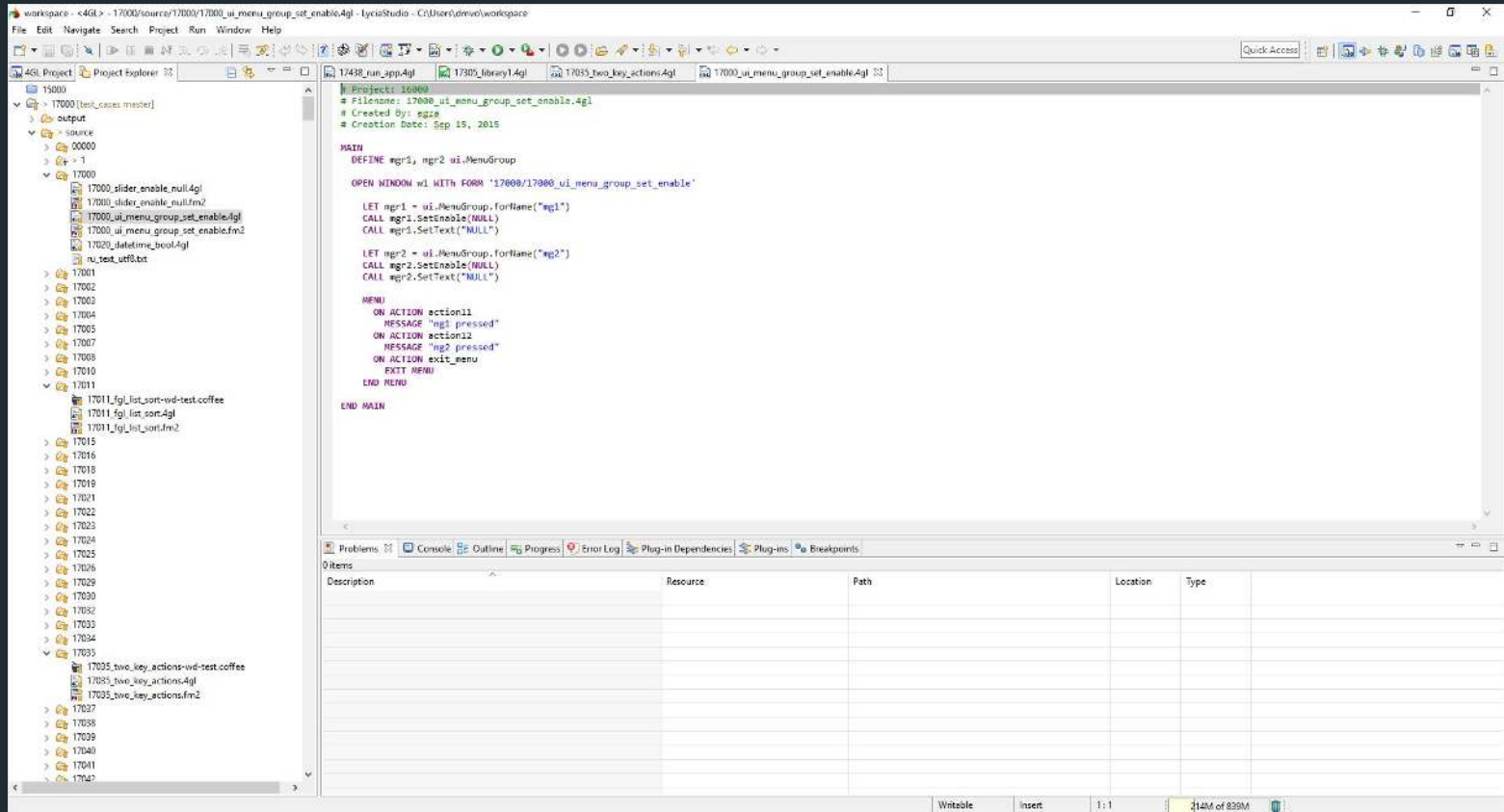
LyciaStudio

# Eclipse-based



- One of the most used IDEs
- Runs on any platform that supports Java
- Rich library of plugins and extensions
- Interfaces with Java, C/C++, REST, ZeroMQ, Swagger, JS, etc...  
→ allows your 4GL programs to call and be called by the routines developed in those languages

# LyciaStudio



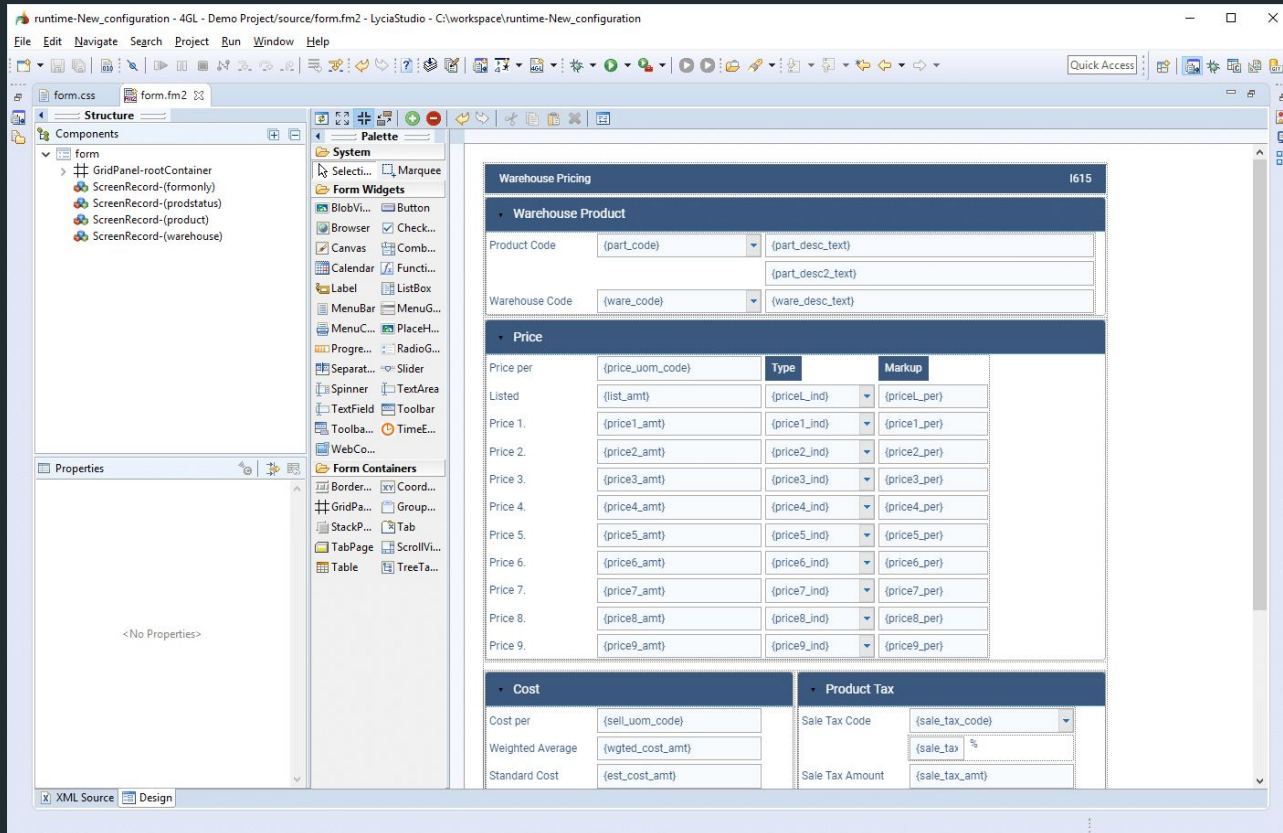
# LyciaStudio key components

- Project manager
- Code editor
- Visual debugger
- Version control
- Graphic Form Designer
- Static 4GL code analyzer
- Theme Designer
- Command line tools
- Local application server to test and debug your programs

# Productivity features

- Static 4GL code analysis
- Application deployment during runtime
- Interactive debugger
- Remote deployment and debug
- Multiple run configurations
- Local history of the source file edits
  - Compare different files or file revisions
  - Revert a source file to its previous revision
- Search & replace facility

# Form Designer



- Use a WYSIWYG layout builder or edit an XML source code directly
- .per and .fm2 form formats are supported, as well as conversion from .per to .fm2
- Rich set of layout containers and widgets (form controls)
- Object properties for better control of the form behavior and layout
  - Configurable
  - Can be redefined in Theme Designer and CSS
  - Bulk edit of properties for multiple objects
- Events to trigger actions, functions, batches, keys, program execution, etc.
- Widgets can be dynamically replaced at runtime, i.e. using touchControls or special wrappers, to support new input devices, user limitations or impairments
- Any widget can be morphed to any widget with a few exceptions

# 4GL Analyzer

The image displays three overlapping screenshots of the 4GL Analyzer software interface, illustrating its capabilities in analyzing 4GL programs.

**Top Screenshot (File Explorer View):** Shows the 'd4-simple' project structure. The left sidebar lists files like `d4_cust.4gl`, `d4_demo.4gl`, `d4_globals.4gl`, `d4_main.4gl` (marked MAIN), `d4_orders.4gl`, `d4_report.4gl`, `d4_stock.4gl`, `pganarchpath.4gl` (marked MAIN), `stores_demo.4gl` (marked MAIN), and `stores.4gl` (marked MAIN). The main pane shows the `d4_cust.4gl` file with tabs for FUNCTIONS, REPORTS, and SOURCE CODE. The FUNCTIONS tab is active, showing a list of functions: `print_labels ()`, `customer ()`, `add_customer (repeat)`, and `input_cust ()`.

**Middle Screenshot (Function Definition View):** Shows the source code for the `add_customer (repeat)` function. The code is as follows:

```
112
113 FUNCTION add_customer (repeat)
114   DEFINE repeat INTEGER
115
116   CALL clear_menu ()
117   MESSAGE "Press F2 or CTRL-Y for field help ";
118   "F2 or CTRL-Y to return to menu"
119   IF repeat THEN
```

**Bottom Screenshot (Table Schema View):** Shows the schema for the `customer (1)` table, declared in `d4-simple\stores_demo.4gl`. The table has the following columns:

Column	Type	Size	Nulls
customer_num	SERIAL		NOT NULL
fname	CHAR	15	
lname	CHAR	15	
company	CHAR	20	
address1	CHAR	20	
address2	CHAR	20	
city	CHAR	15	
state	CHAR	2	
zipcode	CHAR	5	
phone	CHAR	18	

- Focus on a program, not on a project
- Programs, libraries, modules, functions, reports, globals
- Handling duplicates and namesakes
- Database usage, incl. regular and temporary tables
- Multiple encodings supported
- .html output

# Reporting

The screenshot displays the LyciaStudio Report Designer interface for a report titled "Sales Order". The main design area shows a form with the following sections:

- Sales Order** (Header)
- Billing Address** (Form fields: [comp\_name], [comp\_addr1], [comp\_city], [comp\_zip], [comp\_country])
- Contact** (Form fields: [cont\_fname], [cont\_lname], [cont\_phone], [cont\_fax])
- Ship To** (Form fields: [comp\_name], [del\_address1], [del\_address2], [del\_address3])
- Invoice Date** (Form field: [invoice\_date])
- Invoice Number** (Form field: [invoice\_id])
- Item Table** (Table with columns: Item ID, Qty, Item Cost, Item Description, Item Tax (%), Line Total)
- Total Amount Due** (Form field: [Total])

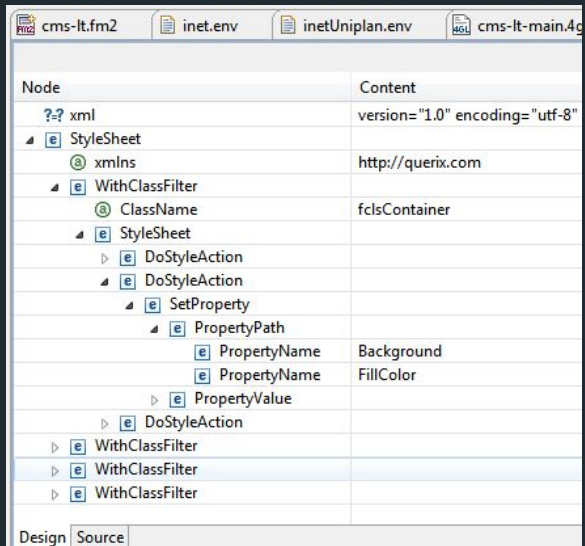
The interface includes a left sidebar with a tree view showing the project structure, including "Shared Resources" and "new\_report.rptdesign". The bottom panel shows the "Properties" window for the selected element, displaying various settings such as Name, Element ID, Font, Size, Color, and Background color.

**Properties Window - General**

Property	Value
Name	
Element ID	505
Font	Arial
Size	10 points
Color	#53707C
Background color	Auto

- BIRT <https://www.eclipse.org/birt/>
- 12M+ downloads, 2.5M+ developers across 157 countries.  
IBM, Cisco, S1 and ABS Nautical Systems incorporated BIRT into their product lines
- BIRT designs are persisted as XML and can access a number of different data sources including JDO datastores, JFire Scripting Objects, POJOs, SQL databases, Web Services and XML
- Data visualizations and reports can be embedded into rich client and web applications

# Theme Designer

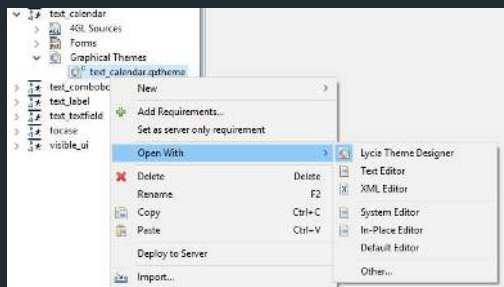


The screenshot shows the Theme Designer application with a tree view on the left. The tree view has a 'Node' column and a 'Content' column. The structure is as follows:

Node	Content
xml	version="1.0" encoding="utf-8"
StyleSheet	
xmlns	http://querix.com
WithClassFilter	
ClassName	fclsContainer
StyleSheet	
DoStyleAction	
DoStyleAction	
SetProperty	
PropertyPath	
PropertyName	Background
PropertyName	FillColor
PropertyValue	
DoStyleAction	
WithClassFilter	
WithClassFilter	
WithClassFilter	

At the bottom, there are tabs for 'Design' and 'Source'.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <StyleSheet xmlns="http://querix.com">
3   <WithClassFilter ClassName="fclsContainer">
4     <StyleSheet>
5       <DoStyleAction>
6         <ApplyClass Name="uclsContainer" />
7       </DoStyleAction>
8       <DoStyleAction>
9         <SetProperty>
10          <PropertyPath>
11            <PropertyName>Background</PropertyName>
12            <PropertyName>FillColor</PropertyName>
13          </PropertyPath>
14          <PropertyValue type="SystemColor" SystemColorName="LightYellow" />
15        </SetProperty>
16      </DoStyleAction>
17      <DoStyleAction>
18        <SetProperty>
19          <PropertyPath>
20            <PropertyName>Margin</PropertyName>
21          </PropertyPath>
22          <PropertyValue type="Thickness" Left="5" Top="5" Right="5" Bottom="5" />
23        </SetProperty>
24      </DoStyleAction>
25    </WithClassFilter>
26  </WithClassFilter>
27 </StyleSheet>
28 </WithClassFilter>
29 </WithClassFilter>
30 </WithClassFilter>
31 </StyleSheet>
32 </WithClassFilter>
33 </WithClassFilter>
34 </WithClassFilter>
35 </WithClassFilter>
36 </WithClassFilter>
37 </WithClassFilter>
38 </WithClassFilter>
39 </WithClassFilter>
40 </WithClassFilter>
41 </WithClassFilter>
42 </WithClassFilter>
43 </WithClassFilter>
44 </WithClassFilter>
45 </WithClassFilter>
46 </WithClassFilter>
47 </WithClassFilter>
48 </WithClassFilter>
49 </WithClassFilter>
50 </WithClassFilter>
51 </WithClassFilter>
52 </WithClassFilter>
53 </WithClassFilter>
54 </WithClassFilter>
55 </WithClassFilter>
56 </WithClassFilter>
57 </WithClassFilter>
58 </WithClassFilter>
59 </WithClassFilter>
60 </WithClassFilter>
61 </WithClassFilter>
62 </WithClassFilter>
63 </WithClassFilter>
64 </WithClassFilter>
65 </WithClassFilter>
66 </WithClassFilter>
67 </WithClassFilter>
68 </WithClassFilter>
69 </WithClassFilter>
70 </WithClassFilter>
71 </WithClassFilter>
72 </WithClassFilter>
73 </WithClassFilter>
74 </WithClassFilter>
75 </WithClassFilter>
76 </WithClassFilter>
77 </WithClassFilter>
78 </WithClassFilter>
79 </WithClassFilter>
80 </WithClassFilter>
81 </WithClassFilter>
82 </WithClassFilter>
83 </WithClassFilter>
84 </WithClassFilter>
85 </WithClassFilter>
86 </WithClassFilter>
87 </WithClassFilter>
88 </WithClassFilter>
89 </WithClassFilter>
90 </WithClassFilter>
91 </WithClassFilter>
92 </WithClassFilter>
93 </WithClassFilter>
94 </WithClassFilter>
95 </WithClassFilter>
96 </WithClassFilter>
97 </WithClassFilter>
98 </WithClassFilter>
99 </WithClassFilter>
100 </WithClassFilter>
```



- Style up the applications, forms, and widgets with the themes
- Apply a theme to all applications, a subset, or a single one
- Manipulate with themes in a graphic or an XML editor

# Teamwork capabilities



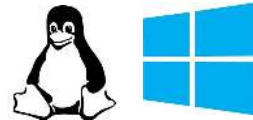
- Built-in version control tools:
  - Store your files securely and provide easy access to them
  - Collaborate on your projects
  - Compare and manage revisions
  - Ensure efficiency and rapidity of application development and maintenance
- Subversion, Git, CVS

MORE

# Language-agnostic compiler

- Retargetable compiler that uses an intermediate representation (IR)
- Compile and deploy code once and run anywhere
- Leaves an open door to the integration of other popular languages such as Java, Python, Ruby, C++, C#, D, Swift, and more...

# Application server



- Deploy the ready-to-use applications to the application server to run them in the production environment
- Runs on Windows and Linux
- Can be installed on premises or in the Cloud
- To ensure the highest availability:
  - include the load balancer
  - run on multiple application servers

# Lycia and the Cloud



Be it public or private cloud, use the built-in Docker support:

1. Package your application with all of its dependencies into a container
2. Deploy it on any Cloud platform

# Integration capabilities

- Include Lycia applications into existing web sites
- Directly call routines and methods from other languages:
  - Include and execute custom C and C++ functions calls
  - Invoke Java methods from 4GL, with inbound and outbound parameters
- Plug Lycia into the message queuing systems
- Consume the internal or external web services
- Employ the rich library of XML and JSON handling methods
- Use the DDE library methods to communicate with Office documents or the browser API to work with the Office Cloud

Dynamically run client-side JavaScript routines from your 4GL application:

```
IF fgl_getenv("qx_child") IS NULL THEN -- only load stuff if it is not a child process
CALL ui.interface.frontcall("html5","styleImport",["qx://application/extensions/_master.css"],[])
CALL ui.interface.frontcall("html5","scriptImport",["qx://application/extensions/custom-v3.js","nowait"],[])
CALL ui.interface.frontcall("html5","scriptImport",["qx://application/extensions/messages.js","nowait"],[])
CALL ui.interface.frontcall("html5","scriptImport",["qx://application/extensions/alterClass.js","nowait"],[])
CALL ui.interface.frontcall("templateModule","changeFooterFrameTemplate",[],[]) --change the viewports/templ
```

Dynamically import CSS files and obtain the highest standard of GUI control:

```
*watson_demo.css
@import "https://code.jquery.com/ui/1.11.4/themes/start/jquery-ui.css";
.qx-identifier-dialog_tab .qx-brow > table,
.qx-identifier-dialog_tab .qx-brow tbody {
  display: flex;
  width: 100%;
  height: 100%;
}
.qx-identifier-dialog_tab .qx-brow > table {
  border-collapse: separate !important;
}
```

# REST API

Web-enable your applications using LyciaWeb API with no 4GL code change:

- Call external web services from your 4GL programs
- External web services can call business logic written in 4GL
- Add extra layers: gateway, URL rewriter, load balancer, etc.
- Scalable to thousands of users
- Flexible web service life cycle model: application, session, request
- REST Services for users and developers
- Secure authentication

# EXTENDING THE 4GL

# Legacy 4GL nightmares

- Limited handling of ARRAYS:
  - Fixed size
  - Can't be passed as a function parameter
  - Programs can't handle the cursor movement between the array elements
  - INPUT ARRAY and DISPLAY ARRAY are "closed" blocks, with a binding logic
- No commands to handle file system files
- Using OS command with RUN
  - Leads to compatibility issues between different OSs
  - Low portability

With Querix 4GL, they are all gone

- 100% compatibility with I4GL + standard SQL language support
- Easy manipulation of static and dynamic tables
  - Insert or delete a row, resize an array, move the cursor programmatically
  - Pass the dynamic arrays as arguments in a function call
  - Dynamically sort an array by simply clicking on the column header
  - Search in an array with the search box
- Pass the cursors as variables and use in different modules
- New data types and associated methods (string, byte, cursor, form, window ...)
- Handling of system objects, incl. files and directories, with Lycia commands and not with OS commands or shell commands
- Regular expressions



Lycia enables your 4GL application to keep pace with technology and successfully live in today's world.

<https://querix.com/products/lycia>