Date and Time Functions

Ferddie Quiroz Canlas

Foreword

- Date and Time Functions are essentials to a programmer specifically for processes like:
  - Payroll
  - Loan
  - Attendance
  - Reservations
  - Etc

Recall

- Recall the VB keywords that reference the current date and/or time:
  - **Now** Returns the current date and time together
  - **Date** Returns the current date
  - **Time** Returns the current time
  - Date and Time Constants are in this format #date/time #

The Date and Time Functions

DateValue()

- Returns the date portion of a Date/Time value, with the time portion "zeroed out".
- Note: When the time portion of a date/time variable is "zeroed out", the time would be interpreted as 12:00 AM.

  Example:

```
Dim dtmTest As Date
dtmTest = DateValue(Now)
```

TimeValue()

- Returns the time portion of a Date/Time value, with the date portion "zeroed out".
- Note: When a date/time variable is "zeroed out", the date will actually be interpreted as December 30, 1899.

Example

```
Dim dtmTest As Date
dtmTest = TimeValue(Now
```

Weekday()

- Returns a number from 1 to 7 indicating the day of the week for a given date, where 1 is Sunday and 7 is Saturday.

Example

IntDOW = Weekday(Now)

Weekday Constants

- Constant Value
- vbSunday 1
- vbMonday 2
- vbTuesday 3
- vbWednesday 4
- vbThursday 5
- vbFriday 6
- vbSaturday 7

WeekdayName()

- Returns a string containing the weekday name ("Sunday" thru "Saturday"), given a numeric argument with the value 1 through 7.

  Example:

  strDOW = WeekdayName(6)

WeekdayName()

- The WeekdayName function takes an optional, second argument (Boolean) indicating whether or not to abbreviate the weekday name. By default, the second argument is False, meaning do not abbreviate and return the full name. If True, the first three letters of the weekday name will be returned:

  Example:

  ```
  strDOW = WeekdayName(6, True)
  ```

WeekdayName()

- You can nest the Weekday function within the WeekdayName function to get the weekday name for a given date:

  Example:

  strDOW = WeekdayName(Weekday(Now))

Month()

- Returns a number from 1 to 12 indicating the month portion of a given date.

Example:

```
intMonth = Month(Now)
```

MonthName()

- Returns a string containing the month name ("January" thru "December"), given a numeric argument with the value 1 through 12.

Example:

strMoName = MonthName(8)

MonthName()

- The MonthName function takes an optional, second argument (Boolean) indicating whether or not to abbreviate the month name. By default, the second argument is False, meaning do not abbreviate and return the full name. If True, the first three letters of the month name will be returned:

Example:

strMoName = MonthName(8, True)

MonthName()

- You can nest the Month function within the MonthName function to get the month name for a given date:

Example:

strMoName = MonthName(Month(Now))

Day()

- Returns a number from 1 to 31 indicating the day portion of a given date.

Example:

intDay = Day(Now)

Year()

- Returns a number from 100 to 9999 indicating the year portion of a given date.

Example:

```
intYear = Year(Now) ' intYear = 2001
```

Hour()

- Returns an integer specifying a whole number between 0 and 23 representing the hour of the day.

Example:

intHour = Hour(Now)

Minute()

- Returns an integer specifying a whole number between 0 and 59 representing the minute of the hour.

Example:

intMinute = Minute(Now)

Second()

- Returns an integer specifying a whole number between 0 and 59 representing the second of the minute.

Example:

```
intSecond = Second(Now) ' intSecond = 20
```

Date and Time Arithmetic

The DatePart Function

- The generic **DatePart** function returns an Integer containing the specified part of a given date/time value.
- It incorporates the functionality of the Weekday, Month, Day, Year, Hour, Minute, and Second functions.
- It can be used to get the quarter of a given date (1 through 4) , the "Julian" date (the day of the year from 1 to 366), and the week number (1 through 53)
- Syntax:

  DatePart(*interval, date[,firstdayofweek[, firstweekofyear]]*)

The DatePart Function

Part	Description
interval	Required. String expression that is the interval of time you want to return. The string expression can be any of the following: Expression / Description / Possible Range of Values "yyyy" / Year / 100 to 9999 "q" / Quarter / 1 to 4 "m" / Month / 1 to 12 "y" / Day of year / 1 to 366 (a "Julian" date) "d" / Day / 1 to 31 "w" / Weekday / 1 to 7 "ww" / Week / 1 to 53 "h" / Hour / 0 to 23 "n" / Minute / 0 to 59 "s" / Second / 0 to 59
date	Required. Date value that you want to evaluate.
firstdayofweek	Optional. A constant that specifies the first day of the week. If not specified, Sunday is assumed.
firstweekofyear	Optional. A constant that specifies the first week of the year. If not specified, the first week is assumed to be the week in which January 1 occurs.

Piecing Separate Numbers Together to Form a Date or Time Value

- If you have the separate parts of a date/time value in different variables and want to piece them together to formulate a date or time, there are two functions you can use to do this: **DateSerial** and **TimeSerial**.
- The **DateSerial** takes three numeric arguments: year, month, and day respectively. It returns a date based on those values.
- The **TimeSerial** takes three numeric arguments: hour, minute, and second respectively. It returns a time based on those values.

DateAdd()

- NewDate = DateAdd(interval, number, date)
- *interval* is a string that indicates a date or time unit (see Slide 22)
- *number* is the number of units you are adding
- *date* is the starting date.

DateDiff()

- When you have two dates and you want to evaluate the difference between them—that is, the time elapsed between one date and the next.
- Syntax:

Result = DateDiff(interval, startdate, enddate [, FirstDayOfWeek[, FirstWeekOfYear]])

DateDiff()

- *interval* has the meaning shown in Slide 22
- *FirstDayOfWeek* is an optional argument that specifies which weekday should be considered as the first day of the week
- *FirstWeekOfYear* is another optional argument that lets you specify which week should be considered as the first week of the year.(As shown on the next slide)

DateDiff()

Constant	Value	Description
vbUseSystem	0	Use the NLS API setting.
vbFirstJan1	1	The first week is the one that includes January 1. (This is the default value for this setting.)
vbFirstFourDays	2	The first week is the first one that has at least four days in the new year.
vbFirstFullWeek	3	This first week is the first one that's completely contained in the new year.

Date and Time Formatting

Format()

- This function gives you seven different, named formats for date and time:
  - *General Date* (date and time in general format; only the date if the fractional part is 0; only the time if the integer part is 0)
  - *Long Date* (for example, *Friday, August 14, 1998,* but results vary depending on your locale)
  - *Medium Date* (for example, *14-Aug-98)*
  - *Short Date* (for example, *8/14/98)*
  - *Long Time* (for example, *8:35:48)*
  - *Medium Time* (for example, *8:35 A.M.)*
  - *Short Time* (for example, *8:35* in a 24 hour format)

FormatDateTime()

- Permits only a subset of the *Format* function's named formats

  Syntax: *result = FormatDateTime(Expression, [NamedFormat]) .*

- *NamedFormat* can be one of the following intrinsic constants:
  - 0-vbGeneralDate (the default)
  - 1-vbLongDate
  - 2-vbShortDate
  - 3-vbLongTime
  - 4-vbShortTime.

Thank you very much and
God Bless!