

Tutorial 10

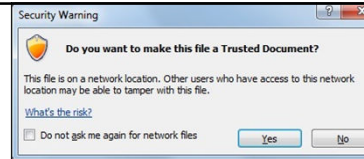
Automating Tasks with Macros

Start File: Tutorial 10 WebCast Practice.accdb

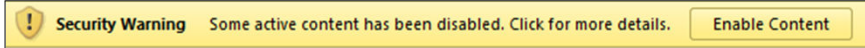
Introduction to Macros

- Macro
 - a list of **actions** you want Access to carry out
 - used to automate repetitive or routine tasks
 - e.g.: clicking a command button runs a 3-action macro that
 - opens a report
 - filters data for the current month
 - automatically prints two copies of the report
 - Tutorial 6: user makes a selection from a form's navigation combo box, causing the form to display the desired record
- Can also use Visual Basic programming code to automate a series of tasks (CIS217)
 - more powerful and flexible than macros
 - Tutorial 11

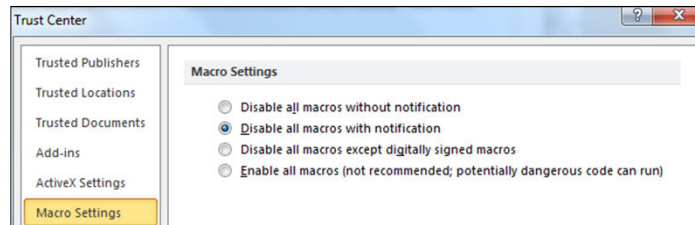
Macro Security



- Macros/Visual Basic code/Action Queries can be **malicious**
 - for safety, macros and VB code can be disabled, optionally with a warning



- **Practice Time:** Adjust macro security settings as illustrated
 - File | Options | Trust Center | Trust Center Settings...



- Access will execute macros/VB code/action queries when a file is opened from a Trusted Location, is a Trusted Document, or you click **Enable Content**.

3

Running a Macro

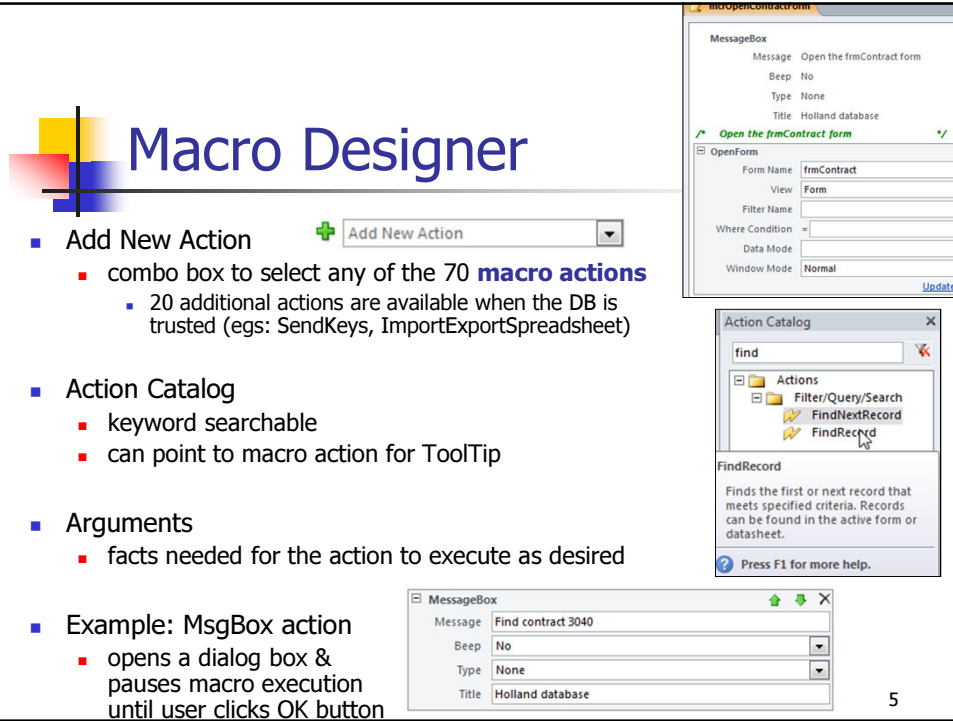


- Macro **actions** used in Tutorial 10:
 - MsgBox
 - OpenForm
 - Beep
 - GoToRecord
 - FindRecord
 - CloseWindow
 - RunMenuCommand
 - to execute an Access command from a macro
 - egs: FilterBySelection, DeleteRecord, PrintSelection, Redo, Undo
 - SelectObject
 - GoToControl
 - SendKeys
 - OpenQuery

4

Macro Designer


- Add New Action
 - Add New Action
 - combo box to select any of the 70 **macro actions**
 - 20 additional actions are available when the DB is trusted (egs: SendKeys, ImportExportSpreadsheet)
- Action Catalog
 - keyword searchable
 - can point to macro action for ToolTip
- Arguments
 - facts needed for the action to execute as desired
- Example: MsgBox action
 - opens a dialog box & pauses macro execution until user clicks OK button



5

When to Use Macros

- Early in the chapter you **directly** ran a macro from the Navigation Pane
 - not realistic!
- Macros are commonly engineered to run when
 - user makes a selection from a combo box (Tut6, Tut10)
 - user opens a form or report (Tut10)
 - user clicks a command button on a form (Tut10)
 - user navigates to a different record on a form (Tut10, GTour)
 - user saves a new/modified record (Tut 10)
 - user closes a form or report (GTour)
 - user opens the database (AutoExec macro, nib)
 - user presses a keystroke combination (AutoKeys macro, nib)
 - user makes a selection from a custom menu or shortcut menu (nib)



6

Event

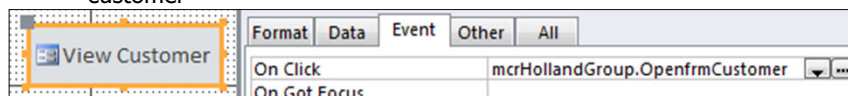
- A state, condition, or occurrence associated with an object for which you can define a response
 - most events occur as a result of a user action
 - e.g.: a mouse click, a key press, navigate to another record
- Events provide an opportunity for your application to do something
 - run a macro or Visual Basic procedure in response to an event
- Access has hundreds of events

Object	Event	Occurs When
Command Button	Click	
Form	Load	
Form	Current	
Form	Close	
Form	BeforeUpdate	
Combo Box	AfterUpdate	

7

Using an Event to Trigger a Macro

- Event Property
 - defines an object's response when the event occurs
 - each form, report, section, and control has **event properties** which determine what happens when the **event** occurs
 - by using a macro's name as the value for the event property, the macro runs whenever the event occurs
- Demo
 - Holland.accdb's frmContract has command button to view the contract's customer



- the command button's **Click** event calls a macro named **mcrHollandGroup.OpenfrmCustomer** which executes a series of actions that opens frmCustomer and displays the Contract's Customer

8

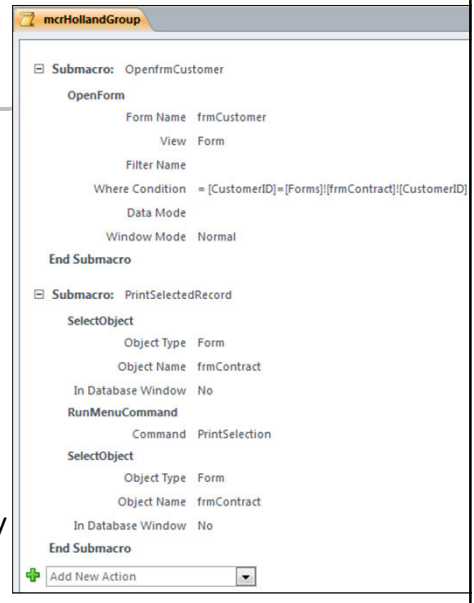


Macro Sheet

- Macro Group
 - a set of related submacros stored together
 - each submacro has its own name so it can be run separately

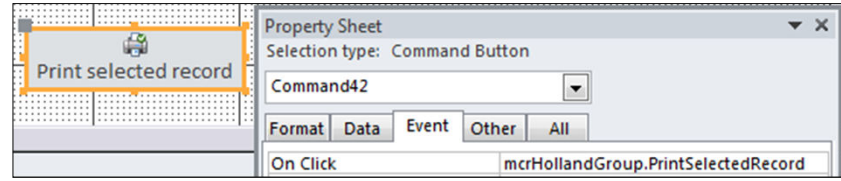
- Submacro
 - a complete macro with a submacro header

- Call one of the group's submacros by
 - **groupname.submacroname**
 - `mcrHollandGroup.OpenfrmCustomer`



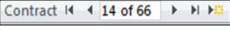
Add a Command Button to a Form Attach a Submacro to a Command Button

- Wanted a command button that prints the current record on the Contract form
 - an alternative to File | Print | Selected Record(s) | OK
- Built a macro named `mcrHollandGroup.PrintSelectedRecord`
- Added **command button** to the form



- set the button's **OnClick** property to submacro's name
- set the button's **Picture** property to display printer icon
- set the button's **Caption** property to Print selected record

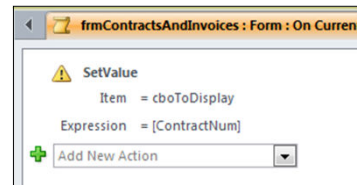
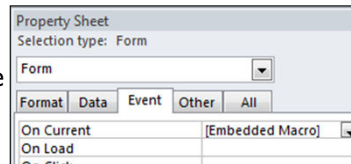
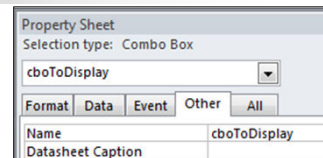
Practice Time: Using a Macro to Synchronize a Combo Box... The Problem

- Open `frmContractsAndInvoices`, use the combo box to navigate to two other contracts
 - which **event** was used to provide this functionality?
 - which **macro** was called?
 - note: it is an **embedded** macro (is not contained in a macro sheet)
 - which macro **action(s)** were executed to provide this functionality?
- Use Navigation Buttons  to view other contracts and watch the combo box as you do so. It doesn't change!
- When a **user navigates to another record on the form**, we need to change the combo box's value to match the contract being displayed
 - create an embedded **macro** to handle the Form's **Current** event
 - the macro will set the combo box control's value equal to the value of the `ContractNum` displayed on the main form

11

Practice Time: Using a Macro to Synchronize a Combo Box... The Solution

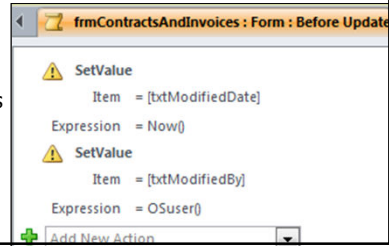
- Name the combo box control so we can refer to it in a macro
- Create an embedded **macro** to handle the form's **Current** event
 - the macro will use the **SetValue** action to set the combo box control's value to the value of the `ContractNum` displayed on the main form
 - note: **SetValue** is not available unless you select **Show All Actions**
- Test that the combo box remains synchronized when users switch to other records
- Save `frmContractsAndInvoices` when working correctly



Practice Time: Using a Macro to Track Who Changes a Record

- Enhancement: When a record is saved, store the date/time and name of the user who created/changed the record
- Modify `tblContract` to include two new fields:
 - ModifiedDate Date/Time Format as GeneralDate
 - ModifiedBy Text(40)
- Modify `frmContractsAndInvoices`
 - add both of these new fields to Detail section
 - set properties for both text boxes
 - Enabled=No, Locked=Yes
 - BorderStyle=Transparent, BackStyle=Transparent
 - Name (txtModifiedDate, txtModifiedBy)
 - Create this embedded macro to handle the form's `BeforeUpdate` event
 - Modify a few Contract records to test that the macro works correctly
 - Save `frmContractsAndInvoices` when working correctly

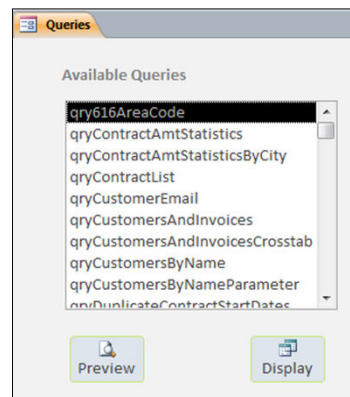
ModifiedDate	ModifiedDate
ModifiedBy	ModifiedBy



Session 2

Designing a User Interface

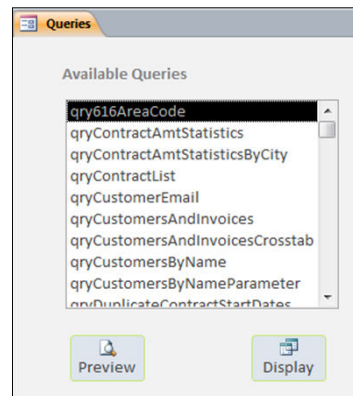
- User Interface
 - how users issue commands to and receive feedback from your application
- UI elements
 - Ribbon
 - Shortcut Menu
 - Data Entry Forms
 - bound forms to view/enter/edit/delete records
 - Dialog Box Form
 - an unbound form used to collect user input, or to present messages to a user
 - Navigation Form
 - an unbound form that provides convenient access to specified database objects



Creating the frmQueries Form

- This dialog box form had
 - a label
 - a list box
 - pair of command buttons

- Form **properties** determine its appearance & behavior
 - Caption (Queries)
 - Record Selectors (No)
 - Navigation Buttons (No)
 - Close button (No)
 - Shortcut Menu (No)
 - RecordSource (null)



15

SQL

- Structured Query Language - IBM - 1970's
- An industry-standard language
- Access uses SQL extensively
 - Tutorial 5: LookupWizard set the **RowSource** property for a foreign key field's combo box
 - Tutorial 10: **RowSource** property for list box that showed query names
 - Query SQL View
 - Access translates your Query Design into the corresponding SQL statement
 - when you change Query Design, Access translates to equivalent SQL statement
 - when you run the query, Access sends the SQL statement to the database engine for execution
 - when you save a query object, Access stores the **text of the SQL statement**

Contract Num	Customer ID
3011	Fernandez, Sabrina
3012	Fernandez, Sabrina 11001
3015	Finn's on the Waterfront 11030
3017	Fox and Hound Grille 11055

General	Lookup
Display Control	Combo Box
Row Source Type	Table/Query
Row Source	SELECT [qryCustomersByName].[Customer], [qryCus
Bound Column	2

Field:	ContractNum	ContractAmt	StartDate	ContractType
Table:	tbiContract	tbiContract	tbiContract	tbiContract
Sort:			Ascending	
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:		> 40000	> = #1/1/2014#	

```

qryLargeContractsAfter2014
SELECT ContractNum, ContractAmt, StartDate, ContractType
FROM tbiContract
WHERE ContractAmt > 40000 AND StartDate > = #1/1/2014#
ORDER BY StartDate;
  
```



SQL: SELECT Queries

- Select queries - select fields from one or more tables

```
SELECT fields
FROM tables
WHERE condition(s)
ORDER BY sortfield(s)
```


- Practice Time

- Create a new query in Design view but close the Show Table dialog without selecting any tables. Switch to SQL view and enter the following SQL statement:

```
SELECT *
FROM tblContract;
```

- Run the query... it shows all 8 fields and all 66 rows from the table
- Save the query as `qrySimpleSelect`

17



SQL: SELECT Queries

```
SELECT fields
FROM tables
WHERE condition(s)
ORDER BY sortfield(s)
```

- Practice Time
 - Switch to SQL view and modify the SQL statement to:

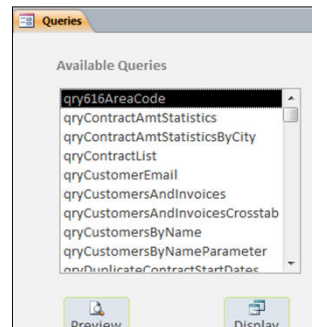
```
SELECT ContractNum, ContractAmt, StartDate, ContractType
FROM tblContract
WHERE ContractAmt > 40000 AND StartDate >= #1/1/2014#
ORDER BY StartDate;
```

- Save the query as `qryLargeContractsAfter2014`
- Run the query, it shows 4 fields and 6 rows, sorted by StartDate
- Switch to Design View, add `tblCustomer` to the query, then add LastName and Phone as new columns to query
- Run the query... it now has customer info
- Switch to SQL View and notice the `INNER JOIN` clause
- Save `qryLargeContractsAfter2014`

18

Adding a List Box to a Form Using an SQL Statement for a List Box

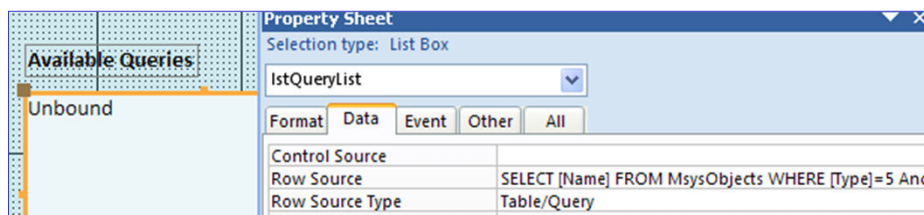
- Wanted list box to display the names of the query objects
 - where can we obtain the name of each query?
- MSysObjects
 - a **hidden system table** contains information about objects in the DB
 - right-click Navigation Pane | Navigation Options... | Show Hidden Objects
 - **Demo:** inspect MSysObjects table
 - **Type** field values indicate each object's type
 - Table = 1
 - Query = **5**
 - Form = -32768
 - Report = -32764
 - Macro = -32766
 - Module = -32761



Adding a List Box to a Form Using an SQL Statement for a List Box

- Set the list box's **Name** property to `IstQueryList`
- Set the list box's **RowSource** property to

```
SELECT [Name]
FROM MSysObjects
WHERE [Type] = 5 AND Left([Name],1) <> "~"
ORDER BY [Name];
```

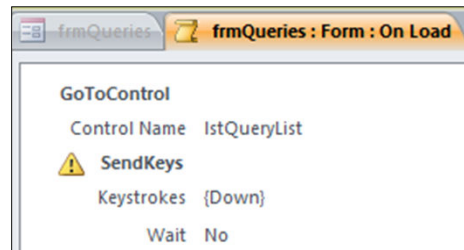
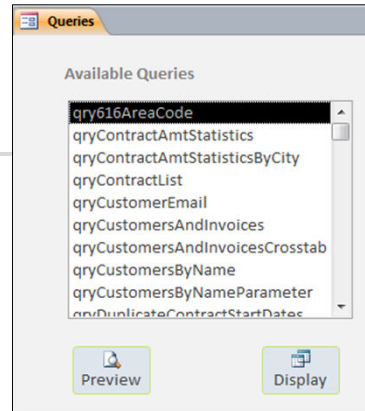


20

Creating Macros for frmQueries

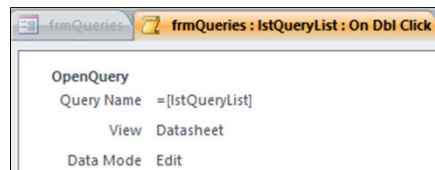
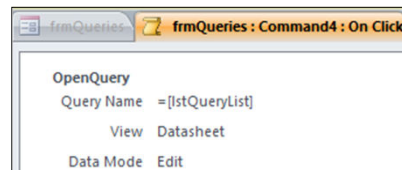
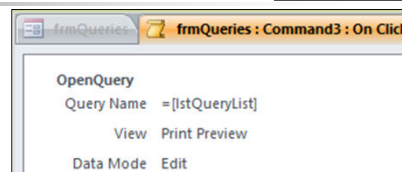
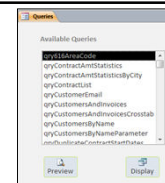
- Wanted the listbox's first query to be selected when form loads
 - why?
 - which event was used?
 - which action(s) did the macro execute?

- Created an embedded macro to handle the Form's **Load event**



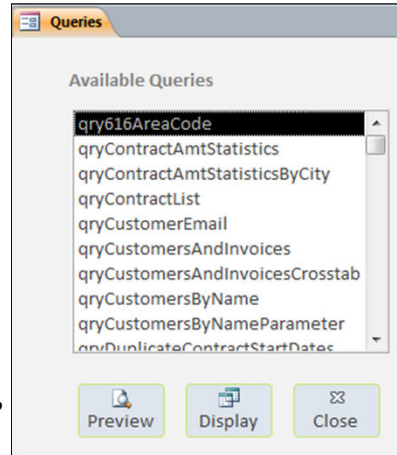
Creating Macros for frmQueries

- Built macros execute **OpenQuery** action in response to
 - Preview button's Click event
 - Display button's Click event
 - List Box's Double-Click event



Practice Time

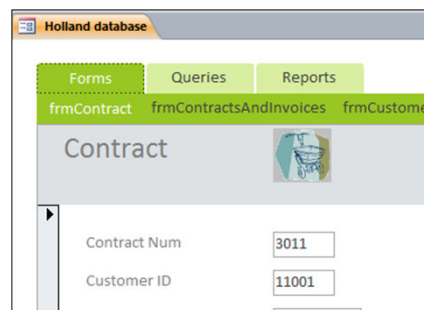
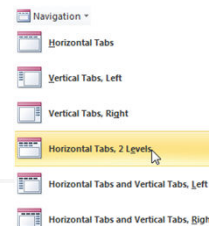
- Add a Close button to frmQueries
 - activate Control Wizards
 - add a command button to form
 - Form Operations, Close Form
 - set Caption and Picture properties
 - set Back Color to match other buttons
 - size, place, align as illustrated
- What did the Control Wizard do?
 - what Name **property** did it set?
 - what **embedded macro** did it create?
 - what **action(s)** did the macro execute?



23

Creating a Navigation Form

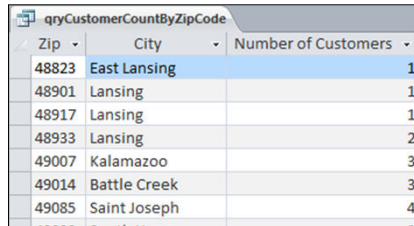
- What type of **object** is it?
- What is its purpose?
- How do **users** benefit?
- How do **developers** benefit?
- Navigation control



24

Practice Time

- Create a new query `qryCustomerCountByZipCode` that displays each zip code and city and the number of customers living in the zip code.



Zip	City	Number of Customers
48823	East Lansing	1
48901	Lansing	1
48917	Lansing	1
48933	Lansing	2
49007	Kalamazoo	3
49014	Battle Creek	3
49085	Saint Joseph	4
49088	South Haven	2

- Create a new report `rptCustomerCountByZipCode` that displays the results of `qryCustomerCountByZipCode`, sorted in descending order of number of customers.
- Modify `frmNavigation`'s Reports page to expose `rptCustomerCountByZipCode`. Test that it works.
- Layout View

25