



## Week 8:

- File Selector
- String Manipulation
- Dictionary Object of Scripting Env
- Data Bases
- Using Access from vbs

---



---



---



---



---



---



---



## FileSelector.vbs

- Look at the example from Week 8
- Common File Dialog Box only in XP
- Using File Dialog from Excel or Word works when otherwise unavailable.
- How will the calling routine know if no file has been selected?
- Be sure to close excel (or word) right away:

`XlAppl.Application.Quit`

---



---



---



---



---



---



---



## ExampleWithStrings.vbs

- *Left, Right, Mid, and Replace*: seen before.
- *Split* breaks string into array of segments:  
`Dim ArrayFromSplitBySpace`  
`ArrayFromSplitBySpace = Split(SourceString, " ", -1, vbTextCompare)`

---



---



---



---



---



---



---



## ExampleWithStrings.vbs

- *Filter* selects from array strings those elements containing (or not) a specified string:  
`Dim ArrayFromFilterIng`
- `ArrayFromFilterIng = Filter(ArrayFromSplitBySpace, "ing", True, vbTextCompare)`
- *FileNameAndExtensionFromFullPath.vbs* uses *split* to break up path into array of the nested directories.

---

---

---

---

---

---

---

---



## Dictionary.vbs

```
Dim myDictionary
Set myDictionary =
CreateObject("Scripting.Dictionary")
```

•See Page 234-43 of Lomax  
 •MSDN Help Pages

---

---

---

---

---

---

---

---



## Dictionary.vbs

```
myDictionary.Add "b", "Banana" ' b is the key, and Banana is item
myDictionary.Add "f", "FruitCake"
myDictionary.Add "g", "Grape Fruit"
myDictionary.Add "e", "Evil"
myDictionary.Add "a", "Apple" ' "a" is the key, and "Apple" is item
myDictionary.Add "d", "Donut"
```

```
allKeys = dic.Keys
' allKeys is an array to all the keys
```

```
allItems = dic.Items
' allItems is an array to all the items
```

Can use For Each loops or For loops to step through them

---

---

---

---

---

---

---

---



## Dictionary.vbs

```
allKeys = dic.Keys
' allKeys is an array to all the keys

allItems = dic.Items
' allItems is an array to all the items

For Each item in dic.Items
    msg = msg & "Item=" & item & vbCrLf
Next
```

Can use For Each loops or For loops to step through them

---

---

---

---

---

---

---

---



## UseDictionary.vbs

```
Function ReadInputStreamToDictionary(ByRef inputStream, ByRef dic)
    Dim counter, idData, otherData
    counter = 0
    Do While inputStream.AtEndOfStream <> TRUE
        idData = inputStream.Read(kNumDigitInId)
        otherData = inputStream.ReadLine
        ' now insert into the dictionary
        dic.Add idData, otherData
        counter = counter + 1
    Loop
    ReadInputStreamToDictionary = counter
End Function
```

---

---

---

---

---

---

---

---



## UseDictionary.vbs

```
Sub PrintDictionary(ByRef dic)
    allKeys = dic.Keys
    allItems = dic.Items

    msg = "There are " & dic.Count & " number of entries in the dictionary"
    For index = 0 To dic.Count-1
        ' Notice, dictionary range goes from 0 to count-1
        msg = msg & "Index=" & index & " Key=" & allKeys(index) & " Item=" &
allItems(index) & vbCrLf
    Next
    MsgBox msg
End Sub
```

---

---

---

---

---

---

---

---



## UseDictionary.vbs

```

Sub DeleteUnwantedEntries (ByRef dic)
  userChoice = "a"
  Do While userChoice <> "-1" And userChoice <> ""
    userChoice = InputBox("What to remove (-1 to stop deleting)? ")
    If userChoice <> "-1" And userChoice <> "" Then
      If dic.Exists(userChoice) Then
        dic.Remove(userChoice)
      Else
        MsgBox userChoice & " is not a defined key! Try again!"
      End If
    End If
  Loop
End Sub

```

---

---

---

---

---

---

---

---



## To write to word file

```

Dim sel
Set sel = WdApp.Application.Selection
' This gets back reference for editing

With sel
  .Font.Bold = wdToggle ' toggle bold on
  .TypeText "User Data Information: "
  .TypeParagraph
  .TypeParagraph
  .Font.Bold = wdToggle
End WithEnd Sub

```

---

---

---

---

---

---

---

---



## To write to word file

```

allKeys = dic.Keys
allItems = dic.Items
Const wdToggle = 9999998
For index = 0 To dic.Count-1
  With sel
    .Font.Bold = wdToggle (also true or false)
    .TypeText index+1 & ". ID=" ' in bold
    .Font.Bold = wdToggle
    .TypeText allKeys(index) & " -- " ' not bold
    .Font.Bold = wdToggle
    .TypeText "Item=" ' in bold
    .Font.Bold = wdToggle
    .TypeText allItems(index) ' not bold
    .TypeParagraph ' new paragraph
  End With
Next

```

---

---

---

---

---

---

---

---



## Relational Data Bases (e.g. Access)

Simple Access Data Base consists of:

- **Tables** containing the data
- **Queries** linking the tables to each other by relationships and extracting particular information from them.
- **Reports**: query results in pleasing form
- **Forms**: user-friendly way to enter data into tables.

---

---

---

---

---

---

---

---



## Relational Data Bases (e.g. Access)

Each **Table** contains:

- **Fields (columns)**, each of which contains a particular type of data (*e.g. FirstName, LastName, ID, etc*)
- **Records (rows)**, each of which contains a related set of data entries in each field. (*e.g. James Smith on one record, Sally Smith on another*)

---

---

---

---

---

---

---

---



## Relational Data Bases (e.g. Access)

Each **Query** :

- (Optionally) joins data tables to each other according to particular entries in them.
- Selects from a table or joined tables particular records (rows) containing desired entries (*e.g. all students from CA*)
- For the selected rows, displays desired fields. (*e.g. last names only*)
- Can be expressed graphically (with Wizard) or by SQL statements.

---

---

---

---

---

---

---

---



## Relational Data Bases (e.g. Access)

**Each Form:** allows error-free, convenient entry of data into data tables.

**Each Report:** provides information from a table or query in a pleasing well-formatted and labelled manner.

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Three tables in demo database:
  - StudentInfo (1 record/student)
  - MajorName (data pertinent to each major)
  - StateNames (data pertinent to each state)
- At least one unique (key) field in each, new row to be added at bottom.
- Design Views: description of each field, key field
- Some fields could provide linkages between tables.

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Query AllCssStudentsbyCities:
  - Selects from table StudentInfo all students with major CSS
  - Reports last name, first name, city, and major for each one.
  - Orders the entries alphabetically by city.

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Query AllCssStudentsbyCities in SQL version:

```
SELECT StudentInfo.LastName, StudentInfo.FirstName,
       StudentInfo.City, StudentInfo.Major
FROM StudentInfo
WHERE (((StudentInfo.Major)="CSS"))
ORDER BY StudentInfo.City;
```

*Could be used to carry out this query in a text-based interface.*

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Query AllBusStudents :
  - Requires three tables
  - Query joins them on State and major symbols
  - Displays results from the joined table.
- Consider design view

*Could be used to carry out this query in a text-based interface.*

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Query AllBusStudents in SQL view :
 

```
SELECT StudentInfo.LastName, StudentInfo.City,
       StateNames.[State Name], MajorName.[Major Name]
FROM (StudentInfo INNER JOIN StateNames ON
      StudentInfo.StateOrProvince =
      StateNames.Abbreviation) INNER JOIN MajorName ON
      StudentInfo.Major = MajorName.Abbreviation
WHERE (((StudentInfo.Major)="BUS"));
```

*Could be used to carry out this query in a text-based interface. Excuse the spelling of abbreviation!*

---

---

---

---

---

---

---

---



## StudentInfo.mdb

- Look at the other queries in this database.
  - AllStudentsByID: sorted on a field that is not printed.

*Could be used to carry out this query in a text-based interface. Excuse the spelling of abbreviation!*

---

---

---

---

---

---

---

---



## AccessCom.vbs

- Familiar: tool we have used with Excel, Word, and PowerPoint.
- Notice that it finds more tables than the three we are familiar with.
- Look at the function *Open\_OFFICE\_Document*. *It is general purpose for all of the office apps that use the file dialog box to open files.*
- *Seems to lack many of the features we need to manipulate the Access database. Actually, the entire object model is there – but may not be easily used.*

---

---

---

---

---

---

---

---



## ADO (ActiveXObject) ;AdoAccess.vbs

- Establish *connection* to the database:  

```
Set conn = CreateObject("ADODB.Connection")
conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=StudentInfo.mdb"
```
- Create a *command object* and set its active connection.  

```
Set cmd = CreateObject("ADODB.Command")
set cmd.ActiveConnection = conn
```

*We have now linked the command object to the connection object (which is the database file)*

---

---

---

---

---

---

---

---



*ADO (ActiveXObject) ;AdoAccess.vbs*

- Select one of the data tables in the database for *cmd*:  
`cmd.CommandText = "StudentInfo"`
- Open the recordset, the entire table.
  - Set `rs = CreateObject("ADODB.RecordSet")`
  - `rs.CursorLocation = adUseClient`
  - `rs.Open cmd, ,adOpenStatic, adLockBatchOptimistic`

This creates a recordset object *rs* and links it to the data source referenced by the command object *cmd*.

---

---

---

---

---

---

---

---



*ADO (ActiveXObject) ;AdoAccess.vbs*

- The program prints the table,

```
Set myFields = rs.Fields
' this gets the fields of the "current record"

For Each fieldItem in myFields
  Msg = Msg & fieldItem.Name & ": [" &
    fieldItem.Value & "]"
Next

rs.MoveNext      (now loops back )
```

---

---

---

---

---

---

---

---



*ADO (ActiveXObject) ;AdoAccess.vbs*

- The program adds a record to it, by polling the user field by field

```
rs.AddNew      current record is newly created one
Set myFields = rs.Fields      fields of newly created record
```

```
For Each fieldItem in myFields
  If CBool(fieldItem.Attributes And adFldsNullable) Then

    fieldItem.Value = InputBox("Value for" & fieldItem.Name)
  End If
Next
```

---

---

---

---

---

---

---

---



### ADO (ActiveXObject) ;AdoAccess.vbs

- The program adds a second record to it, by filling array and then adding it as a record

```
StudentInfoFieldNames = Array("First Name", "Last Name", "City",
                              "State/Province", "Major")
StudentInfoField = Array(1, 2, 3, 4, 5)
fieldSize = 4
```

```
For index = 0 To fieldSize
  data(index) = InputBox("Please enter the value for:" &
                        StudentInfoFieldNames(index))
Next
rs.AddNew StudentInfoField, data
```

---

---

---

---

---

---

---

---

---

---



### ADO (ActiveXObject) ;AdoAccess.vbs

- The program sorts the record set

```
rs.Sort = "StateOrProvince DESC, StudentID ASC, Major DESC"
```

- The program filters the record set

```
stateStr = InputBox("We can limit the printing of the database
... Enter a state to filter on:")
rs.Filter = "StateOrProvince" & "=" & stateStr & "'"
```

Notice the name of the variable (*stateStr*) is in single quotes.  
Does the string "stateStr" or its value appear here?  
Hides, but does not change the record set.

---

---

---

---

---

---

---

---

---

---



### ADO (ActiveXObject) ;AdoAccess.vbs

- The program saves the record set

```
conn.BeginTrans
rs.UpdateBatch
conn.CommitTrans
' Performing the update
```

```
rs.Close
```

```
conn.Close
```

---

---

---

---

---

---

---

---

---

---



### ADO (ActiveXObject) ;AdoAccess.vbs

- *rs* methods such as *fields*, *addnew*, *EOF*, *MoveFirst*, etc can be used to add records, print them, manipulate individual fields, etc.
- Finally, the data is saved with the *BeginTrans*, *UpdateBatch*, and *CommitTrans* methods.
- Create a separate *recordset* for each table we wished to manipulate.
- Have not yet included queries....

---

---

---

---

---

---

---

---



### ADOQueryAccess.vbs

- Can access and print either data tables or queries.
- Create a connection to the database file, as before:

```
CreateConnection conn, "StudentInfo.mdb"
' Establish Connection to "StudentInfo.mdb"
```

---

---

---

---

---

---

---

---



### ADOQueryAccess.vbs

- Create command *cmd* and record set *rs* to each table or query to be used

```
CreateCommand cmd, conn, "StudentInfo"
' The all the records from the specified table (StudentInfo)
CreateRecordSet rs, cmd ' Create Record Set
```

```
' We can have more than one record set (now a query)...
CreateCommand qcmd, conn, "AllWASStudentsByID"
CreateRecordSet qrs, qcmd
```

```
CreateCommand qcacmd, conn, "StudentLastNameInCA"
CreateRecordSet qcars, qcacmd
```

---

---

---

---

---

---

---

---



## ADOQueryAccess.vbs

- In this example, we just print the three records sets.
- A query is run when called, so this represents a printout of the current queried data.

```
' Manipulate the data
MsgBox "All Students info details for students from WA;"
PrintRecordSet qrs

MsgBox "All students Lastname only from CA"
PrintRecordSet qcars

PrintRecordSet rs
```

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Activity 14 with partner

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- In this example, we work through the database modifications required in Activity 13
- First we add data to all three tables.
- Then, we run new queries defined within the vbs code.

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- After connecting to the data base set up commands and record sets for the three data tables:

```
CreateCommand cmdStdInf, conn, "StudentInfo"
CreateRecordSet rsStdInf, cmdStdInf
```

```
CreateCommand cmdMajNam, conn, "MajorName"
CreateRecordSet rsMajNam, cmdMajNam
```

```
CreateCommand cmdStNam, conn, "StateNames"
CreateRecordSet rsStNam, cmdStNam
```

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Add new records to each data table:

```
recordData = Array("Minnesota", "MN")
AddNewDataToRecordSet rsStNam, recordData
```

```
'Add new major to table of majors
recordData = Array("Applied Computing", "AC")
AddNewDataToRecordSet rsMajNam, recordData
```

```
'Add two new students to the data table
recordData = Array("Carol", "Johnson", "St. Paul", "MN", "AC")
AddNewDataToRecordSet rsStdInf, recordData
```

```
recordData = Array("Frank", "Stevens", "Minneapolis", "MN", "CSS")
AddNewDataToRecordSet rsStdInf, recordData
```

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Update and close the three tables in the original file:

```
UpdateData conn, rsStNam
UpdateData conn, rsMajNam
UpdateData conn, rsStdInf
```

```
Sub UpdateData(ByRef conn, ByRef rs)
    conn.BeginTrans
    rs.UpdateBatch
    conn.CommitTrans
    rs.Close
End Sub
```

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Run the **four queries** defined within the database file, creating record sets, :

```
CreateCommand qcmdWASstds, conn, "AllWASStudentsByID"
CreateRecordSet qrsWASstds, qcmdWASstds
```

```
CreateCommand qcmdStdLastNamCA, conn, "StudentLastNameInCA"
CreateRecordSet qrsStdLastNamCA, qcmdStdLastNamCA
```

```
CreateCommand qcmdAllBusStudents, conn, "AllBusStudents"
CreateRecordSet qrsAllBusStudents, qcmdAllBusStudents
```

```
CreateCommand qcmdAllCSsStudentsbyCities, conn, "AllCSsStudentsbyCities"
CreateRecordSet qrsAllCSsStudentsbyCities, qcmdAllCSsStudentsbyCities
```

---

---

---

---

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Create record set for new query on AC major. The SQL command takes place of query *name* in previous examples :

```
queryString = "SELECT StudentInfo.LastName, StudentInfo.FirstName,
StudentInfo.City, StudentInfo.Major " & _
"FROM StudentInfo " & _
"WHERE (((StudentInfo.Major)= 'AC')) " & _
"ORDER BY StudentInfo.City;"
```

```
CreateCommand qcmdAllACstds, conn, queryString
CreateRecordSet qrsAllACstds, qcmdAllACstds
```

This comes nearly from the SQL view of the Access query. Minor issues only.

---

---

---

---

---

---

---

---

---

---

---

---



## ADOActivity13Access.vbs

- Create record set for new query on MN state. The SQL command takes place of query *name* in previous examples :

```
Dim qcmdAllMNstds, qrsAllMNstds
```

```
queryString = "SELECT StudentInfo.LastName, StudentInfo.City,
StudentInfo.Major, StateNames.[State Name] " & _
"FROM StudentInfo INNER JOIN StateNames ON StudentInfo.StateOrProvince =
StateNames.Abbreviation " & _
"WHERE (((StudentInfo.StateOrProvince)= 'MN')) " & _
"ORDER BY StudentInfo.StudentID;"
```

```
CreateCommand qcmdAllMNstds, conn, queryString
CreateRecordSet qrsAllMNstds, qcmdAllMNstds
```

---

---

---

---

---

---

---

---

---

---

---

---

## ADOActivity13Access.vbs

- AddNewDataToRecord() :

```
Sub AddNewDataToRecordSet(ByRef rs, ByRef recordData)
    Dim myFields, fieldItem
    Dim numDataFields, indexRecordData
    numDataFields = UBound(recordData) (input new data)

    ' now let's add a new row into the table ...
    rs.AddNew
    ' now, the "current record" is the newly created one, we get the data
    ' from the calling parameter array and add them in

    Set myFields = rs.Fields (collection object of all fields)
```

---

---

---

---

---

---

---

---

## ADOActivity13Access.vbs

- AddNewDataToRecord() :

```
' This will add a record row
indexRecordData = 0
For Each fieldItem in myFields
    ' now add in the value

    fieldItem.Value = recordData(indexRecordData)
    indexRecordData = indexRecordData + 1
Next

Requires the data array provided at the call to have elements
that match up properly with the fields in the record set.
```

---

---

---

---

---

---

---

---

## ADOQuerywithPARG.vbs

- Will run a query with input parameter required
- Create a parameter object :

```
Dim state
state = InputBox("Enter the desired state abbreviation: ")

CreateCommand majorStateCmd, conn, "MajorsfromEnteredState"

' we create a "Parameter" :-)

Set majorStateParm = CreateObject("ADODB.Parameter")
majorStateParm.Name = "StateOrProvince"
majorStateParm.Type = adVarChar
majorStateParm.Direction = adParamInput
majorStateParm.Size = Len(state)
majorStateParm.Value = state
```

---

---

---

---

---

---

---

---



## ADOQuerywithPARM.vbs

- Append the parameter to the query command
- The command object has a parameter collection.
- You append this parameter object to that collection
- and create the record set :

```
majorStateCmd.Parameters.Append majorStateParm  
' now append qparm as a parameter to the command
```

```
CreateRecordSet majorStateRS, majorStateCmd
```

---

---

---

---

---

---

---

---