

KOKUM

GUI Designer

User Manual & Practical Workbook

Visual forms. Signals. Slots. Working applications.

A complete, source-based guide to designing forms, connecting events to Kokum code and delivering responsive GUI applications.

The screenshot shows a window titled "Hello Designer". Inside, there is a label "Your name" next to a text input field containing "Anil". Below the input field is a checkbox labeled "Use uppercase" which is checked. Underneath the checkbox are two blue buttons: "Greet" and "Clear". At the bottom left of the window, the text "Hello, ANIL!" is displayed. The window has a light blue header and a white body.

A small form, a clear event contract, and a working Kokum application.

KOKUM 0.29.0 • PHASE 4.17.7.1

September 2026 edition

Kokum programming language created by Anil Jagtap

kokum.dev

ABOUT THIS BOOK

Read, build, bind and verify

This book teaches the GUI Designer as a practical development tool. It begins with a blank project, explains how layout and code are connected, and finishes with a small database-backed application. The central lesson is that a signal needs an explicit binding to a valid exported slot; naming a procedure after a button does not connect it automatically.

Edition and platform scope

The reference is the complete Kokum 0.29.0 Phase 4.17.7.1 source tree. Designer screenshots and teaching applications were exercised with the Linux compiler, native IDE and KokumVM. The Windows build hotfix does not constitute native Windows or macOS GUI certification. Use compatible target VMs and perform platform acceptance testing before release.

How to use the listings

Create each named project in the GUI wizard, draw the controls in its design table and connect the binding table. Replace the programmer-owned slot code with the printed listing, rather than appending duplicate stubs. Preserve the generated declaration blocks and allow Check to reconcile them. A project’s generated main.kokum, manifest and web files remain in place unless an example explicitly changes them.

Convention	Meaning
Code blocks	Editable source or commands; no line numbers to remove.
x, y, w, h	Logical design coordinates in pixels, relative to the parent.
true / false	JSON or property-editor Boolean values. Kokum source uses .T. / .F.
Signal / event	The same designer-facing event concept in this book.
SOURCE BASIS	Archive documents, implementation files or laboratory evidence listed in Appendix E.
CHECKPOINT / CAUTION	An observable result to test, or an important implementation boundary.

WORKING EXAMPLES Nine teaching projects were compiled, bundled and exercised through the real KokumVM/browser bridge. Reference-only fragments, external database services and platform-specific deployment instructions are identified separately. See Appendix D for the exact validation scope.

The source tree reports 37 schema controls. SubFormHost is retained for compatibility and hidden from ordinary new designs. CustomCanvas is retired from the public control contract; this book does not teach it as an available widget. [S1, S4]

CONTENTS

A practical route through the Designer

Click a chapter title to navigate. Page numbers include the cover and front matter.

1 From a visual form to running code.....	4
2 Start the IDE and choose a GUI project.....	6
3 Place, select and arrange controls.....	9
4 Build “Hello Designer”	12
5 Signals and slots: the exact contract.....	15
6 Automatic variables and control properties.....	21
7 Choice controls and the Items editor.....	23
8 Responsive forms and container layouts	26
9 Tables, trees and property grids.....	30
10 Menus, toolbars and shared commands.....	33
11 Subforms and controlled closing.....	36
12 Timer components and responsive work.....	39
13 Customer Desk: a small complete application	44
14 Check, build, run and package.....	49
Appendix A Control catalogue · basic and input.....	52
Appendix B Signal and property quick reference.....	55
Appendix C Practice tasks and acceptance criteria.....	57
Appendix D What was checked for this edition	59
Appendix E Source map and edition record.....	61

CHAPTER 1

From a visual form to running code

A GUI application has two cooperating parts. The browser renders the form and captures interaction. KokumVM keeps the persistent application session and executes Kokum code. The Designer edits the description that joins these parts; it is not the application's business-logic runtime.

Stage	What happens	Your responsibility
1 · Design	A .kform stores the control tree, IDs, typed properties and event targets.	Choose useful names, layout and parent relationships.
2 · Bind	A named event resolves to a slot in the declared Kokum module.	Create or connect the export and implementation.
3 · Check / build	The compiler validates form structure and bindings, then produces form resources and bytecode.	Resolve diagnostics before trusting runtime behavior.
4 · Run	A browser event is sent to the owning VM session.	Read current values and make bounded changes.
5 · Update	Validated state changes return to the browser as updates.	Change controls on the owner session, not inside a background worker.

A signal is a notification; a slot is the response

EVENT ROUTE — CONCEPTUAL

```

Button clicked
-> binding: hello.designer.slots.btnGreet_clicked
-> PROCEDURE(sender, event, form)
-> form.lblResult.Text = "Hello!"
-> browser updates the label

```

The clicked event says what happened. The binding names the routine to call. The slot reads application state and decides what should change. The same routine may serve more than one control when its behavior is genuinely shared.

PREVIEW IS NOT RUN Fit to Designer and Runtime Preview help inspect geometry. They do not replace executing the actual application, triggering an event and observing the slot result.

SOURCE BASIS [D1] Designer; [D2] slot binding contract; [D3] Web Form runtime.

CHAPTER 1 · PROJECT ANATOMY

Know which file owns each change

File / location	Owned content	Safe working practice
kokum.toml	Project identity, modules, forms, resources and launch settings.	Let the wizard create it; edit deliberately when adding a form or asset.
ui/main_window.kform	JSON layout, properties, IDs and signal targets.	Normally change it in the Designer. Source edits still require validation.
src/slots.kokum	Module exports, event handlers, helpers and managed declarations.	Write behavior here; preserve marked generated blocks.
src/main.kokum	The generated entry module and Main routine.	Keep the scaffold for these labs; GUI handlers belong in the slots module.
web/	Application HTML, CSS, JavaScript and runtime assets.	Preserve the current generated runtime files.
build/ and dist/	Generated resources, bytecode and distributable applications.	Rebuild rather than editing generated output.

Names that look similar have different jobs

The project/module name identifies compiled code, for example `hello.designer`. `MainWindow` is a form ID/name. `btnGreet` is a control ID. `Greet` is a visible caption. `btnGreet_clicked` is a routine name. Keep these roles separate when diagnosing a mismatch.

GENERATED ENTRY MODULE — HELLO DESIGNER

```
MODULE hello.designer VERSION "1.0.0"

EXPORT FUNCTION Main

FUNCTION Main AS INTEGER
  ? "Hello Designer initialized"
  RETURN 0
ENDFUNC
```

AUTHORING RULE Design the control, give it its final ID, and then connect its events. `PUBLIC txtName AS STRING` declares a value; it does not create a `TextBox` on a form.

SOURCE BASIS [L1] Wizard-generated Hello Designer project; [D4] automatic widget variables.

CHAPTER 2

Start the IDE and choose a GUI project

Build or install the Linux toolchain first. The three public programs used in this book are kokum-ide, kokumc and kokumvm. Old instructions referring to tlang-form or a separate form compiler are not the current workflow.

FROM THE KOKUM SOURCE / INSTALLATION DIRECTORY

```
./kokum-ide --no-restore
```

Open the local address printed by the IDE when automatic browser launch is not used. Use a fresh working directory for each teaching project. Save paths are on the Linux host running the IDE, not necessarily on the computer displaying the browser.

Create the first project

Choose New. Select GUI, enter Hello Designer as the project name, choose a writable Parent directory and use hello-designer as the Directory name. Confirm the generated module name hello.designer, then choose Create Project. A Console project does not provide the same form scaffold.

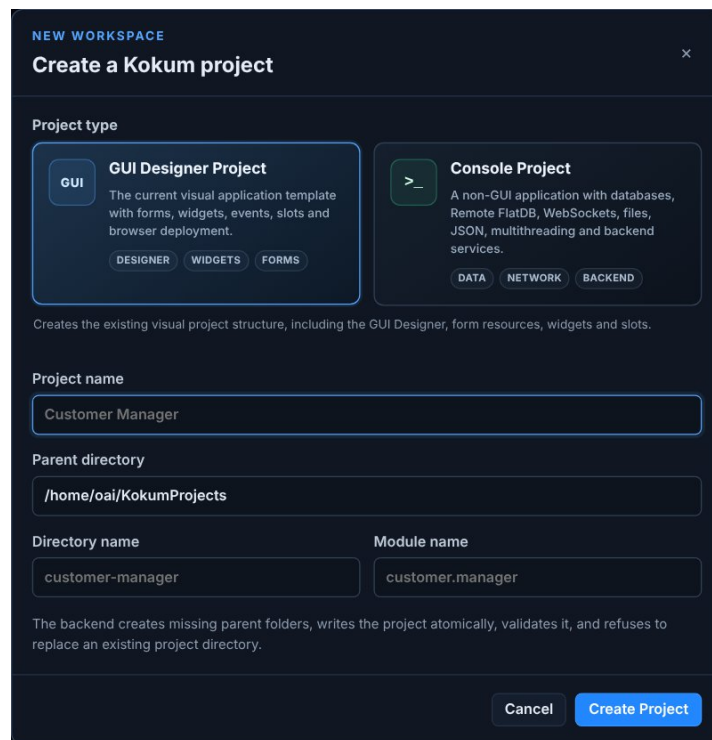


Figure 2.1 · The actual New Project dialog. GUI and Console are distinct project types.

NAMING Use descriptive names made from ordinary identifiers. Avoid language keywords in a generated module name: the HTTP lab is named HTTP Viewer, not Async GUI Lab.

SOURCE BASIS [L1] Native project wizard; [S2] IDE frontend; [V1] compiler checks.

CHAPTER 2 · WORKSPACE TOUR

Find the files, canvas and diagnostics

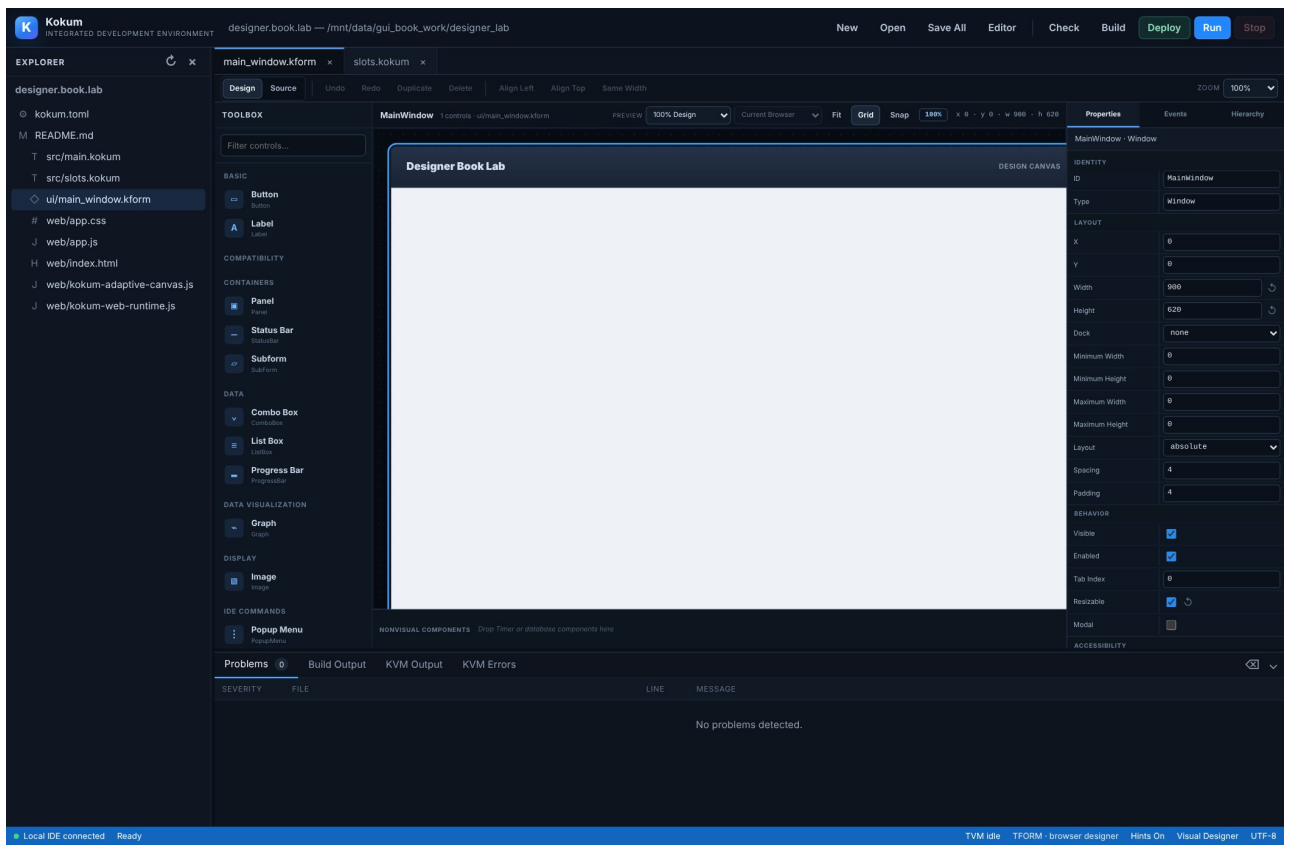


Figure 2.2 · Native IDE with the form open. The screenshot is an orientation view; the table below identifies the working areas.

Area	Use it for
Explorer · left	Open files. Double-click ui/main_window.kform to display the visual Designer.
Toolbox · next to canvas	Choose and filter control types. Nonvisual components live in their tray.
Canvas · centre	Select, move, resize and nest controls. Design and Source represent the same form document.
Properties / Events / Hierarchy · right	Edit a control, bind its signals, or inspect its true parent/child structure.
Problems / Build Output / KVM Output / KVM Errors	Read validation errors, build diagnostics, program output and runtime faults.
Check / Build / Deploy / Run / Stop · top	Validate, produce artifacts, package, start and stop the application.

SOURCE BASIS [S2] Current IDE labels; [L1] live IDE capture.

CHAPTER 2 · YOUR WORKING RHYTHM

Save, validate and test one change at a time

A reliable edit cycle

Open the form, add one small group of controls, and set their IDs and key properties. Connect one event and implement its handler. Use Save All, then Check. Build and Run after the first useful interaction exists. Test it before adding the next feature.

Autosave and document revisions support this workflow, but they are not a substitute for checkpoints or backups. Wait for pending edits to synchronize before interpreting build output. Check validates source and project consistency; Run exercises the actual event path.

Action	Expected observation
Open .kform	The Design tab shows the same control tree as Hierarchy.
Change text or size	The selected control updates; its ID remains stable.
Create / Edit a slot	The slots source opens and contains a valid export plus implementation.
Check	No unresolved required bindings or syntax errors.
Build / Run	The application opens independently of the designer preview.
Trigger the signal	KVM Output and the control state agree with the handler code.

Protect work during a conflict

When a slots revision conflict appears, stop repeatedly clicking Connect or Save. Preserve your unsaved source, inspect the current slots file, compare the revisions and reload deliberately. Avoid overwriting another writer's changes. Then Check and reconnect only the missing binding. A generated declaration update should not be "fixed" by deleting the user's routines.

NETWORK IDE All browser users of a network IDE act against host-side projects. Use the authenticated network-IDE setup described in the separate Network Programming manual; do not expose an unauthenticated development endpoint.

SOURCE BASIS [D1] Designer transactions; [D9] managed slots reconciliation and path identity.

CHAPTER 3

Place, select and arrange controls

Choose a control from the Toolbox and place it in the intended parent. For precise business forms, begin with an absolute layout and set x, y, width and height in Properties. Dragging is convenient; numeric values make a tutorial reproducible.

The parent is part of the layout

A label inside a Panel uses coordinates relative to the Panel, not the main Window. A control that visually overlaps a container is not necessarily its child. Use Hierarchy to confirm ownership after placing or reparenting it. In a layout-managed container, the layout can override a child's apparent hand placement.

Operation	Working method
Select one	Click the control; use Hierarchy for small, covered or nonvisual items.
Select several	Use Ctrl/Command to toggle membership or drag a selection rectangle across empty canvas.
Primary selection	Property changes target the primary control unless an explicit group command applies.
Move / resize	Drag the body or a resize handle; use numeric geometry for exact positioning.
Align / equalise	Use Align Left, Align Top or Same Width on an appropriate selection.
Duplicate / remove	Use Duplicate or Delete. Inspect new IDs and bindings rather than assuming copied behavior is correct.

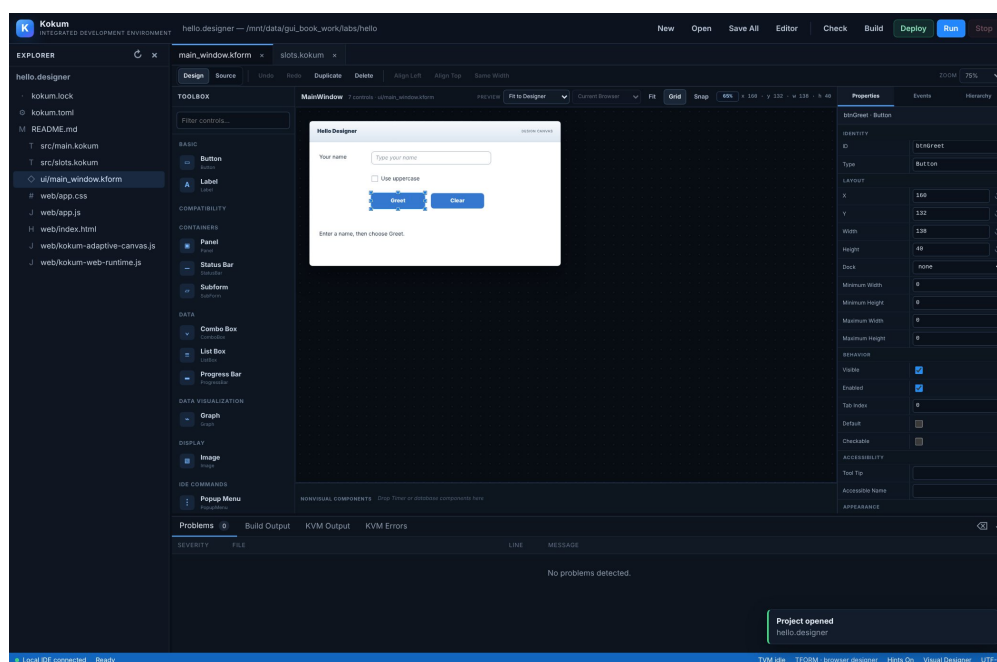


Figure 3.1 · Hello Designer arranged on the real canvas, with btnGreet selected.

SOURCE BASIS [S2] Designer interaction implementation; [L1] visual capture.

CHAPTER 3 · PROPERTIES

IDs, typed values and appearance

Set the ID before writing code. Use names such as txtCustomer, btnSave and lblStatus rather than Button17. An ID is used by bindings and form references; changing a visible Text caption does not rename the control. Avoid duplicate names, including names that differ only by letter case.

Property group	What to decide
Layout	x/y/width/height, minimum and maximum sizes, parent layout and dock behavior.
Behavior	Visible, Enabled and control-specific settings such as readOnly, running or selection.
Appearance	Text/title, theme-sensitive colours, font family/size/weight and cursor.
Accessibility	Accessible name, tooltip and a useful tab order. Do not use colour as the only error indicator.
Control-specific data	Items, columns, selectedPath, image source, database file or timer interval.

Reset means inherit the default

The reset action removes the explicit property value and returns to the schema/theme default. It is different from assigning an empty string or false. Foreground Color serializes as textColor; Background Color can use the system/theme default. Set a concrete colour only when it remains legible in the intended theme.

Typed values are not arbitrary text

A Boolean property accepts a Boolean; an enum uses the supported choices; dimensions need valid numbers. The Designer validates changes before committing the form. In JSON Source, use true and false. In Kokum slot code, use .T. and .F. A property display label may include spaces even when its stored key does not.

CHECKPOINT Rename a TextBox to txtName, set its text to an empty string and its placeholder to Type your name. The prompt must disappear when the user types; it must not become the input's actual value.

SOURCE BASIS [S1] Typed control schema; [D10] property editor and text-colour refinements.

CHAPTER 3 · PRECISION TOOLS

Grid, snap, zoom and safe editing

Control / gesture	Effect
Grid	Shows a visual design grid; it does not alone force snapping.
Snap	Enables alignment to the grid. Shift can temporarily request snapping; Alt bypasses it.
Zoom / Fit	Changes the design view, not the stored logical dimensions.
100% Design / Fit to Designer	Choose an editing view. Do not confuse scaling the canvas with resizing the form.
Runtime Preview	Inspect a browser-size presentation. Current Browser and fixed viewport presets help compare sizes.
Arrow keys	Nudge selected geometry. Shift+Arrow resizes; snapping may use grid-sized increments.
Ctrl/Command+D, C, V	Duplicate, copy and paste within the designer workflow.
Delete / Backspace	Delete the selection. Confirm that parent and children are the intended targets.
Ctrl/Command+Z; Shift+Z / Y	Undo; redo. The current focus must be in the intended editor.
Ctrl/Command+wheel; Space+drag	Zoom; pan. Middle-button dragging is also supported for panning.

When a control seems to disappear

Inspect its actual parent, Visible setting, dimensions, clipping container and selected tab. Zoom to fit before assuming deletion. A Timer or database component is intentionally nonvisual. A SubForm may have a separate design location and is not displayed at runtime until shown.

SAVE A CHECKPOINT After a successful layout change, use Save All and Check. Undo history is not source control, and deleting a bound control is not a reason to delete reusable slot code blindly.

SOURCE BASIS [S2] Designer commands and selection behavior; [D1] workflow.

CHAPTER 4

Build “Hello Designer”

LAB SETUP Create a GUI project named “Hello Designer”. Use MainWindow, width 650, height 320, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
lblPrompt · Label	MainWindow	24, 24, 130, 30	Your name
txtName · TextBox	MainWindow	160, 24, 310, 34	
chkUpper · CheckBox	MainWindow	160, 80, 220, 30	Use uppercase
btnGreet · Button	MainWindow	160, 132, 138, 40	Greet
btnClear · Button	MainWindow	314, 132, 138, 40	Clear
lblResult · Label	MainWindow	24, 204, 580, 68	Enter a name, then choose Greet.

Connect these three signals

Sender ID	Signal / event	Exported slot
txtName	enterPressed	btnGreet_clicked
btnGreet	clicked	btnGreet_clicked
btnClear	clicked	btnClear_clicked

The TextBox and button intentionally share btnGreet_clicked. The checkbox needs no extra handler for this example: the current chkUpper value is available when either greeting signal arrives. Keep lblResult wordWrap true so the validation message fits.

CHECKPOINT Before writing the handler body, verify all three required bindings in Events. The generated module must be hello.designer.slots and the form must name that same module.

SOURCE BASIS [L1] Complete compiled and browser-tested Hello Designer project.

CHAPTER 4 · FIRST SLOTS

Write the greeting and reset logic

Place the following programmer-owned code in `src/slots.kokum`. Keep the managed variable block created from the form below the routines; do not add duplicate `PUBLIC` declarations. Save All and run Check so `txtName` and `chkUpper` are synchronized with the form.

KOKUM

```
MODULE hello.designer.slots VERSION "1.0.0"

EXPORT PROCEDURE btnGreet_clicked
EXPORT PROCEDURE btnClear_clicked

PROCEDURE btnGreet_clicked(sender, event, form)
    LOCAL name AS STRING
    name = txtName
    IF LEN(name) = 0
        form.lblResult.Text = "Please enter a name."
        RETURN
    ENDIF
    IF chkUpper
        name = UPPER(name)
    ENDIF
    form.lblResult.Text = "Hello, " + name + "!"
ENDPROC

PROCEDURE btnClear_clicked(sender, event, form)
    txtName = ""
    chkUpper = .F.
    form.lblResult.Text = "Enter a name, then choose Greet."
ENDPROC
```

The greeting reads the current `TextBox` value, rejects an empty string, optionally converts it to uppercase and updates the `Label`. This is deliberately a minimal validation rule: spaces alone are not rejected. The reset handler changes the input variables and the feedback label.

CASE AND STRINGS Kokum identifiers are case-insensitive. Text inside quoted strings is data and retains its case. Use the printed routine and control names consistently even when the language permits case variations.

CHAPTER 4 · OBSERVE THE RESULT

Run the form, not just the preview

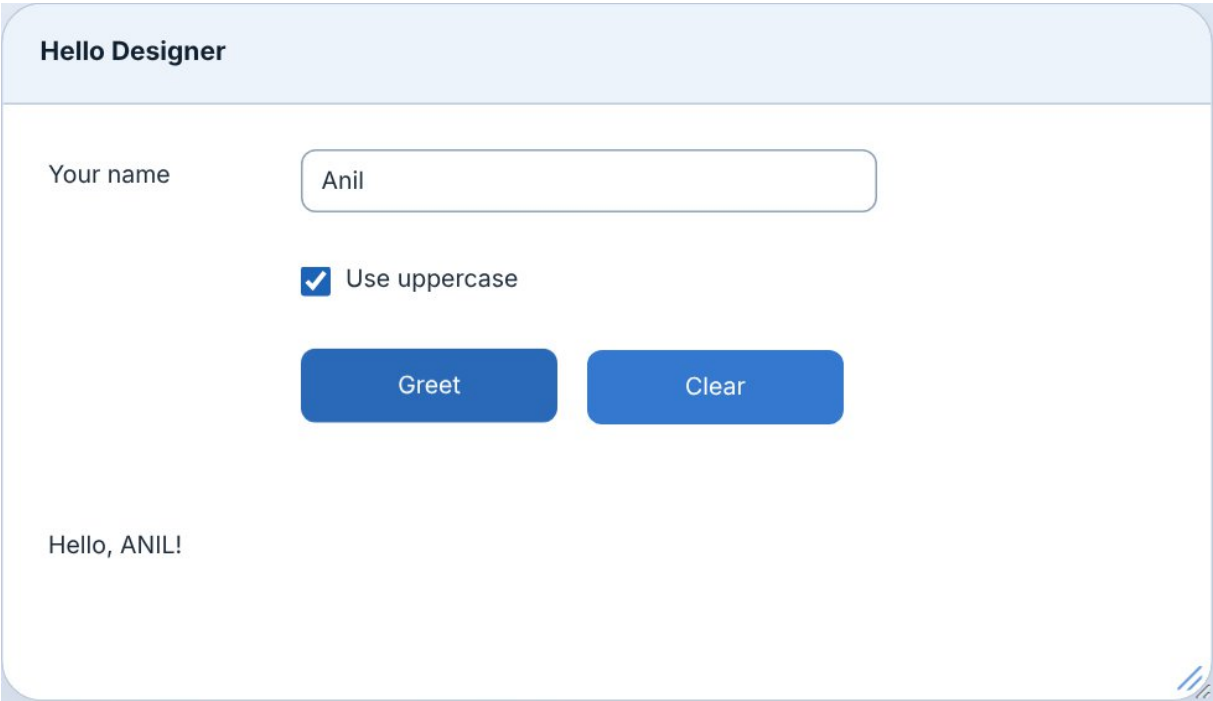


Figure 4.1 · Native KokumVM result after entering Anil and selecting Use uppercase.

Try this	Expected result
Click Greet with an empty TextBox.	Please enter a name.
Type Anil; tick Use uppercase; click Greet.	Hello, ANIL!
Click Clear.	The input is empty, the checkbox is cleared and the prompt returns.
Type Maya; press Enter in the TextBox.	Hello, Maya! The TextBox binding calls the same slot.
Change only the Greet caption.	Behavior is unchanged because the stable ID and binding are unchanged.

Diagnose a silent button

Check that clicked is connected, not merely that a procedure exists. Confirm the correct .kform is in the project, the application was rebuilt, and Run is launching the new artifact. Read Problems and KVM Errors before changing the source at random.

SOURCE BASIS [V1] Hello browser interactions passed through the actual VM bridge.

CHAPTER 5

Signals and slots: the exact contract

Select a control and open Events. The list is generated for that control type: a Button offers clicked, a TextBox offers textChanged and enterPressed, and a Timer offers timerTick. Display labels such as “Double Clicked” correspond to canonical names such as doubleClicked in the form.

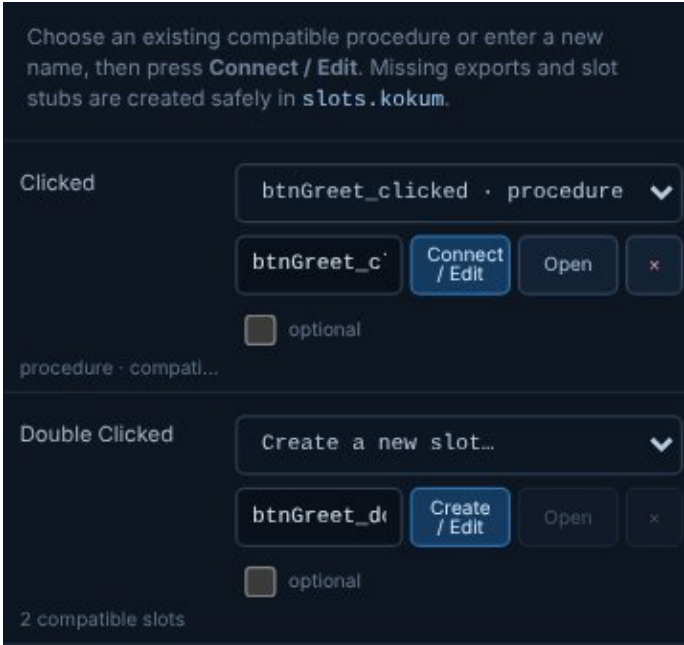


Figure 5.1 · Actual Events rows: an existing clicked slot and an unbound doubleClicked event.

Part	Meaning
Signal name	The event raised by the selected control; it must belong to that control's schema.
Slot selector / name	Choose a compatible routine or provide a new routine name.
Create / Edit or Connect / Edit	Create/connect the handler and open the code. Existing bodies are not meant to be replaced.
Open	Navigate to the current routine; it does not by itself create a new binding.
Disconnect ×	Remove this event connection, not the reusable procedure body.
optional	Allow a missing target deliberately. A resolved optional target still must be valid.

SOURCE BASIS [D2] Binding validation; [S2] actual Events UI; [L1] live Connect / Edit capture.

CHAPTER 5 · BINDING WORKFLOW

Create, reuse and disconnect safely

Create a new slot

Select btnGreet, open Events and locate Clicked. Choose Create a new slot..., accept btnGreet_clicked or enter a suitable name, then choose Create / Edit. The IDE records the binding and ensures a compatible exported routine is present. Replace the stub body, not the module identity or generated sections.

Connect an existing slot

Select txtName and locate Enter Pressed. Choose btnGreet_clicked from the compatible-routine list and choose Connect / Edit. The action can add a missing export while preserving the existing implementation. After the operation, verify the Events row and inspect the single routine in slots.kokum.

BINDING FRAGMENT INSIDE THE BUTTON'S .kform OBJECT

```
"events": {  
  "clicked": {  
    "module": "hello.designer.slots",  
    "slot": "btnGreet_clicked",  
    "optional": false  
  }  
}
```

The short form "clicked": "btnGreet_clicked" uses the form's declared slotsModule. The long form makes the module and optional policy explicit. This fragment is not a complete .kform document; do not paste it as a standalone file.

Disconnect or rename

Disconnect only the event that should stop calling the routine. Keep the procedure if other controls still use it. For renaming, update the export, implementation and every binding together, or use the Designer's supported operation and then Check. A text search helps reveal remaining references.

REQUIRED BY DEFAULT Do not mark a Save or Delete handler optional merely to suppress a build error. That turns a missing action into a deliberate no-op rather than correcting the program.

SOURCE BASIS [D2] .kform target forms and required/optional semantics; [S2] slot editing.

CHAPTER 5 · INSPECT THE CONNECTION

Read the source opened by Connect / Edit

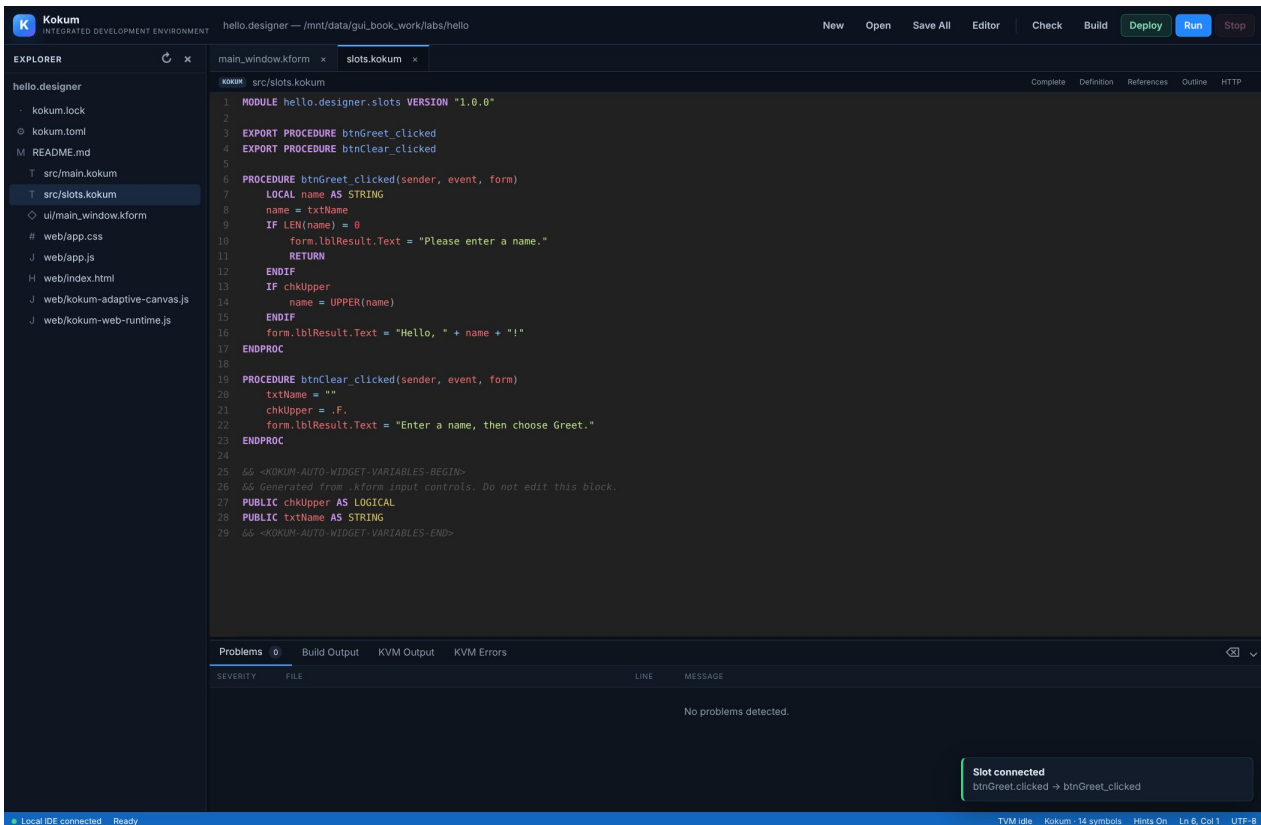


Figure 5.2 · The actual slots.kokum tab after connecting btnGreet.clicked. The existing body and generated declarations remain visible.

Three areas to confirm

At the top, MODULE names the slots module and EXPORT makes each bound procedure visible. In the body, the handler declares sender, event, form and contains your application logic. At the bottom, the marked managed block contains generated widget declarations. Each area has a different owner and a different reason to exist.

Open an existing handler through Events rather than creating a second similarly named procedure. Keep one implementation for the shared action. After a connection is changed, return to the form and verify the correct event row as well as the source.

A CONNECTED SOURCE IS NOT A RUNTIME TEST The status message confirms the designer operation. You must still Check, Build, Run and trigger the signal in the application. The Hello Designer acceptance table provides the expected results.

SOURCE BASIS [L1] Live native IDE Connect / Edit capture; [D2] binding contract.

CHAPTER 5 · ROUTINE STRUCTURE

Exports, parameters and helper procedures

MINIMUM HANDLER STRUCTURE — REFERENCE FRAGMENT

```
MODULE hello.designer.slots VERSION "1.0.0"

EXPORT PROCEDURE btnGreet_clicked

PROCEDURE btnGreet_clicked(sender, event, form)
    form.lblResult.Text = "Connected correctly"
ENDPROC
```

Requirement	Why it matters
Matching module	The source module must match the module named by the binding.
Explicit export	The public slot name must be exported, not only defined privately.
Local implementation	The export must resolve to a local callable implementation.
Three named parameters	The bound callable declares sender, event, form in that order.
Valid source	The complete slots module must parse and pass semantic validation.
Correct event target	The .kform must connect this control's event to this exported name.

Keep reusable logic separate

A helper such as RefreshCustomers(form) is an ordinary routine and may have its own parameter list. It is not a valid direct GUI slot because it has only one parameter. Call it from a three-parameter exported handler. This keeps business logic reusable without weakening the binding contract.

Calling btnGreet_clicked(sender, event, form) from another handler is a normal Kokum call. It does not manufacture a new browser event, and it retains the original arguments. A shared helper is often clearer when the source control should not matter.

CURRENT BOUNDARY Both exported procedures and compatible functions are understood by the binder. This book uses procedures for GUI actions and functions for returned computations or async work.

SOURCE BASIS [D2] Production slot validation; [L1–L8] compiled callable examples.

CHAPTER 5 · EVENT CONTEXT

Understand sender, event and form

Object	Use	Example
sender	The control that raised the current event. Inspect identity before applying sender-specific logic.	sender.Id, sender.Type
event	The native event envelope and supported cancellation field.	event.Name, event.Key, event.Cancel
form	The owning form/control map and form state.	form.lblStatus.Text, form.btnSave.Enabled

Native event fields in this edition

Fields	Type and meaning
Name / Kind	STRING canonical event name / INTEGER event kind.
Key / Text	STRING values supplied for relevant input events; otherwise commonly empty.
X / Y / Button	INTEGER pointer data, where applicable.
Modifiers / Timestamp	INTEGER modifier mask / millisecond timestamp.
Repeat / Cancel	LOGICAL repeat state / cancellation request, initially false.

Do not assume every event provides a populated value for every field. Read the selected item, checkbox value or TextBox contents from the current control or automatic variable. The portable native envelope does not promise generic event.Value, event.Data, event.OldValue or event.NewValue fields.

CANCELLATION FRAGMENT — REQUIRES THE CHAPTER 11 FORM

```
PROCEDURE DetailsPane_closing(sender, event, form)
  IF chkDirty
    event.Cancel = .T.
    form.lblStatus.Text = "Please finish editing first."
  ENDIF
ENDPROC
```

CANCELLATION IS SPECIFIC The closing route respects event.Cancel. It is not a general transaction rollback and does not cancel an HTTP request, database operation or arbitrary previously completed action.

SOURCE BASIS [S3] wf_build_event_map in src/tlang_web_form.c; [V1] SubForm cancellation test.

CHAPTER 5 · VALIDATION AND LIFETIME

Check a binding before debugging its body

FROM THE PROJECT DIRECTORY; TOOLS MUST BE ON PATH

```
kokumc form check ui/main_window.kform
kokumc form bind-check ui/main_window.kform src/slots.kokum
kokumc project check kokum.toml
```

Diagnostic family	Likely cause	Corrective action
TLF7101	Invalid or empty slots source.	Fix the full module syntax first.
TLF7102	Module mismatch.	Compare MODULE, slotsModule and target module.
TLF7103	Required target missing.	Create or reconnect the intended routine.
TLF7104	Callable kind/signature is incompatible.	Use exactly sender, event, form for the bound callable.
TLF7105	Target not exported.	Add the export or reconnect through Events.

Lifecycle events

loaded initializes a created form instance. closing is the point to allow or prevent disposal. closed is a completion notification after closing proceeds. Avoid running expensive database or network work on every high-frequency input event. Use an explicit action and visible status feedback instead.

Avoid event feedback loops

When a handler writes back into an input or selection, guard against repeating unnecessary work. Use a state flag or compare the new value to the intended value. Do not assume a browser-side event name implies a native callback payload or native method that has not been documented.

SMALL DIAGNOSTIC STEP First set a label to a constant string in the suspected handler. If it never changes, solve the binding/build/runtime path. Only after that succeeds should you debug validation, database calls or calculations.

SOURCE BASIS [D2] TLF binding diagnostics; [D3] persistent Web Form session.

CHAPTER 6

Automatic variables and control properties

Input controls generate named PUBLIC values in slots.kokum. The bridge supplies current input state before the slot runs and propagates supported changes back afterwards. This lets the handler read txtName directly without writing a textChanged handler just to copy its contents.

Designer control	Generated value	How to use it
TextBox / TextArea	STRING	Read or assign the text with the same control ID.
CheckBox / RadioButton	LOGICAL	Read checked state; assign .T. or .F.
ComboBox	STRING	The selected text value, not a numeric array index.
ListBox / ListView / TreeView	ARRAY	Selected values; use the control bridge for explicit selection metadata.
TableView	ARRAY	Row arrays for the table; keep ownership consistent when refreshing it.
Button / Label / Timer / containers	No ordinary input variable	Use form.controlId.Property for supported bridge properties.

GENERATED BLOCK FROM HELLO DESIGNER — READ, DO NOT HAND-MAINTAIN

```
&& <KOKUM-AUTO-WIDGET-VARIABLES-BEGIN>
&& Generated from .kform input controls. Do not edit this block.
PUBLIC chkUpper AS LOGICAL
PUBLIC txtName AS STRING
&& <KOKUM-AUTO-WIDGET-VARIABLES-END>
```

Choose the representation deliberately

txtName is a STRING value. form.txtName is a control bridge object with properties. form.lblResult.Text changes a label that has no ordinary input variable. The bare lblResult name is not a substitute for the label control.

NAME OWNERSHIP Managed names must remain unique across forms that share the module. Do not declare a separate PUBLIC variable with the same name or manually add entries to the generated block.

SOURCE BASIS [D4] Automatic variables; [L1] emitted declarations; [S3] state bridge.

CHAPTER 6 · STATE AND RESOURCES

Keep values on the correct session

Non-input component	Generated native value	Typical use
Image	IMAGE	<code>imgPhoto.SetImage(...), imgPhoto.Clear()</code> .
Graph	GRAPH	<code>graphSales.SetLabels(...), AddSeries(...)</code> .
SQLite / TigerDB / PostgreSQL	DATABASE	<code>dbDesk.Open(), Execute(...), QueryTable(...)</code> .
WebSocketClient	WEBSOCKET	Connect, send and poll queued network events on the owner session.

These generated native resources are different from ordinary input values. A source-level DATABASE or WEBSOCKET variable is not a visible widget, and a Timer is controlled through `form.tickTimer` rather than through an automatic TIMER variable.

Understand synchronization boundaries

State is maintained by the persistent module/session. Hiding a SubForm preserves its UI instance. Disposing its UI does not erase every PUBLIC value in the module; showing it again can restore the edited input values. Reset application state explicitly when that is the desired behavior.

Avoid writing both an automatic variable and a competing control property for the same data in one handler. Prefer a single owner. For database tables, use a consistent refresh helper that assigns the latest query result to the automatic table variable.

TABLE RECONCILIATION CAUTION In this edition, a subsequent unrelated slot can reapply a table's Rows state. The Customer Desk lab therefore re-queries at each action boundary, including Reset and empty-input validation. This is a tested refresh pattern, not a claim that the underlying reconciliation issue was fixed.

Background work must return data

Do not pass live form objects, images, database components or widget bridges to HTTP worker tasks. Pass plain request data and return a result. Update the GUI from a Timer or other owner-session callback after completion. Chapter 12 gives a complete tested example.

SOURCE BASIS [L1–L8] generated declarations; [S3] owner-session bridge; [V1] state and table observations.

CHAPTER 7

Choice controls and the Items editor

LAB SETUP Create a GUI project named “Choice Lab”. Use MainWindow, width 720, height 470, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
cmbRole · ComboBox	MainWindow	24, 24, 240, 36	Developer / Analyst / Tester
lstSkills · ListBox	MainWindow	24, 84, 240, 160	Kokum / SQL / Testing
chkActive · CheckBox	MainWindow	300, 28, 210, 30	Active
rdoBasic · RadioButton	MainWindow	300, 88, 160, 30	Basic
rdoExpert · RadioButton	MainWindow	300, 128, 160, 30	Expert
btnSummary · Button	MainWindow	300, 192, 170, 40	Read choices
lblSummary · Label	MainWindow	24, 280, 650, 100	Ready

Configure the choice data

Use the Items editor for cmbRole: Developer, Analyst and Tester on separate lines. Set selectedIndex to 0. For lstSkills use Kokum, SQL and Testing on separate lines and enable multiSelect. Give both RadioButtons the same group, experience; start Basic checked and Expert unchecked.

Sender ID	Signal / event	Exported slot
btnSummary	clicked	btnSummary_clicked

The caption “Basic” is not the radio group. Sharing the group is what makes the choices mutually exclusive. The automatic variables rdoBasic and rdoExpert report their checked values when the action runs.

SOURCE BASIS [L2] Choice Lab; [S1] ComboBox/ListBox/RadioButton schema.

CHAPTER 7 · READ THE CHOICES

A complete selection summary handler

KOKUM

```
MODULE choice.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE btnSummary_clicked

PROCEDURE btnSummary_clicked(sender, event, form)
  LOCAL level AS STRING
  level = "Basic"
  IF rdoExpert
    level = "Expert"
  ENDIF
  form.lblSummary.Text = cmbRole + " / " + level + ;
    " / active=" + STR(chkActive) + ;
    " / skills=" + JSONENCODE(lstSkills)
ENDPROC
```

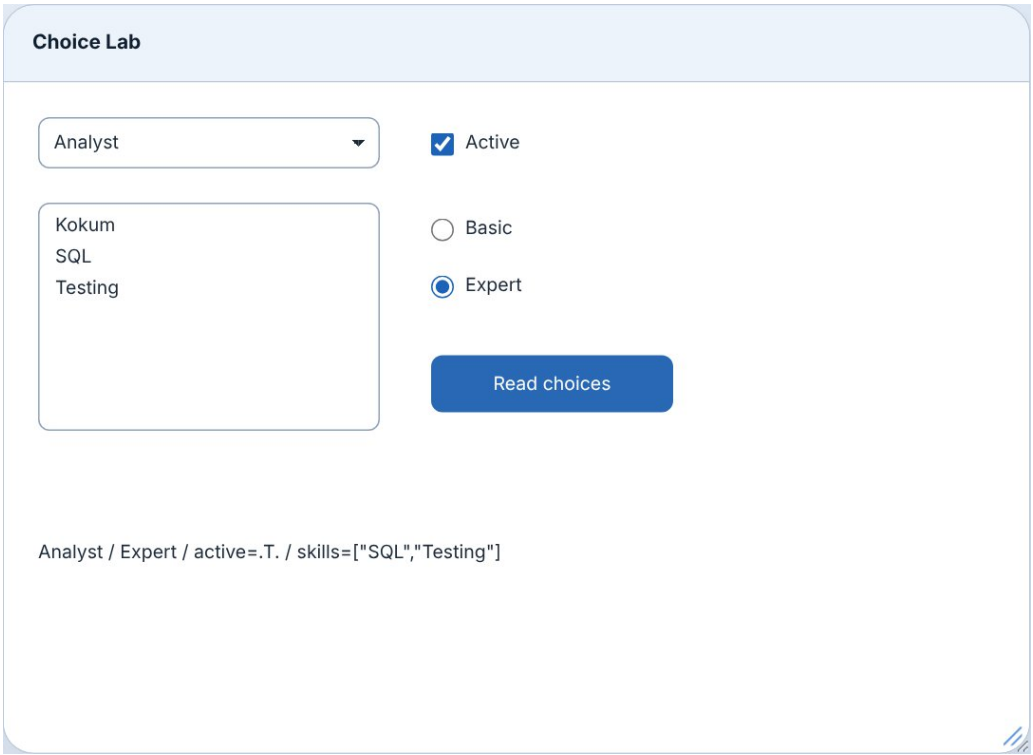


Figure 7.1 · Analyst and Expert selected, with SQL and Testing in the multi-selection.

INDEX CONVENTIONS A designer selectedIndex is zero-based. Kokum ARRAY indexing is one-based. Some native list methods also use one-based positions, while TableView row methods use zero-based positions. Do not transfer an index between these APIs without checking its contract.

SOURCE BASIS [V1] Choice Lab browser test; [S1/S3] selection contracts.

CHAPTER 7 · USABLE FORMS

Text, focus and accessible interaction

TextBox and TextArea

Use TextBox for a short value and TextArea for multi-line input. Placeholder is guidance, not actual text. Password display masks the on-screen value; it does not encrypt it or authorize the user. Keep private values out of printed diagnostics and screenshots.

Read-only is not disabled

Read-only inputs can display data without allowing edits. Disabled controls communicate that an action is currently unavailable. Use `Enabled = .F.` while an async action is outstanding and restore it after completion. Also guard in the handler: a visual state is not a substitute for application validation.

Keyboard and focus

Give controls a sensible `TabIndex` order and meaningful `accessibleName` values. Pair input fields with clear labels. Bind `enterPressed` only when Enter has an unambiguous action; in a multi-line editor, Enter may be part of the user's text.

Scenario	Recommended interaction
A required field is empty.	Keep the user's other data, show a specific nearby message and allow correction.
A slow task is running.	Disable duplicate submission, retain the task and show a status/progress indication.
A form contains many options.	Use groups, panels or tabs. Avoid a single unstructured sheet of controls.
The browser window is narrow.	Check minimum sizes and clipping in the actual runtime, not only at 100% design zoom.

PRACTICE Add a second required TextBox to Hello Designer. Bind both Enter Pressed signals to one action. Read both values in that handler, preserve them after validation failure, and verify keyboard-only use.

SOURCE BASIS [S1] Input/accessibility properties; [L1/L2/L8] tested interaction patterns.

CHAPTER 8

Responsive forms and container layouts

A form’s width and height are logical design dimensions. Browser presentation can adapt without changing those saved coordinates. This is separate from parent layouts, which arrange their child controls.

Setting	Purpose
sizeMode = fixed	Keep a fixed design-size presentation where that fits the application.
sizeMode = responsive	Use browser-relative viewport sizing and responsive presentation.
sizeMode = fullscreen	Use the available browser view as the application presentation.
viewportWidth / viewportHeight	Percentage-based viewport targets in responsive mode.
scaleContent / preserveAspectRatio	Control content scaling and whether the logical aspect ratio is retained.
allowUpscale	Control whether content may grow beyond its design scale.
theme	Choose midnight, dark, aqua, light, ocean, forest, sunset or graphite.

Choose the parent layout before positioning children

Layout	Good starting use
absolute	Precisely positioned data-entry dialogs and small teaching forms.
horizontal / vertical	Rows or stacks where padding and spacing should control placement.
dock	Application areas attached to top, bottom, left, right or the remaining fill space.

Do not combine manual positions with a layout that intentionally calculates them and then diagnose the result as a lost move. Use minimum sizes and sensible nesting. Test both small and large browser sizes with long captions and real data.

SOURCE BASIS [S1] Window and Panel schema; [D5] layout/runtime model.

CHAPTER 8 · CONTAINERS

Tabs, splitters, scroll views and docking

Control	Correct structure	Important setting
Panel	A named parent for ordinary children.	layout, padding and spacing.
TabView → TabPage	Put pages inside the TabView; put page content inside each TabPage.	The TabPage text is its header, not body content.
Splitter	Use two visual pane children. Put multiple controls inside a Panel in a pane.	orientation, position and minimumPaneSize.
ScrollView	Own the content that should scroll.	horizontalScrollBar, verticalScrollBar, scrollX and scrollY.
DockPanel	Group a dockable application area.	Use its layout and child dock properties deliberately.
ToolBar / StatusBar	Use as application command and status surfaces.	Confirm they belong to the intended container and do not overlap content.

Splitter example in words

For a navigation-and-detail layout, add a Splitter to the main content area. Add a narrow Panel as its first pane and a ScrollView as its second pane. Put navigation controls in the Panel and the long detail content inside the ScrollView. Set the splitter position and a minimum pane size, then test dragging the divider at runtime.

Scrolling depends on real content geometry

A ScrollView needs content extending beyond its visible area. A label at $y = 480$ inside a 330-pixel view is useful for a small test. A control placed beside the ScrollView in the hierarchy will never scroll with it.

NATIVE VS BROWSER HELPERS Use the supported Kokum properties such as `form.splitMain.Position` and `form.scrollDetail.ScrollY`. Browser JavaScript helpers named `SetSplitPosition` or `ScrollTo` are not a promise of identically named native slot methods in this edition.

SOURCE BASIS [S1] Container schemas; [S3/S4] native bridge versus browser contracts.

CHAPTER 8 · LAYOUT LAB

Check tab and scroll ownership

LAB SETUP Create a GUI project named “Layout Lab”. Use MainWindow, width 720, height 450, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

Create tabsMain with two child TabPages, pageDetails and pageSettings. Use Details and Settings as their text. In pageDetails place splitMain, whose children are panellList and scrollDetail. Put navigation controls in panellList and a low-positioned label in scrollDetail. In pageSettings use a vertical layout with its own setting control. No custom slot body is required for switching these tabs.

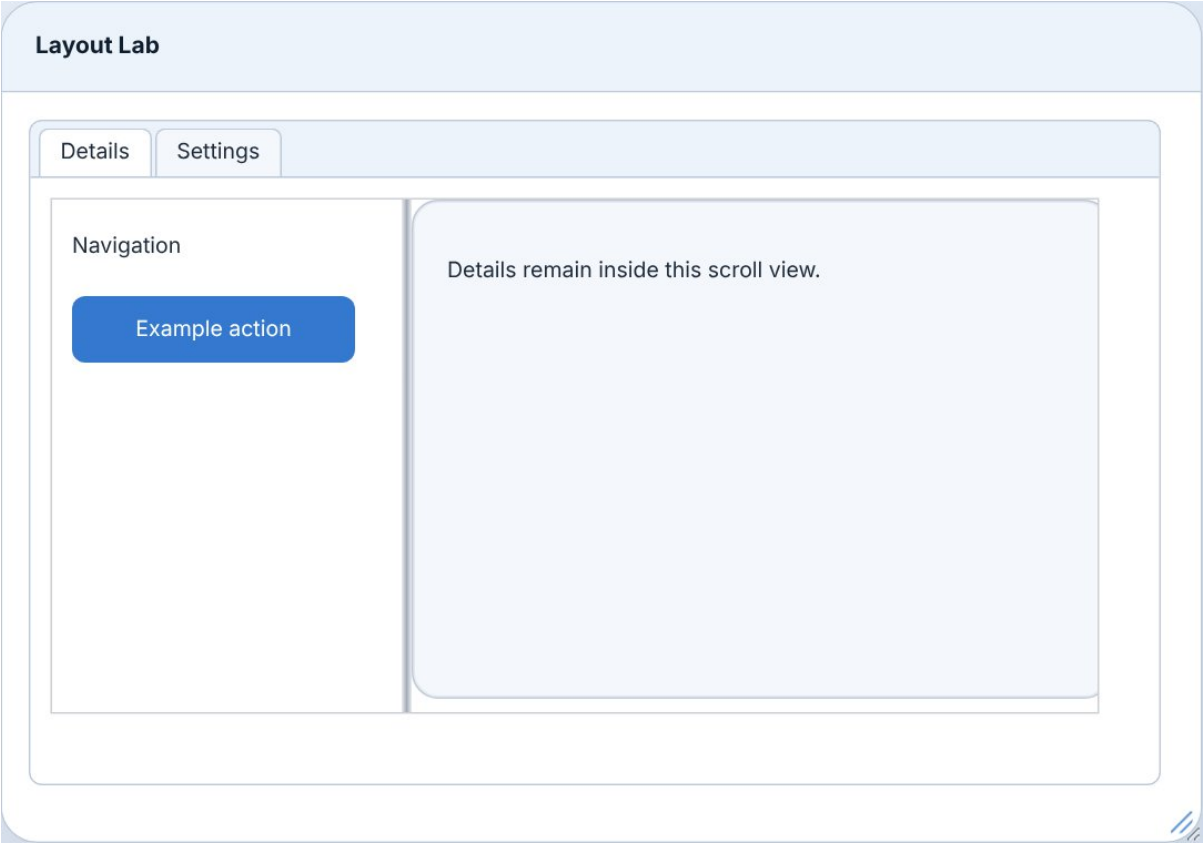


Figure 8.1 · Layout Lab running through KokumVM. The selected page owns its content.

Check	Expected observation
Switch to Settings.	Only that page's content is displayed.
Return to Details.	The two-pane layout remains intact.
Scroll the detail area.	Its child content moves; navigation remains in its own pane.
Resize the browser.	The layout remains usable within the chosen minimum sizes.

SOURCE BASIS [L9] Layout Lab; [V1] actual tab switch executed. Divider/resize acceptance remains a reader check.

CHAPTER 8 · EXACT LAYOUT RECIPE

Reproduce the nested control tree

ID · type	Parent	x, y, w, h	Caption / content
tabsMain · TabView	MainWindow	16, 16, 680, 400	
pageDetails · TabPage	tabsMain	0, 0, 660, 360	Details
splitMain · Splitter	pageDetails	12, 12, 630, 310	
panelList · Panel	splitMain	0, 0, 200, 300	
lblNavigation · Label	panelList	0, 0, 170, 34	Navigation
btnExample · Button	panelList	0, 0, 170, 40	Example action
scrollDetail · ScrollView	splitMain	0, 0, 420, 300	
lblLong · Label	scrollDetail	20, 20, 310, 44	Details remain inside this scroll view.
lblBottom · Label	scrollDetail	20, 480, 310, 44	Scroll down to reach this content.
pageSettings · TabPage	tabsMain	0, 0, 660, 360	Settings
chkRemember · CheckBox	pageSettings	0, 0, 240, 30	Remember this setting
lblHint · Label	pageSettings	0, 0, 480, 40	The page caption appears only in the tab header.

Use selectedIndex 0 on tabsMain. Both TabPages are 660 × 360. Set pageDetails layout absolute. Set pageSettings layout vertical, padding 20 and spacing 12. Set panelList layout vertical, padding 12 and spacing 12. For splitMain use orientation horizontal, position 210 and minimumPaneSize 100. Enable both scroll bars on scrollDetail; keep lblBottom at y = 480.

Preserve the wizard-generated MainWindow.loaded binding or connect it to the following small routine. The example button is deliberately unbound: this laboratory checks containment, not a business command.

KOKUM

```
MODULE layout.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE MainWindow_loaded

PROCEDURE MainWindow_loaded(sender, event, form)
    ? "Layout Lab ready"
ENDPROC
```

SOURCE BASIS [L9] Exact form hierarchy and compiled source used for Figure 8.1.

CHAPTER 9

Tables, trees and property grids

LAB SETUP Create a GUI project named “Data Widgets Lab”. Use MainWindow, width 720, height 470, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
treeTopics · TreeView	MainWindow	20, 20, 210, 280	Development / GUI Designer / Data access
tblPeople · TableView	MainWindow	248, 20, 430, 190	
gridOptions · PropertyGrid	MainWindow	248, 230, 430, 180	
lblSelection · Label	MainWindow	20, 320, 210, 72	Select a node

Tree hierarchy starts with indentation

TREEVIEW ITEMS — TWO LEADING SPACES ON EACH CHILD

```
Development
  GUI Designer
    Data access
```

Use the TreeView Items editor and preserve the leading spaces. Set selectedPath to the empty string initially. Do not substitute a selectedIndex property for a tree path. The tree’s automatic value is an ARRAY of selection data; SelectedPath is an explicit bridge property.

Sender ID	Signal / event	Exported slot
MainWindow	loaded	MainWindow_loaded
treeTopics	selectionChanged	treeTopics_selectionChanged

CURRENT METHOD BOUNDARY Native tests rejected TreeView.Clear(), AddItem() and Expand() as unavailable bridge members. Define this lab’s hierarchy in the Designer and read SelectedPath in the handler. Do not copy the richer browser-JavaScript method list into Kokum slots as though it were the native API.

SOURCE BASIS [L4] Data Widgets Lab; [S1/S3/S4] current contracts; [V1] native method boundary observation.

CHAPTER 9 · WORKING DATA CONTROLS

Initialize the table and property grid

Keep the generated `tblPeople` and `treeTopics` variables in the managed block. Use this complete programmer-owned module for the bindings on the preceding page. Table row method indices are zero-based, so `SetCell(1, ...)` edits the second row.

KOKUM

```
MODULE data.widgets.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE MainWindow_loaded
EXPORT PROCEDURE treeTopics_selectionChanged

PROCEDURE MainWindow_loaded(sender, event, form)

    form.tblPeople.ClearColumns()
    form.tblPeople.AddColumn("id", "ID", 70, "right")
    form.tblPeople.AddColumn("name", "Name", 210, "left")
    form.tblPeople.ClearRows()
    form.tblPeople.AddRow([1, "Anil"])
    form.tblPeople.AddRow([2, "Maya"])
    form.tblPeople.SetCell(1, "name", "Sara")

    form.gridOptions.Clear()
    form.gridOptions.AddProperty( ;
        "General", "title", "Title", "Customer view", "string")
    form.gridOptions.AddProperty( ;
        "General", "active", "Active", .T., "logical")
ENDPROC

PROCEDURE treeTopics_selectionChanged(sender, event, form)
    form.lblSelection.Text = "Path: " + form.treeTopics.SelectedPath
ENDPROC
```

The table has stable column keys `id` and `name`, separate from the visible headings `ID` and `Name`. The property grid has category, key, label, value and type for each entry. Keep stable keys even when labels change.

SOURCE BASIS [L4] Compiled and initialized through native KokumVM. The selection slot was compiled; interactive tree selection is a reader acceptance check.

CHAPTER 9 · DATA RESULTS

Use a single owner for displayed rows

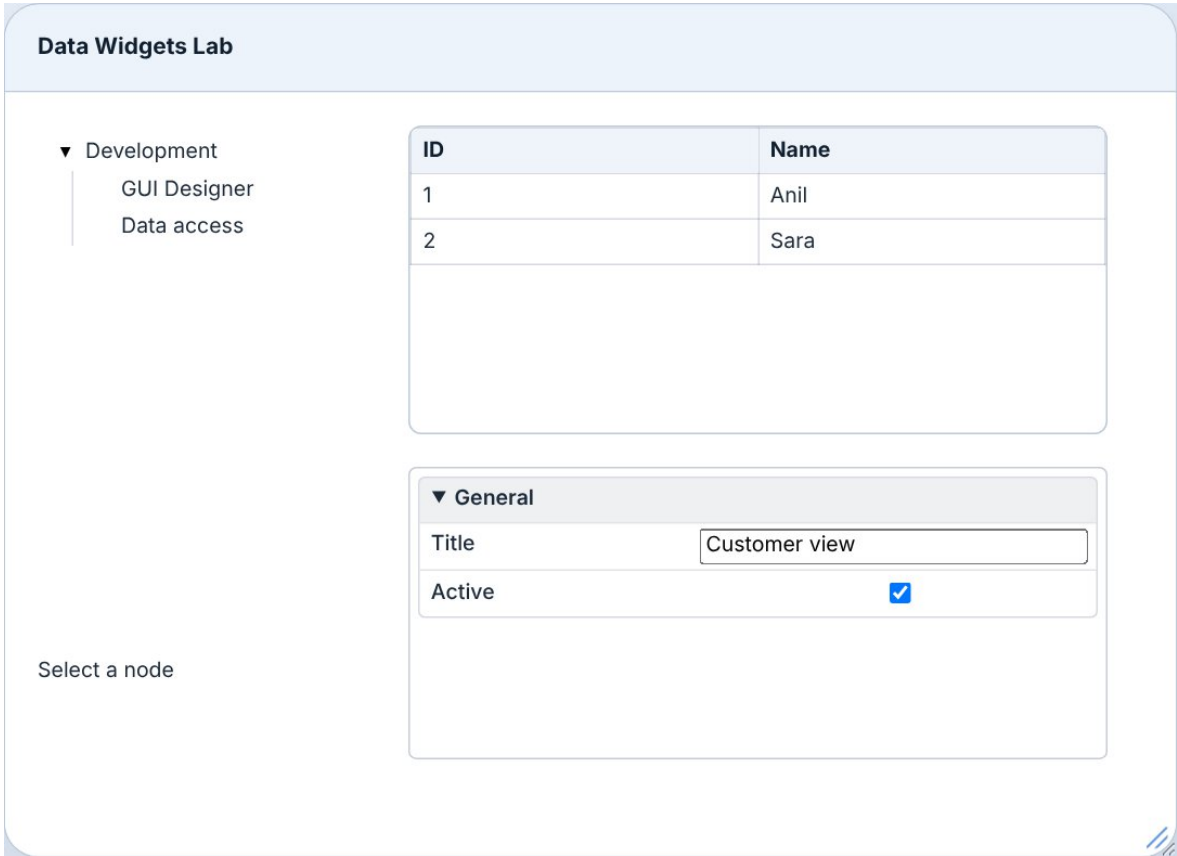


Figure 9.1 · Native table and property-grid initialization, with the tree built from indented Items.

The second table row is Sara because the loaded handler updates zero-based row 1. The property grid shows the title and logical Active setting. The tree comes from the form document rather than native AddItem calls.

Two table strategies

For small manually constructed tables, use the supported form.tblPeople methods consistently. For query results, assign the returned row ARRAY to the automatic table variable, as Customer Desk does. Mixing independent Rows properties, method mutations and automatic values makes synchronization harder to reason about.

REFRESH POLICY Rebuild the displayed query result after every relevant action. If a later event appears to empty a table while the database still contains rows, inspect automatic-variable/Rows reconciliation before assuming data loss. The capstone includes an explicit reload pattern for this edition.

SOURCE BASIS [L4/L7] native data examples and observed reconciliation boundary.

CHAPTER 10

Menus, toolbars and shared commands

Menu controls represent a hierarchy, not arbitrary floating buttons. Create MenuBar, then Menu children, then MenuItem children. Set the Menu text to File and a MenuItem text to Refresh chart. Bind the item's menuSelected signal, not the Button's clicked signal name.

VISUALS LAB HIERARCHY

```
MainWindow
  mainMenu : MenuBar
    menuFile : Menu      text = File
    miRefresh : MenuItem text = Refresh chart
  btnRefresh : Button
  graphSales : Graph
```

Control	Binding
miRefresh.menuSelected	btnRefresh_clicked
btnRefresh.clicked	btnRefresh_clicked
MainWindow.loaded	MainWindow_loaded, which calls the same update routine.

The menu item and button share a handler because both request the same chart refresh. The handler does not assume the sender is a Button. A more complex application can call a separate non-slot helper from both entry points.

Popup and toolbar surfaces

PopupMenu provides a menu surface; MenuItem provides the command event. ToolBar groups ToolButtons, which offer clicked and checkedChanged among their signals. Use the schema-supported relationships and inspect the hierarchy before diagnosing a menu that overlaps content.

ACCEPTANCE TEST Run the application, activate commands by pointer and keyboard, and verify there is one visible menu surface and one action per selection. This book's visual lab executes the loaded chart path; complete popup and toolbar interaction coverage is not claimed.

SOURCE BASIS [S1/S2] menu/toolbar schema and hierarchy; [L6] shared binding.

CHAPTER 10 · GRAPH LAB

Build a small chart from a slot

LAB SETUP Create a GUI project named “Visuals Lab”. Use MainWindow, width 620, height 410, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

Add graphSales at 24, 78, 560, 280; btnRefresh at 24, 24, 160, 38; and lblStatus at 200, 24, 380, 38. Use the menu hierarchy on the preceding page. Set the Graph type to bar and enable its legend. The managed graphSales value has type GRAPH.

KOKUM

```
MODULE visuals.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE MainWindow_loaded
EXPORT PROCEDURE btnRefresh_clicked

PROCEDURE MainWindow_loaded(sender, event, form)
    btnRefresh_clicked(sender, event, form)
ENDPROC

PROCEDURE btnRefresh_clicked(sender, event, form)
    graphSales.Clear()
    graphSales.SetType("bar")
    graphSales.SetTitle("Weekly enquiries")
    graphSales.SetLabels(["Mon", "Tue", "Wed", "Thu"])
    graphSales.AddSeries("Enquiries", [12, 18, 9, 22])
    graphSales.SetLegend(.T.)
    form.lblStatus.Text = "Chart refreshed"
ENDPROC
```

SetLegend is the native method used here; do not substitute a guessed name such as SetShowLegend. SetLabels provides category labels and AddSeries supplies matching values. Clear prevents duplicate series when the action is invoked repeatedly.

SOURCE BASIS [D6] Graph widget; [L6] compiled native chart example.

CHAPTER 10 · IMAGES AND RESOURCES

Display assets without hard-coded host paths



Figure 10.1 · The rendered Graph after the loaded handler sets its data.

An Image component is a native IMAGE value

Add an Image and set its source, size mode and alignment in Properties. The source path is resolved by the application host, not by an arbitrary file:// URL in the browser. For runtime changes use its generated native variable, such as `imgPhoto`. `SetImage` accepts a supported JPEG/PNG file; `Clear` restores the designed image state.

REFERENCE FRAGMENT — REQUIRES `imgPhoto` AND A REAL IMAGE FILE

```
imgPhoto.SetImage("assets/profile.png", .T.)
imgPhoto.Clear()
```

RESOURCE FRAGMENT FOR `kokum.toml`

```
[[resource]]
source = "assets/logo.png"
target = "web/images/logo.png"
```

With this resource target, use `images/logo.png` as the form's web-relative image source. Test the packaged artifact on a clean machine; a file visible only in your source directory is not automatically a deployment resource.

SOURCE BASIS [D7] Image widget/resource handling. Image fragments require an external file and are not among the executed labs.

CHAPTER 11

Subforms and controlled closing

LAB SETUP Create a GUI project named “Subform Lab”. Use MainWindow, width 720, height 470, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
btnShow · Button	MainWindow	24, 24, 150, 38	Show details
btnHide · Button	MainWindow	190, 24, 130, 38	Hide details
lblStatus · Label	MainWindow	24, 85, 600, 44	Ready
DetailsPane · SubForm	MainWindow	160, 150, 420, 225	Details
txtDetail · TextBox	DetailsPane	22, 24, 320, 36	Retained text
chkDirty · CheckBox	DetailsPane	22, 80, 260, 30	Prevent closing
btnDispose · Button	DetailsPane	22, 128, 160, 38	Dispose

Add DetailsPane as an inline SubForm directly under MainWindow. Set designerX = 810 and designerY = 30 to keep it beside the main design while editing. These are design-workspace coordinates. Runtime x = 160 and y = 150 are independent. Set closable and resizable true.

Sender ID	Signal / event	Exported slot
btnShow	clicked	btnShow_clicked
btnHide	clicked	btnHide_clicked
DetailsPane	loaded	DetailsPane_loaded
DetailsPane	closing	DetailsPane_closing
btnDispose	clicked	btnDispose_clicked

INITIAL VISIBILITY The SubForm is created for the runtime when SHOW DetailsPane is used. A design surface beside the main form is not proof that the SubForm should be visible at application startup.

SOURCE BASIS [D8] Inline SubForms; [L5] complete tested form.

CHAPTER 11 · SHOW, HIDE AND DISPOSE

Use lifecycle signals deliberately

KOKUM

```
MODULE subform.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE btnShow_clicked
EXPORT PROCEDURE btnHide_clicked
EXPORT PROCEDURE DetailsPane_loaded
EXPORT PROCEDURE DetailsPane_closing
EXPORT PROCEDURE btnDispose_clicked

PROCEDURE btnShow_clicked(sender, event, form)
    SHOW DetailsPane
ENDPROC

PROCEDURE btnHide_clicked(sender, event, form)
    HIDE DetailsPane
ENDPROC

PROCEDURE DetailsPane_loaded(sender, event, form)
    form.lblStatus.Text = "Details created"
ENDPROC

PROCEDURE DetailsPane_closing(sender, event, form)
    IF chkDirty
        event.Cancel = .T.
        form.lblStatus.Text = "Untick Prevent closing, then close again."
    ENDIF
ENDPROC

PROCEDURE btnDispose_clicked(sender, event, form)
    DISPOSE DetailsPane
ENDPROC
```

Command	Meaning in this workflow
SHOW DetailsPane	Show the pane, creating its runtime view when needed.
HIDE DetailsPane	Hide the view while retaining it.
DISPOSE DetailsPane	Request closing; respect Cancel; then dispose the view when allowed.

The loaded and closing events belong to DetailsPane. btnDispose is a child control, but its handler may request disposal by the SubForm’s stable ID.

SOURCE BASIS [L5] Full source; [V1] cancellation, disposal and re-show exercised.

CHAPTER 11 · LIFETIME AND MULTIPLE FORMS

UI disposal is not an application-state reset

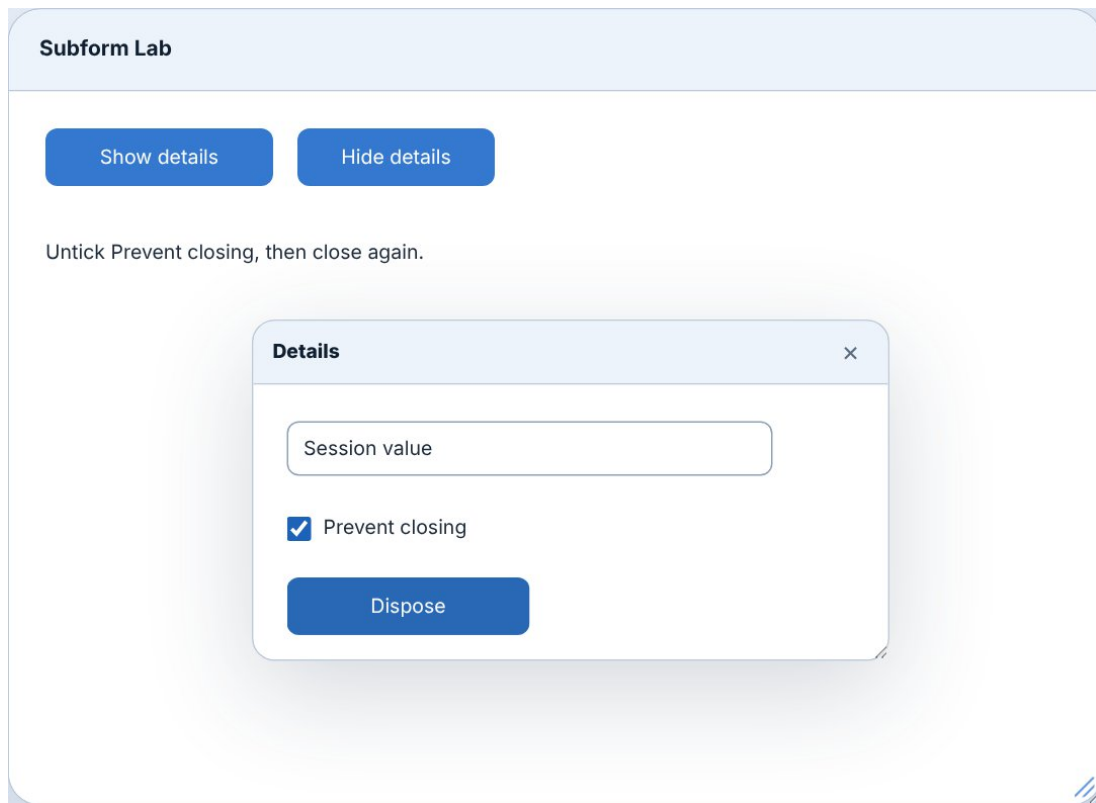


Figure 11.1 · Prevent closing is checked; the closing handler cancels disposal and leaves the pane visible.

Show the pane, edit txtDetail, then hide and show it. The text remains. Check Prevent closing and choose Dispose: the pane remains. Clear the checkbox and dispose it again: the view is removed. A later SHOW can still restore the module's automatic input value. Reset txtDetail explicitly when the application requires a fresh editor.

A separate form document

MANIFEST FRAGMENT — REQUIRES THAT FORM FILE

```
[[ form]]
name = "DetailsWindow"
source = "ui/details_window.kform"
slots = "src/slots.kokum"
```

A multi-document project lists every form explicitly. The form's declared slotsModule must match the linked module. Keep IDs unique when forms share a slots module and do not assume closing one view destroys every PUBLIC value or native resource in the session. The manifest fragment is a reference, not an additional executed lab.

SOURCE BASIS [D8] inline and multi-form lifecycle; [L5/V1] observed state retention.

CHAPTER 12

Timer components and responsive work

LAB SETUP Create a GUI project named “Timer Lab”. Use MainWindow, width 540, height 250, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
btnStart · Button	MainWindow	24, 24, 130, 38	Start
btnStop · Button	MainWindow	172, 24, 130, 38	Stop
lblTicks · Label	MainWindow	24, 100, 300, 38	Ticks: 0
progressWork · ProgressBar	MainWindow	24, 160, 460, 28	
tickTimer · Timer	MainWindow	tray	

The Timer appears in the Nonvisual Components tray rather than on the visible form. Set interval = 500 milliseconds, running = false and repeat = true. ProgressBar uses minimum 0, maximum 10 and value 0. Set the root loaded event and the three control signals below.

Sender ID	Signal / event	Exported slot
MainWindow	loaded	MainWindow_loaded
btnStart	clicked	btnStart_clicked
btnStop	clicked	btnStop_clicked
tickTimer	timerTick	tickTimer_timerTick

Do small work on each tick

A Timer callback runs on the GUI owner session. It is useful for polling completion, sampling state and making short UI updates. It is not an invitation to sleep or perform a long synchronous request inside timerTick. The next callback still depends on the session being available.

START / STOP Use form.tickTimer.Running to enable or disable this component. A Timer does not generate an ordinary input PUBLIC variable with its own ID.

SOURCE BASIS [L3] Timer Lab; [S1/S3] timer schema and bridge.

CHAPTER 12 · TIMER IMPLEMENTATION

Count ten ticks and stop

KOKUM

```
MODULE timer.lab.slots VERSION "1.0.0"

EXPORT PROCEDURE MainWindow_loaded
EXPORT PROCEDURE btnStart_clicked
EXPORT PROCEDURE btnStop_clicked
EXPORT PROCEDURE tickTimer_timerTick

PUBLIC tickCount AS INTEGER

PROCEDURE MainWindow_loaded(sender, event, form)
    tickCount = 0
ENDPROC

PROCEDURE btnStart_clicked(sender, event, form)
    tickCount = 0
    form.progressWork.Value = 0
    form.tickTimer.Running = .T.
ENDPROC

PROCEDURE btnStop_clicked(sender, event, form)
    form.tickTimer.Running = .F.
ENDPROC

PROCEDURE tickTimer_timerTick(sender, event, form)
    tickCount = tickCount + 1
    form.lblTicks.Text = "Ticks: " + STR(tickCount)
    form.progressWork.Value = tickCount
    IF tickCount >= 10
        form.tickTimer.Running = .F.
    ENDIF
ENDPROC
```

Start resets the count and progress. Each tick updates the status and progress, then disables the Timer at ten. Stop disables the Timer without restarting it. The expected stopping condition was observed through the real runtime.

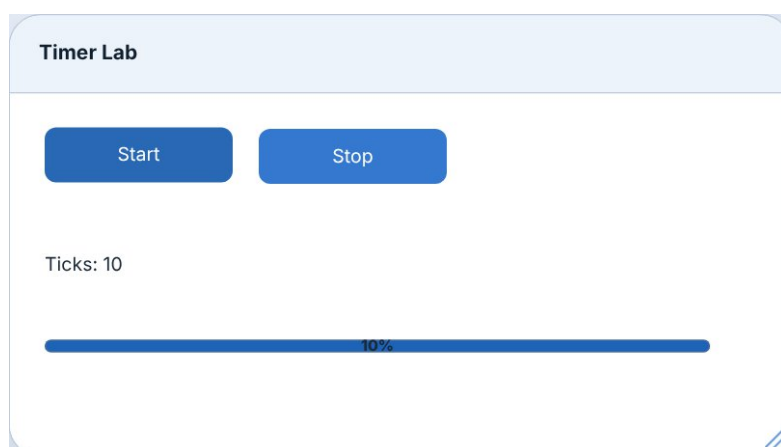


Figure 12.1 · Timer Lab at Ticks: 10 with the progress at its maximum.

CHAPTER 12 · ASYNC HTTP VIEWER

Return from the click handler promptly

LAB SETUP Create a GUI project named “HTTP Viewer”. Use MainWindow, width 680, height 430, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
btnFetch · Button	MainWindow	24, 24, 180, 38	Fetch local page
lblStatus · Label	MainWindow	224, 24, 410, 38	Ready
txtResponse · TextArea	MainWindow	24, 86, 610, 300	
pollTimer · Timer	MainWindow	tray	

Set pollTimer interval to 100 milliseconds, running true and repeat true. txtResponse is a TextArea. Bind btnFetch.clicked to btnFetch_clicked and pollTimer.timerTick to pollTimer_timerTick. The next page is the complete programmer-owned slots module.

START A LOCAL TEST ENDPOINT IN A SEPARATE TERMINAL

```
mkdir -p http-lab
printf 'Hello from the local GUI book service.' > http-lab/index.html
python3 -m http.server 18765 --bind 127.0.0.1 --directory http-lab
```

The task receives only the URL and returns the HTTP response MAP. The click handler retains the TASK and returns. The Timer checks IsCompleted and retrieves the result only after completion, then changes the TextArea and Label on the owning GUI session.

DO NOT BLOCK THE GUI AWAIT or Result() before completion blocks the calling session. A task-wait timeout does not cancel the HTTP operation. This example uses bounded response size and transport timeout, disables repeat submission and retains its task until completion.

SOURCE BASIS [L8] HTTP Viewer; [D11] URL async semantics. The endpoint is local and requires no external service.

CHAPTER 12 · COMPLETE ASYNC SOURCE

Poll the task, then update the form

KOKUM

```

MODULE http.viewer.slots VERSION "1.0.0"

EXPORT PROCEDURE btnFetch_clicked
EXPORT PROCEDURE pollTimer_timerTick

PUBLIC pending AS TASK

ASYNC FUNCTION FetchPage(url AS STRING) AS MAP
    RETURN URLGET(url, {"TimeoutMs": 3000, ;
        "MaxResponseBytes": 65536, "CookiePolicy": "omit"})
ENDFUNC

PROCEDURE btnFetch_clicked(sender, event, form)
    IF pending != NULL
        RETURN
    ENDIF
    TRY
        pending = FetchPage("http://127.0.0.1:18765/")
        form.btnFetch.Enabled = .F.
        form.lblStatus.Text = "Loading..."
    CATCH error
        form.lblStatus.Text = "Could not start request."
    ENDTRY
ENDPROC

PROCEDURE pollTimer_timerTick(sender, event, form)
    LOCAL response AS MAP
    IF pending = NULL
        RETURN
    ENDIF
    IF .NOT. pending.IsCompleted
        RETURN
    ENDIF
    TRY
        response = pending.Result()
        txtResponse = response["Body"]
        form.lblStatus.Text = "HTTP " + STR(response["Status"])
    CATCH error
        form.lblStatus.Text = "Request failed."
        ? STR(error)
    ENDTRY
    pending = NULL
    form.btnFetch.Enabled = .T.
ENDPROC

```

KokumVM supplies real worker-backed overlap. Live form objects and database/image/graph resources do not cross this task boundary. Closing a form while work is outstanding needs an application-specific lifetime policy; this listing is not a cancellation API.

SOURCE BASIS [V1] Local request returned HTTP 200 and the expected body; nine runtime message exchanges captured.

CHAPTER 12 · NONVISUAL SERVICES

Database and WebSocket components

Component	Designer work	Application work
SQLite	Set databaseFile, autoOpen and any parent-directory policy.	Open, run parameterized commands and refresh visible results.
TigerDB / PostgreSQL	Set the appropriate host, port, database, credentials and provider options.	Handle connection failure; keep credentials out of browser-owned logic.
WebSocketClient	Set endpoint and connection options in the nonvisual component.	Use its WEBSOCKET object and poll queued network events on the GUI owner session.
Timer	Set interval/running/repeat and bind timerTick.	Perform short polling work; do not block the UI with waiting loops.

WebSocketClient has no ordinary designer signal list in this schema. Network notifications are queued on the native object; they are not automatically mapped to invented designer events such as messageReceived. Use the documented WebSocket methods and a Timer-driven owner-session pattern.

The separate Network Programming manual covers provider configuration, remote FlatDB, WebSocket authentication, HTTP options and network deployment in detail. In a GUI, the additional rule is ownership: perform UI updates in the slot session and present generic failures to users while retaining useful diagnostics for the operator.

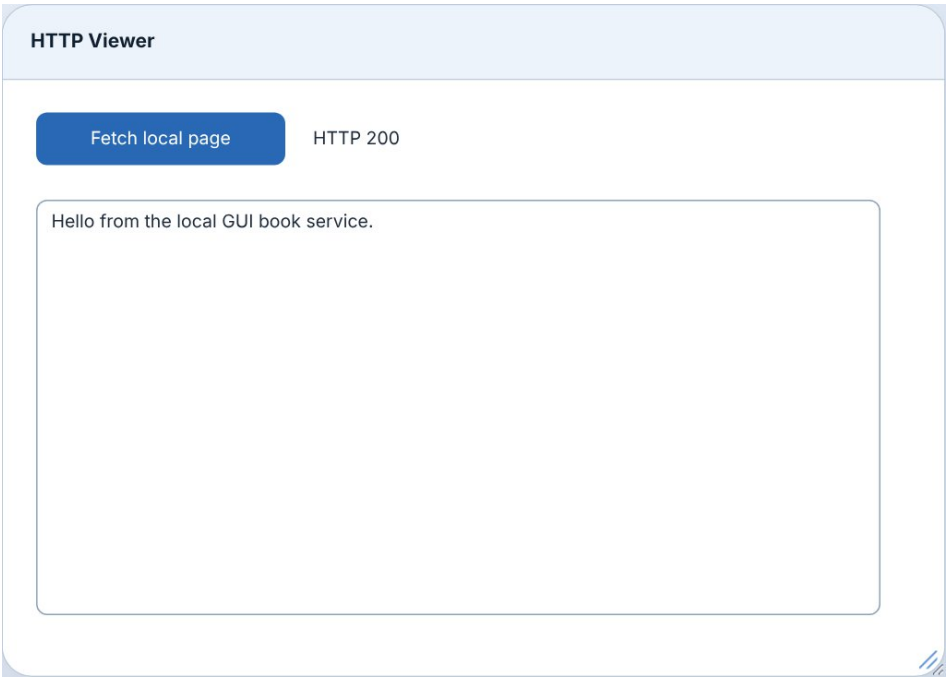


Figure 12.2 · HTTP Viewer after a successful local request. The click handler has restored the button.

SOURCE BASIS [D11] async network ownership; [D12] component references; [S1] zero WebSocketClient designer events.

CHAPTER 13

Customer Desk: a small complete application

This capstone combines a useful form, generated values, lifecycle slots, parameterized SQLite operations and a TableView. It intentionally avoids external servers so you can test the Designer and event flow independently of network configuration.

LAB SETUP Create a GUI project named “Customer Desk”. Use MainWindow, width 700, height 550, layout absolute and theme light. Examples use responsive sizing, viewportWidth 94, viewportHeight 90, scaleContent true, preserveAspectRatio true and allowUpscale false. Coordinates below are logical pixels, relative to the named parent.

ID · type	Parent	x, y, w, h	Caption / content
lblTitle · Label	MainWindow	24, 20, 600, 36	Customer Desk
lblName · Label	MainWindow	24, 76, 100, 30	Name
txtCustomer · TextBox	MainWindow	136, 72, 280, 36	Customer name
lblCity · Label	MainWindow	24, 128, 100, 30	City
cmbCity · ComboBox	MainWindow	136, 124, 280, 36	Ratnagiri / Pune / Mumbai
chkCustomerActive · CheckBox	MainWindow	444, 74, 175, 32	Active
btnSave · Button	MainWindow	24, 184, 126, 40	Save customer
btnReset · Button	MainWindow	164, 184, 116, 40	Reset
btnReload · Button	MainWindow	294, 184, 126, 40	Reload table
tblCustomers · TableView	MainWindow	24, 250, 632, 190	
lblStatus · Label	MainWindow	24, 462, 632, 54	Starting...
dbDesk · SQLite	MainWindow	tray	

Use Ratnagiri, Pune and Mumbai as the ComboBox Items; selectedIndex = 0. Start chkCustomerActive checked. Set TableView columns to ID, Name, City and Active on separate lines and keep headersVisible and gridVisible true. Enable wordWrap on lblStatus.

SOURCE BASIS [L7] Complete Customer Desk form and runtime test.

CHAPTER 13 · WIRING AND DATA LIFETIME

Create the database and connect actions

Sender ID	Signal / event	Exported slot
MainWindow	loaded	MainWindow_loaded
MainWindow	closed	MainWindow_closed
btnSave	clicked	btnSave_clicked
btnReset	clicked	btnReset_clicked
btnReload	clicked	btnReload_clicked

Safe repeatable database setup

For this lab, dbDesk is an SQLite component with databaseFile = :memory: and autoOpen = false. The loaded handler opens it and creates the table. Each application process starts with a new in-memory database; closing the process loses those demo records. No real customer data should be stored in this configuration.

Read the code as one module

The next two pages contain one complete programmer-owned slots module, split at a routine boundary for readability. Insert both parts in order. Keep the generated widget and database blocks produced by Check, including PUBLIC dbDesk AS DATABASE and the automatic input/table declarations. Do not duplicate MODULE or EXPORT statements.

A single refresh helper

RefreshCustomers(form) assigns QueryTable output to tblCustomers. The true third argument requests column headings, so the table displays ID, Name, City and Active. This helper is called after saving, reloading, resetting inputs and the empty-name validation branch.

DELIBERATE WORKAROUND The reload after Reset does not alter the database. It ensures the table is repopulated at the slot boundary in this runtime edition. If an unrelated action visually clears rows, re-query before concluding the data was deleted.

SOURCE BASIS [L7] SQLite native component and recorded reconciliation behavior; [D12] QueryTable usage.

CHAPTER 13 · SLOTS, PART 1

Open the database and save a customer

KOKUM

```

MODULE customer.desk.slots VERSION "1.0.0"

EXPORT PROCEDURE MainWindow_loaded
EXPORT PROCEDURE btnSave_clicked
EXPORT PROCEDURE btnReset_clicked
EXPORT PROCEDURE btnReload_clicked
EXPORT PROCEDURE MainWindow_closed

PROCEDURE MainWindow_loaded(sender, event, form)
  TRY
    dbDesk.Open()
    dbDesk.Execute( ;
      "CREATE TABLE IF NOT EXISTS customers (" + ;
      "id INTEGER PRIMARY KEY, name TEXT NOT NULL, " + ;
      "city TEXT, active INTEGER)"
    )
    RefreshCustomers(form)
  CATCH error
    form.lblStatus.Text = "Database unavailable."
    ? STR(error)
  ENDTRY
ENDPROC

PROCEDURE RefreshCustomers(form)
  tblCustomers = dbDesk.QueryTable( ;
    "SELECT id AS ID, name AS Name, city AS City, " + ;
    "active AS Active FROM customers ORDER BY id", [], .T.)
  form.lblStatus.Text = "Table refreshed."
ENDPROC

```

The loaded slot creates the table only if it does not already exist. RefreshCustomers is intentionally not exported as a GUI handler: it accepts only form and is called by the actual three-parameter slots.

Why bind parameters?

The Save implementation on the next page passes the name, city and active state in an ARRAY corresponding to the SQL placeholders. A name such as O'Neil is data, not part of SQL syntax. Never concatenate an input value into a command string to make quotation marks “work”.

SOURCE BASIS [L7] The runtime acceptance test includes a name containing an apostrophe.

CHAPTER 13 · SLOTS, PART 2

Save, reset, reload and close

KOKUM

```
PROCEDURE btnSave_clicked(sender, event, form)
  IF LEN(txtCustomer) = 0
    RefreshCustomers(form)
    form.lblStatus.Text = "Enter a customer name."
    RETURN
  ENDIF
  TRY
    dbDesk.Execute( ;
      "INSERT INTO customers(name, city, active) " + ;
      "VALUES (?, ?, ?)", ;
      [txtCustomer, cmbCity, chkCustomerActive])
    RefreshCustomers(form)
    txtCustomer = ""
    form.lblStatus.Text = "Customer saved."
  CATCH error
    form.lblStatus.Text = "Save failed. See KVM Output."
    ? STR(error)
  ENDTRY
ENDPROC
```

KOKUM

```
PROCEDURE btnReset_clicked(sender, event, form)
  txtCustomer = ""
  cmbCity = "Ratnagiri"
  chkCustomerActive = .T.
  RefreshCustomers(form)
  form.lblStatus.Text = "Input reset. No records deleted."
ENDPROC

PROCEDURE btnReload_clicked(sender, event, form)
  TRY
    RefreshCustomers(form)
  CATCH error
    form.lblStatus.Text = "Reload failed."
    ? STR(error)
  ENDTRY
ENDPROC

PROCEDURE MainWindow_closed(sender, event, form)
  dbDesk.Close()
ENDPROC
```

Reset changes only the input values and refreshes the view. It does not issue DELETE. The closed handler releases the database component after the form has finished closing.

CHAPTER 13 · ACCEPTANCE AND PERSISTENCE

Verify behavior before extending it

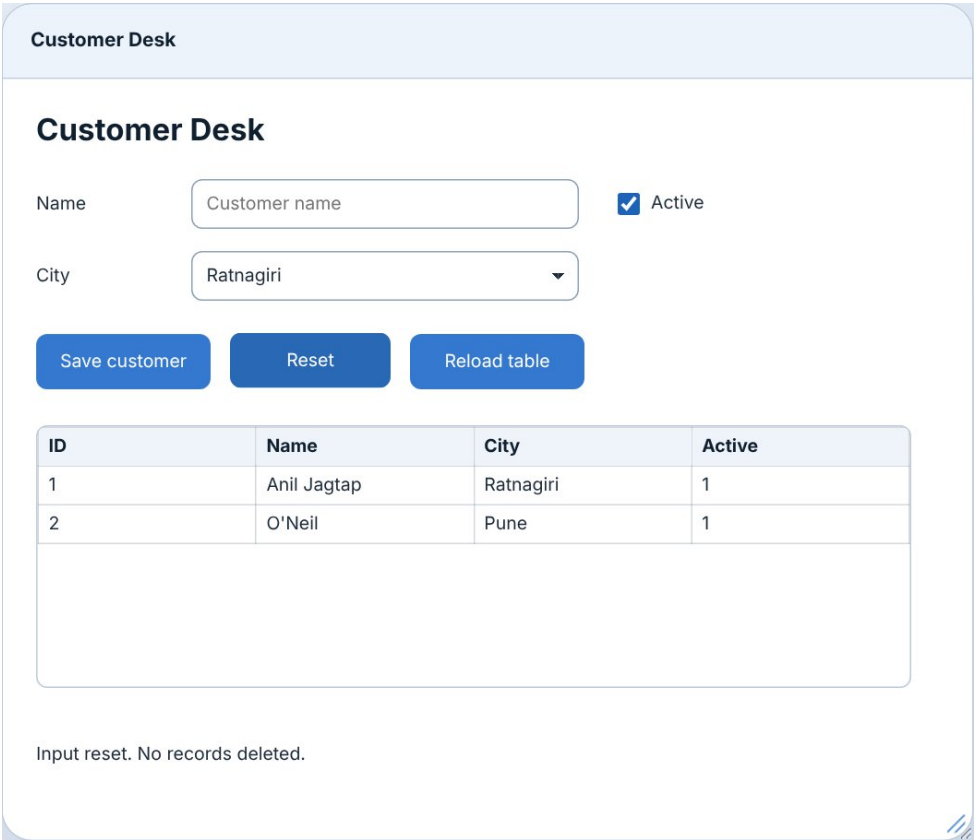


Figure 13.1 · Customer Desk after saving two rows and resetting inputs. The stored rows remain visible.

Acceptance step	Expected result
Save with an empty name.	Specific validation message; existing rows are preserved.
Save Anil Jagtap, then O'Neil.	Both rows appear; the apostrophe needs no manual SQL escaping.
Choose Reset.	Default inputs return; no SQL rows are deleted.
Choose Reload table.	Rows are obtained again from the current database.
Restart the :memory: application.	The demonstration database is empty again.

To explore persistence, change databaseFile to data/customers.db and enable createParentDirectory, or create the directory yourself. Treat the database as external writable application data, not as an embedded read-only .kapp resource. Back up that file and retest initialization, failures and deployment permissions. The persistent-file variant is an extension exercise, not an executed claim.

SOURCE BASIS [V1] Save, apostrophe, validation and Reset browser checks; [D12] SQLite configuration.

CHAPTER 14

Check, build, run and package

Use the IDE buttons for the normal loop and the command line for reproducible diagnostics. Commands below are run from the Hello Designer project directory with the Kokum programs on PATH. Otherwise replace each executable with its actual installed path.

PROJECT VALIDATION AND PORTABLE GUI APPLICATION

```
kokumc project check kokum.toml
kokumc form check ui/main_window.kform
kokumc form bind-check ui/main_window.kform src/slots.kokum

kokumc build kokum.toml
kokumc bundle kokum.toml
kokumvm dist/hello.designer.kapp
```

When inspecting the form compiler directly

FORM RESOURCE CHECK — NOT A REPLACEMENT FOR PROJECT BUNDLING

```
mkdir -p build
kokumc form compile ui/main_window.kform src/slots.kokum \
-o build/MainWindow.kres
kokumc form verify build/MainWindow.kres
```

A successful form compile validates bindings and creates the compiled resource. The application still needs its entry module, linked slots and web resources. Do not distribute a .kres alone and expect it to be a complete application.

IDE panel	Read it when
Problems	The form, module, export or source has a validation error.
Build Output	Compilation, linking, packaging or a resource step fails.
KVM Output	A handler prints progress or diagnostics using ?.
KVM Errors	The application raises a runtime failure or cannot execute a method.

SOURCE BASIS [L1] actual CLI/project outputs; [D2] form compile binding validation.

CHAPTER 14 · DISTRIBUTION

Portable application versus native executable

Artifact	What it contains	How to deploy
.kform / .kokum	Editable layout and source.	Developer inputs, not the normal end-user artifact.
.kres / .kbc	Compiled form resource / bytecode.	Low-level build products; need the surrounding application context.
.kapp	Bundled code, forms and declared resources.	Run with a compatible KokumVM on the target platform.
Standalone executable	Application plus a host-native embedded VM.	Build for the intended host; rebuild to receive runtime fixes.

LINUX HOST-NATIVE STANDALONE BUILD — REFERENCE COMMAND

```
kokumc build --standalone kokum.toml -o dist/hello-designer
```

The compiler and IDE in this release are Linux tools. Windows and macOS use the corresponding VM distribution for portable applications; this book does not claim their native GUI execution was tested. The MinGW OpenSSL hotfix improves the Windows build path but does not turn a Linux simulation into platform certification.

External files remain your responsibility

Package image and static web resources explicitly. Keep writable databases, network configuration and credentials outside the read-only application bundle when they must change at deployment. Test from a clean directory so a source-tree asset cannot hide a missing resource.

RELEASE CHECKPOINT Test every required binding, empty and long input, double submission, window resizing, tab order, failure messages, form closing and background-task lifetime. Then launch the exact artifact that will be delivered, not only the IDE's latest session.

SOURCE BASIS [D3] runtime/bundle model; [D11] portable VM limits; latest build-hotfix guide.

CHAPTER 14 · TROUBLESHOOTING

Find the failing boundary

Symptom	Check first	Avoid
Click does nothing.	Correct event bound, exported signature, rebuilt artifact, actual Run view.	Renaming routines at random or marking the handler optional.
Cannot connect slot / unresolved export.	Module identity, local implementation, 3 parameters, syntax errors.	Adding an export without an implementation.
Slots revision conflict.	Unsaved edits, external writer, current file path and revision.	Repeated force-save or deleting managed blocks.
Widget is missing or looks detached.	Hierarchy, parent, visibility, clipping, layout and selected page.	Assuming visual overlap means containment.
MAP has no callable member.	Whether the method exists on the native bridge, not only in JavaScript.	Guessing an API name from a browser helper.
Tree item selection fails.	Indented Items and SelectedPath; correct tree events.	Using selectedIndex as the tree's identity.
Table appears empty after another action.	Re-query, automatic table state and Rows reconciliation.	Assuming the SQL data was deleted without checking.
GUI freezes during a request.	Blocking URLGET/AWAIT or long work on owner callback.	Updating widgets directly from a worker.
Image works only on developer machine.	Resource declaration and deployed path.	Hard-coded source-host file paths.
CustomCanvas is absent.	Current public schema: it is retired.	Searching for a hidden supported toolbox switch.

KEEP A MINIMAL REPRODUCTION Reduce the failure to one form, one control and one handler. Record the exact signal, binding, diagnostic and runtime artifact. A compiler check and a browser interaction test answer different questions.

SOURCE BASIS [S1–S4] current boundaries; [V1] observed faults and tested workarounds.

APPENDIX A

Control catalogue · basic and input

The 37 entries below come from the current schema, not an old slide deck. Canonical event names are used. Common keyboard, pointer and focus events are summarized on the event-reference page; the Events inspector remains the authoritative list for a selected control.

Control	Typical designer use	Useful signal / note
Window	Main application root and theme/layout owner.	loaded, closing, closed
Label	Read-only captions, prompts and status.	clicked, doubleClicked where needed
Button	Explicit user command.	clicked
TextBox	Single-line input, placeholder, password display.	textChanged, enterPressed
TextArea	Multi-line input or returned text.	textChanged, enterPressed
CheckBox	Independent logical choice.	checkedChanged; automatic LOGICAL
RadioButton	One choice in a named group.	checkedChanged; same group for exclusivity
ComboBox	Choose one item from Items.	selectionChanged; automatic STRING
ListBox	Single or multiple list selections.	selectionChanged, doubleClicked
ProgressBar	Display bounded progress.	No designer signals; set Value.
Image	Display a local or bundled supported image.	Pointer signals; native IMAGE variable.
Timer	Periodic nonvisual callback.	timerTick; interval/running/repeat

Place only what the user needs

Prefer a clear Label and one action over many automatically firing handlers. Use a Timer only when there is actual periodic work or task completion to poll. Display validation beside the relevant input, and keep controls large enough to work at the chosen runtime scale.

SOURCE BASIS [S1] GUI schema 1.7; complete set continues on the next two pages.

APPENDIX A · CONTAINERS AND COMMANDS

Control catalogue · layout and navigation

Control	Typical designer use	Signal / structural guidance
Panel	Group controls under a layout.	No designer signals.
TabView	Own a set of selectable pages.	tabChanged, selectionChanging; children are TabPages.
TabPage	Own the body of one tab.	No signals; text supplies the tab heading.
Splitter	Divide two panes interactively.	splitterMoved; use pane wrappers.
ScrollView	Scrollable child-content viewport.	scrollChanged; keep content inside it.
DockPanel	A layout/docking area.	No designer signals.
MenuBar	Top-level menu surface.	No signals; contains Menu entries.
Menu	Group related menu commands.	No signals; contains MenuItem.
MenuItem	An individual command.	menuSelected, command.
PopupMenu	Contextual menu surface.	No own signals; command items supply actions.
ToolBar	Group compact application commands.	No own signals.
ToolButton	Toolbar command or toggle.	clicked, checkedChanged, command.
StatusBar	Persistent application status area.	No designer signals.
ModalDialog	A dialog with lifecycle behavior.	loaded, closing, closed.
SubForm	Inline secondary form with show/hide/dispose.	loaded, closing, closed.
SubFormHost	Legacy host compatibility entry.	Hidden from normal new-design toolbox; prefer SubForm.

COMPATIBILITY IS NOT A RECOMMENDATION SubFormHost contributes to the schema count but is not the preferred way to start a new inline form. CustomCanvas is not an extra current control.

SOURCE BASIS [S1] schema; [D8] current inline SubForm model.

APPENDIX A · DATA AND SERVICES

Control catalogue · structured and nonvisual

Control	Typical designer use	Signal / native object
TreeView	Hierarchical navigation from indented Items.	selectionChanged, itemActivated, itemExpanded, itemCollapsed; use SelectedPath.
ListView	Structured list selection.	selectionChanged, itemActivated, doubleClicked.
TableView	Tabular data and results.	selectionChanged, itemActivated, columnResized; ARRAY value and supported table methods.
PropertyGrid	Named editable/display properties.	selectionChanged, itemActivated; supported native grid methods.
SQLite	Local database component in the tray.	No designer events; DATABASE object.
TigerDB	TigerDB provider component in the tray.	No designer events; DATABASE object.
PostgreSQL	PostgreSQL provider component in the tray.	No designer events; DATABASE object.
Graph	Native chart/data visualization.	Native GRAPH object; common focus/key/context events.
WebSocketClient	Native connection component in the tray.	No designer events; WEBSOCKET object and queued notifications.

The count

Basic/input: 12. Containers/commands: 16. Structured data/services: 9. Total: 37 schema entries, including the hidden legacy SubFormHost. Capabilities described here do not imply that every possible interaction was tested; Appendix D lists the actual teaching-lab coverage.

A component is not a listener

Placing a database or WebSocket component supplies configuration and a native object. It does not create a network server, automatically bind arbitrary incoming notifications, or make a long operation GUI-safe. Use the component's documented methods and the owner-session rules.

SOURCE BASIS [S1] schema dump; [D12] native data/network component contracts.

APPENDIX B

Signal and property quick reference

Purpose	Canonical event names / properties
Form lifecycle	loaded, closing, closed
Command activation	clicked, doubleClicked; menuSelected for menu commands
Text entry	textChanged, enterPressed
Logical inputs	checkedChanged
Selection	selectionChanged; selectionChanging only on controls declaring it
Tabs and structures	tabChanged, itemActivated, itemExpanded, itemCollapsed, columnResized
View changes	splitterMoved, scrollChanged
Keyboard / focus	keyDown, keyUp, focusReceived, focusLost
Pointer / context	mouseDown, mouseUp, contextMenu; mouseMove is declared for Splitter
Timer	timerTick
Common state	form.controlId.Visible, Enabled, Text where supported
Selected / structural state	SelectedPath for TreeView; Position for Splitter; ScrollX/ScrollY for ScrollView
Owner event envelope	Name, Kind, Key, Text, X, Y, Button, Modifiers, Timestamp, Repeat, Cancel

Not every event or property belongs to every widget. For example, a Timer has timerTick but no clicked; WebSocketClient has no ordinary designer events. Use Events and Properties after selecting the actual type.

PAYLOAD RULE Current values live in the generated variables and control bridges. The native envelope does not guarantee event.Value, event.Data or an old/new-value pair. Build behavior from fields that the runtime actually supplies.

SOURCE BASIS [S1] per-type events; [S3] native event map.

APPENDIX B · NATIVE METHODS USED HERE

Do not confuse object families

Object family	Methods / operations demonstrated	Index / ownership note
TableView bridge	ClearColumns, AddColumn, ClearRows, AddRow, SetCell.	Table method row indices are zero-based.
PropertyGrid bridge	Clear, AddProperty.	Category, key, label, value, type arguments.
GRAPH value	Clear, SetType, SetTitle, SetLabels, AddSeries, SetLegend.	Plain labels/data; owner-session native object.
IMAGE value	SetImage, Clear (reference only).	Requires a real image file; Clear restores designed state.
DATABASE value	Open, Execute, QueryTable, Close.	SQL placeholders and value ARRAY, not string concatenation.
TASK value	IsCompleted, Result after completion.	Retain the handle; retrieving it does not issue another request.
Form lifecycle	SHOW, HIDE, DISPOSE.	Target a known form/SubForm ID.
Timer bridge	Running property.	Use form.tickTimer.Running, not a guessed native TIMER method.

The browser renderer can expose convenience methods that are not callable members of the native Kokum form bridge. This was observed for `TreeView.Clear/AddItem/Expand`. Treat completion and the native contract as evidence, not a reason to assume every JavaScript name exists in Kokum.

Useful supporting built-ins in the labs

LEN checks the string length; UPPER changes case; STR produces display text; JSONENCODE displays selection arrays; URLGET returns an HTTP response MAP. The actual request names are URLGET and URLPOST, not HTTPGET and HTTPPOST.

SOURCE BASIS [S3/S4] native method dispatch; [L1-L8] exercised source.

APPENDIX C

Practice tasks and acceptance criteria

Task	Build it	Prove it
1 · One action, two signals	Let a button click and TextBox Enter share a handler.	One export, two bindings; both update the same label.
2 · Validation without erasing	Add a second required field to Hello Designer.	Invalid input preserves all other values.
3 · A settings page	Put Choice Lab controls on a second TabPage.	Tab switching hides the other page, and settings remain intact.
4 · A cancellable close	Use a logical dirty flag in SubForm closing.	Cancel keeps it open; clearing the flag allows disposal.
5 · A useful busy state	Extend HTTP Viewer to show elapsed progress text.	The click handler returns, no duplicate task is submitted, and the button is restored.
6 · Persistent Customer Desk	Switch from :memory: to an external SQLite file.	Restart retains rows; Reset still does not delete them.
7 · Packaging audit	Add a real local image as a declared resource.	The exact .kapp works from a clean deployment directory.

A disciplined solution approach

Do not change layout, bindings and database behavior all at once. Keep a working checkpoint. Make one change, inspect its form metadata or source, run Check, then exercise the real event. Record the expected outcome before pressing Run.

EXTENSION SCOPE These tasks are exercises beyond the nine executed teaching projects. Their success must be demonstrated on your edited project and deployment environment.

APPENDIX C · SOLUTIONS IN OUTLINE

Reason about the event boundary

Tasks 1–2: shared action and validation

Connect both signals through Events to one exported routine with sender, event, form. Read the generated input variables once at entry. Validate each required value before starting a side effect. Return after showing a specific message, but do not clear input until the action succeeds.

Task 3: real containment

Create TabView, then TabPage, then move the controls under that page in Hierarchy. A visual overlap is not containment. Confirm selected-page behavior in the runtime. Keep the form's generated variables in the same owning module so ordinary tab changes do not reinitialize them.

Task 4: explicit state

Use a PUBLIC logical dirty flag outside the managed block, or the demonstrated checkbox variable. In closing set event.Cancel = .T. only while dirty. HIDE is not a save/dispose operation; reset or persist the state explicitly when the action calls for it.

Task 5: async completion

The worker performs URL work and returns a MAP. The owner retains the TASK and polls IsCompleted. On completion, handle Result in TRY/CATCH, clear the pending variable and restore the button. A transport failure is not permission to blindly replay a POST; the server may already have applied it.

Tasks 6–7: deployment ownership

A persistent database belongs in a writable external directory with an explicit backup policy. An image required by the GUI belongs in the declared resources or in a deliberately managed external asset location. Test from outside the source tree. Rebuild a standalone application when its embedded VM changes.

REVIEW QUESTION Why can a procedure compile correctly but never run when a user clicks? Because compilation of the routine is only one boundary: the form must contain the correct binding, the callable must be exported and compatible, and the executed artifact must include the new form and source.

SOURCE BASIS [D2/D3/D4/D11] event, state and deployment contracts.

APPENDIX D

What was checked for this edition

The nine projects were created from the current Web GUI scaffold, their form documents and slots were checked, and their application bundles were built. Each was then launched with the native Linux KokumVM and exercised in Chromium through the actual Web Form bridge. Source listings reconstructed from this DOCX also passed project checks, binding validation and bundling for all nine projects.

Teaching project	Executed evidence
Hello Designer	Empty input, uppercase greeting, Clear and Enter Pressed.
Choice Lab	Selected role, radio group and multi-selection shown in the summary.
Timer Lab	Progress reached ten, stopped and remained at ten.
Data Widgets Lab	Native loaded handler created table rows and property-grid entries; tree layout came from Items.
Subform Lab	Show/hide, edited-state retention, cancelled disposal, allowed disposal and re-show.
Visuals Lab	Native loaded handler populated the Graph and rendered the chart.
Customer Desk	Open/create, empty-name validation, two parameterized saves including an apostrophe, Reset retaining rows.
HTTP Viewer	Local GET completed through TASK polling, updated the text/status and restored the button.
Layout Lab	Real tab switch showed the corresponding page.

All nine runtime tests passed without captured browser JavaScript errors or native stderr output. The browser used a local Python transport relay to the real VM because the test environment restricts direct loopback navigation; no VM event responses were fabricated.

NO IMPLIED COVERAGE These checks are not an exhaustive widget suite. Tree selection, full menu/popup/toolbar interaction, divider dragging, keyboard accessibility, persistent-file variants, external providers and native Windows/macOS execution require their own acceptance tests.

SOURCE BASIS [V1] lab_build_results.json, browser_results.json and native request/reply captures retained during preparation.

APPENDIX D · LIMITATIONS AND SAFEGUARDS

Keep the claims tied to the evidence

Designer capture scope

Screenshots were captured from the native IDE and actual runtime applications. The Hello binding was connected through the Events interface and its source opened. The capture harness encountered a CodeMirror overlay-initialization warning while bringing up the IDE; the interface continued, but the book does not claim a complete warning-free IDE regression suite.

Implementation boundaries found during preparation

Boundary	How this book handles it
TreeView native mutation helpers unavailable	Use Designer Items and native SelectedPath; do not advertise the browser helper API as Kokum methods.
Table row reconciliation across slot actions	Customer Desk re-queries at each action boundary. The manuscript identifies this as a pattern/workaround, not a production fix.
CustomCanvas retired; legacy SubFormHost	No CustomCanvas tutorial; compatibility entry is identified clearly.
Live resources and worker ownership	Async requests receive plain data; UI mutation occurs only in owner callbacks.
External resources / platform certification	Reference fragments declare prerequisites; Linux tests are not Windows/macOS certification.

Reproduce a failure responsibly

Record the exact source revision, project manifest, control ID, signal target and compiled artifact. Keep a minimal form and the shortest handler reproducing the issue. Do not remove authentication, TLS verification or runtime validation merely to obtain a screenshot.

The book edits and documents teaching projects; it does not modify the production runtime to hide observed behavior. Code remains editable in this DOCX, and complete module listings are kept in source order even when continued across pages.

APPENDIX E

Source map and edition record

Primary basis: Kokum_0_29_0_Phase4_17_7_1_MinGW_OpenSSL_Fix.zip. Paths below are relative to its complete source root. Current schema, native implementation and executable observations take precedence over historical prose when they differ.

ID	Primary source / preparation evidence
D1	docs/BROWSER_VISUAL_FORM_DESIGNER.md
D2	docs/SLOT_BINDINGS.md
D3	docs/WEB_FORM_RUNTIME.md; docs/WEB_GUI_PROJECT_MANIFEST.md
D4	docs/AUTOMATIC_WIDGET_VARIABLES.md
D5	docs/GUI_LAYOUT_MODEL.md; docs/GUI_CONTROL_SCHEMA.md
D6 / D7	docs/GRAPH_WIDGET.md; docs/IMAGE_WIDGET.md
D8	docs/INLINE_SUBFORMS.md; docs/MULTI_FORM_WEB_APPLICATIONS.md
D9	docs/PHASE4_7_INTERNAL_SLOTS_SAVE_RECONCILIATION.md; docs/PHASE4_12_3_MANAGED_SLOTS_PATH_IDENTITY.md
D10	docs/PHASE4_16_7_TEXT_COLOR_PROPERTY_INSPECTOR_TABPAGE_REFINEMENT.md
D11	docs/URL_ASYNC_MICROSERVICES_IDE.md; docs/URL_HTTP_CLIENT.md
D12	docs/DATABASE_COMPONENTS.md; docs/WEBSOCKET_CLIENT.md
S1	src/tlang_gui_schema.c and executed kokumc form schema output
S2	web/ide/index.html; web/ide/kokum-form-designer.js
S3	src/tlang_web_form.c; src/kokum_form_control_internal.h
S4	web/shared/kokum-advanced-widget-contract.json; tests/phase4_15_1_2_12/test_advanced_widget_contract.py
L1-L9 / V1	Nine purpose-built teaching projects, compiler/bundle results, actual browser/VM captures and screenshots prepared for this book.

L1 Hello Designer · L2 Choice Lab · L3 Timer Lab · L4 Data Widgets Lab · L5 Subform Lab · L6 Visuals Lab · L7 Customer Desk · L8 HTTP Viewer · L9 Layout Lab. The listings and design instructions are contained in this book; no separate teaching-project download is required to follow them.