Tematický celek 05

5.1 Práce ze soubory

- slouží pro ukládání záznamů,
- každý ze záznamů obsahu jedno nebo více polí.

Definice typu záznamu

- musíme definovat datový typ, který bude sloužit k uložení záznamu:

```
Type Zaznam
     Jmeno As String * 15
     Prijmeni As String * 15
     Datum As Date
End Type
```

- všechny záznamy musí mít stejnou délku, proto musíme u stringů definovat jejich délku.

Otevření souboru

```
Open cesta [For Random] As cislo Len=reclength
```

- *For Random* – je nepovinné, protože je to výchozí nastavení,
- *Len* – určuje délku souboru v bajtech.

Přečtení záznamu do proměnných

```
Get cislo,pozice, promenna
```

- *pozice* – číslo záznamu, který se má kopírovat
- *promenna* – kam se má záznam uložit.

Uložení záznamu

```
Put cislo,pozice, promenna
```

- *pozice* – kam se má záznam uložit
- *promenna* – zdroj záznamut.

Test konce souboru

```
EOF (cislo)
```

- vrátí True, jestliže jsme na konci souboru *cislo*.

Mazání záznamů

- VB bohužel neumožňuje přímo smazat záznam,
- musíme zkopírovat všechny platné záznamy do nového souboru,
- uzavřít původní soubor a smazat ho příkazem **Kill**,
- přejmenovat nový soubor na původní jméno pomocí příkazu **Name**.

5.2 Objektový model FSO

- umožňuje pro práci se složkami a soubory používat objektové nástroje,
- je obsažen v knihovně Scripting (Scrrun.dll),
- zatím umožňuje pouze práci s textovými soubory.
- obsahuje tyto objekty:

DriveListBox - umožňuje získat informace o jednotkách v systému.

DirListBox - práce se složkami.
FileListBox - práce se soubory.
FileSystemObject – hlavní objekt této skupiny obsahující množství metod pro práci s těmito objekty.
TextStream - pomocí tohoto objektu můžeme zapisovat a číst textové soubory.

1. Vytvoření objektu FileSystemObject

- můžeme k tomu použít jednu ze dvou metod:
  - vytvoření objektové proměnné:

```
Dim fso As New FileSystemObject
```

  - vytvoření objektu pomocí metody CreateObject:

```
Set fso = CreateObject(„Scripting.FileSystemObject“)
```

2. Použití vhodné metody

- vytvoření dalších objektů:
  - metoda **CreateFolder:**

```
Dim f As Folder
Set f = fso.CreateFolder(cesta)
```

  - metoda **CreateTextFile:**

```
Dim tf As TextStream
Set tf = fso.CreateTextFile(cesta+jmeno, True)
```

- přístup k existujícím jednotkám, souborům a složkám:
  - GetDrive,
  - GetFolder,
  - GetFile.
- kopírování (CopyFile, CopyFolder),
- přesouvání (MoveFile, MoveFolder),
- mazání (DeleteFile, DeleteFolder),
- ověření existence (DriveExists, FolderExists, FileExists),
- otevření textového souboru (OpenTextFile).

3. Přístup k vlastnostem objektu

jednotky* (objekt Drive)*:

- velikost jednotky v bajtech (TotalSize), volný prostor (AvailableSpace, FreeSpace),
- písmeno jednotky (DriveLetter),
- typ jednotky (DriveType),
- sériové číslo jednotky (SerialNumber),
- funkce: CurDir (aktuální adresář), ChDrive, Chdir (změna jednotky, adresáře), App.Path (cesta k adresáři k aplikaci).

***adresáře* (objekt Dir):**

- zrušení složky (Delete nebo FileSystemObject.DeleteFolder),
- přemístění složky (Move nebo FileSystemObject.MoveFolder),
- zkopírování složky (Copy nebo FileSystemObject.CopyFolder),
- získání jména složky (Name).

5.3 Ošetření chyb

- chyby vznikají problémy s hardwarem nebo nepředpokládanými akcemi uživatele,
- pro ošetření chyby slouží **chybové rutiny**.

5.3.1 Návrh chybové rutiny

- chybová rutina je číst procedury určená pro zachytávání chyb a reakcí na ně,
- chybové rutiny bychom měli přidávat do každé procedury, ve které se dá očekávat výskyt chyb,
- postup při navrhování chybové rutiny obsahuje tři kroky:

1) Zapnout zachytávání chyb

- určíme, která část kódu se bude provádět při výskytu chyby za běhu programu (která chybová rutina bude aktivována),
- zachytávání chyb je zapnuto příkazem **On Error:**

```
On Error GoTo jméno_chybové rutiny
```

- zachytávání chyb je zapnuté v proceduře, ve které je tento příkaz uveden – tedy do doby, než jsou provedeny příkazy End Sub, Exit Sub, End Function, Exit Function, End Property, Exit Property,
- současně může být zapnut pouze jeden způsob zachytávání chyby,
- lze vytvořit různé chybové rutiny a zpřístupňovat je pomocí příkazu On Error,
- zachytávání chyb lze rovněž dodatečně vypnout použitím zvláštní podoby příkazu On Error:

```
On Error GoTo 0
```

2) Psaní rutiny pro obsluhu chyb

- prvním krokem je určení návěští řádku, kde bude chybová rutina začínat,
- návěští se píše ve tvaru:

```
jméno_chybové_rutiny:
```

- tělo chybové rutiny obsahuje kód ošetřující chyby,
- musíme určit, ke kterým chybám může dojít, a stanovit odpovídající způsob ošetření,
- vlastnost **Number** objektu **Err** obsahuje číslo představující nejběžnější chyby za běhu programu.

3) Opuštění chybové rutiny

- v závislosti na aktuálním stavu můžeme k opuštění chybové rutiny použít kterýkoliv z následujících příkazů:
- **Resume** – běh programu pokračuje na řádku, kde byla předchozí chyba vyvolána. Používá se tehdy, když byla příčina chyby úplně odstraněna.
- **Resume Next** – běh programu pokračuje na řádku následujícím za řádkem, v němž došlo k chybě.
- **Resume *line*** – běh programu pokračuje na řádku určeném parametrem *line*, což je návěstí řádku nebo nenulové číslo řádku, které musí být v té samé proceduře jako chybová rutina.

5.4 Užití předdefinovaných dialogových oken

- můžeme použít tyto funkce, které umožňují vložit do aplikace předdefinovaná dialogová okna:
    - InputBox – zobrazí výzvu a vrací cokoliv, co je uživatelem zadáno do textového pole.
    - MsgBox – zobrazí zprávu a vrací hodnotu, určující, na které tlačítko uživatel klepl myší.

Funkce InputBox

- vybídne uživatele k zadání nějakých dat,
- tato funkce zobrazí modální dialogové okno, které vyžaduje od uživatele zadání údaje,
- zapisuje se takto:

```
InputBox (Prompt,Titul) As String
```

    - *Prompt* – text, který se zobrazí v okně,
    - *Titul* – nadpis okna.
- funkce vrací údaje zadané uživatelem ve formátu *String*.

Funkce MsgBox

- slouží pro získávání odpovědí typu ano/ne od uživatele a pro zobrazení krátkých zpráv, jako jsou chybová hlášení, varování nebo výstrahy v dialogovém okně,
- po přečtení zprávy uživatel vybere tlačítko, kterým zavře okno,
- zapisuje se takto:

```
MsgBox (Prompt,Buttons,Titul) As VbMsgBoxResult
```

    - *Prompt* – text, který se zobrazí v okně,
    - *Titul* – nadpis okna,
    - *Buttons* – může například obsahovat tyto hodnoty:

cvičný 1

Konstanta:	Hodnota:	Význam:
MsgBoxStyle.OKOnly	0	Zobrazí jen tlačítko **OK**.
MsgBoxStyle.OKCancel	1	Zobrazí tlačítka **OK** a **Cancel**.

cvičný 2

MsgBoxStyle.AbortRetryIgnore	2	Zobrazí tlačítka **Abort**, **Retry** a **Ignore**.
MsgBoxStyle.YesNoCancel	3	Zobrazí tlačítka **Yes**, **No**, a **Cancel**.
MsgBoxStyle.YesNo	4	Zobrazí tlačítka **Yes** a **No**.

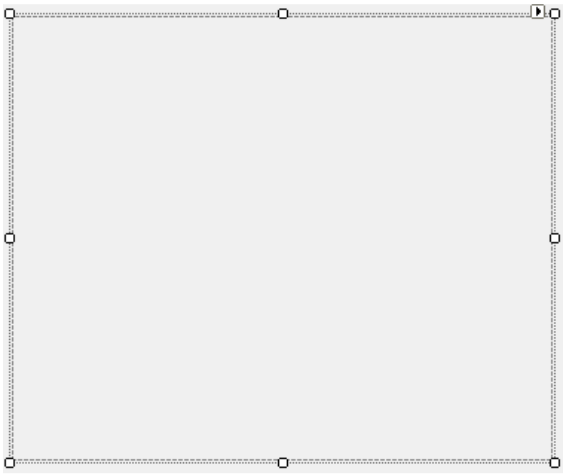
- funkce vrací hodnotu `MsgBoxResult`, kterou můžeme testovat jako číslo nebo konstantu dle následující tabulky:

Konstanta:	**Hodnota:**	**Tlačítko:**
vbOK	1	OK
vbCancel	2	Cancel
vbAbort	3	Abort
vbRetry	4	Retry
vbIgnore	5	Ignore
vbYes	6	Yes
vbNo	7	No

5.5 Ovládací prvky Image, Picture, DriveListBox, DirListBox, FileListBox

PictureBox

- slouží k zobrazování obrázků,
- aktuální obrázek je zadán do vlastnosti **Image**,
- načtení a zobrazení:
  `Name.Image = Image.FromFile(„Jmeno_obrazku“)`
- vlastnost **SizeMode** způsobí přizpůsobení obrázku ovládacímu prvku podle jeho velikosti (***StretchImage***).
- vlastnost **Location** nastavuje pomocí souřadnic X a Y umístění ovládacího prvku na formuláři.
- vlastnost **BackColor** umožňuje nastavit jednotnou barvu pozadí ovládacího prvku PictureBox,
- vlastnost **BackroundImage** umožňuje nastavit obrázek na pozadí ovládacího prvku PictureBox.
- Kreslení do ovládacího prvku PictureBox je v VB 2010 komplikované a proto se mu budeme podrobněji věnova až v následující kapitole.

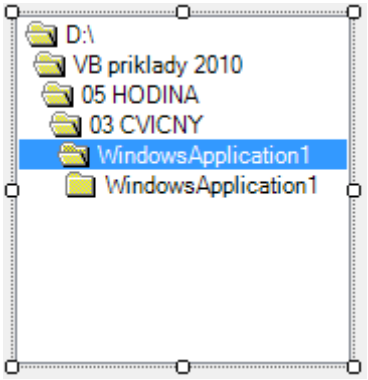


DriveListBox

- Umožňuje zobrazení jednotek připojených k danému počítači
- Bez kódu ukazuje svazek odkud je spuštěn program
- Příkazy: `Drive`
- Syntaxe: `DriveListBox1.Drive`

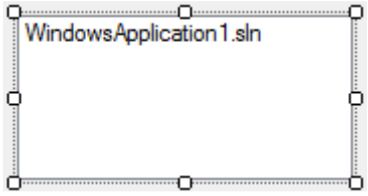
DirListBox

- Umožňuje zobrazení složek obsažených na disku
- Bez kódu ukazuje složku odkud je spuštěn program
- Příkazy: `Path`
- Syntaxe: `DirListBox1.Path`



FileListBox

- Umožňuje zobrazení souborů obsažených ve složce
- Bez kódu ukazuje soubory ve složce odkud je spuštěn program
- Příkazy: `Path, Pattern`
- Syntaxe: `FileListBox1.Path` a `FileListBox1.Pattern = „řetězec“`



5.6 Základní příkazy

Path a Drive

- První procedurou tedy bude synchronizace ovládacího prvku DriveListBox1 a DirListBox1.
- To znamená, že propojíme oba ovládací prvky tak, aby se podle typu vybrané jednotky automaticky měnil i seznam zobrazených složek.

- Stručně řečeno, podle toho, kterou jednotku vybereme, tak se změní i seznam složek, které jsou na dané jednotce obsaženy.
- K tomuto slouží příkaz PATH (cesta). Zde uvádíme jeho syntaxi:

***Objekt1*.Path = *Objekt2* .Drive**

- Jako Objekt1 můžeme použít volání ovládacího prvku (v našem případě DirListBox1).
- První objekt vždy zadává objekt, který bude přizpůsobován, v tomto případě je jasné, že se musí měnit seznam složek podle toho, která jednotka je vybrána.
- Potom následuje příkaz Path (nastav cestu, od názvu objektu je oddělen tečkou). Poslední částí příkazu je přirazení Objektu2, který je nadřazen Objektu1.
- V našem případě je to ovládací prvek DriveListBox1, protože podle toho, kterou jednotku zvolíme, tak se změní obsah seznamu složek.
- Za objekt se umisťuje klíčové slovo Drive, které je od Objektu2 odděleno tečkou. Toto klíčové slovo říká počítači to, že se jedná o seznam jednotek (Drive).

LoadFile

- příkaz pro načtení obrázku do ovládacích prvků PictureBox
- syntaxe:

```
Objekt.Image = Image.FromFile (název souboru_nebo cesta
k němu)
```

Spojování řetězců

- Spojení řetězců (konkatenace) se děje pomocí:
- **znaku & (AltGr + C)**
- syntaxe: *řetězec1 & řetezec2 & řetězec3 & ……. & řetezec$_n$*
- **znaku +**
- *řetězec1 + řetezec2 + řetězec3 + ……. + řetezec$_n$*

Pro spojení řetězců je obecnější operátor &, neboť ten na rozdíl od + umožňuje do konketance vložit číselné hodnoty a výrazy.