Diario di lavoro

Luogo	SAM Trevano
Data	14.11.2019

Lavori svolti

Oggi ho cominciato l'integrazione di quanto scoperto e testato nelle scorse lezioni riguardo la libreria per lavorare con documenti di Office (Office.Interop.Word). Per prima cosa ho dovuto aggiungere i seguenti riferimenti al progetto QDCeValutazioni:

- **Microsoft Office 16.0 Object Library (v. 2.8)**
- **InteropExtension 1.0 Type Library (v. 1.0)**
- **Microsoft Word 16.0 Object Library (v. 8.7)**

Dopodiché ho aggiunto anche alcuni using (per ora in QdcView):

```
using Word = Microsoft.Office.Interop.Word;
using System.Reflection;
using System.IO;
```

A questo punto prima di cominciare ho impostato dei placeholder a tutti gli elementi della view. Avevo già scoperto in una delle prime lezioni che non c'è una proprietà che permette di impostare i placeholder nei TextBox, quindi ho adottato un'altra soluzione. Ho impostato il placeholder nella proprietà **Text"<Placeholder>"** e ho impostato anche il colore del testo: **Foreground="Gray".** Poi ho creato due metodi, sugli eventi **MouseEnter** e **MouseLeave**:

```
private void PathTextBox_MouseEnter(object sender, MouseEventArgs e)
{
    if (PathTextBox.Text == "Path file")
    {
        PathTextBox.Text = "";
        PathTextBox.Foreground = Brushes.Black;
    }
}

private void PathTextBox_MouseLeave(object sender, MouseEventArgs e)
{
    if (PathTextBox.Text == "")
    {
        //if(PathTextBox.AreAnyTouchesCaptured == true)
        //{
            PathTextBox.Text = "Path file";
            PathTextBox.Foreground = Brushes.Gray;
        //}
    }
}
```

In questo modo quando il puntatore entra nel TextBox per la prima volta viene tolto il placeholder. Se il puntatore esce con il TextBox vuoto il placeholder viene reimpostato.
Finalmente ho iniziato con l'implementazione del codice per gestire Office. Per il momento gestisco questa parte nel metodo **OnClick** del pulsante salva (QdcView). Ho istanziato la variabile che contiene la path del file Word di template TemplateQdc, che viene inserita nel form:

```
var path = PathTextBox.Text;
```

e ho istanziato altre variabili che contengono il pattern da andare a modificare nel file di template:

```
var titoloKey = "<Titolo>";
var nomeFormatoreKey = "<NomeFormatore>";
. . .
var numeroOreKey = "<NumeroOre>";
```

creo l'oggetto che contiene l'istanza di Word:

```
Word.Application wordApp = new Word.Application();
```

Creo l'oggetto che contiene il documento e assegno la path a un oggetto:

```
Word.Document aDoc = null;
object filename = path;
```

Poi se il file esiste lo apro con il metodo Documents.Open(). In questo metodo definisco il file da aprire in modalità ReadOnly. Le proprietà che contengono missing sono proprietà che il metodo mette a disposizione ma che non mi servono. Poi viene richiamato il metodo FindAndReplace(), che riceve l'istanza di Word, il pattern da modificare e con cosa modificarlo. Infine salvo il file e chiudo il processo.
Se il file non esite semplicemente viene mostrato un messaggio di errore.

```
if (File.Exists((string)filename))
{
    object readOnly = false;
    object isVisible = false;
    wordApp.Visible = false;
    //  apertura del file copiato
    aDoc = wordApp.Documents.Open(ref filename, ref missing,
        ref readOnly, ref missing, ref missing, ref missing,
        ref missing, ref missing, ref missing, ref missing,
        ref missing, ref isVisible, ref missing, ref missing,
        ref missing, ref missing);
    aDoc.Activate();
    // richiama la funzione FindAndReplace passandogli due parametri,
    // il testo da sostituire e con cosa sostituirlo
    this.FindAndReplace(wordApp, titoloKey, TitoloTextBox.Text);
    this.FindAndReplace(wordApp, nomeFormatoreKey, NomeFormatoreTextBox.Text);
    . . .
    this.FindAndReplace(wordApp, numeroOreKey, NumeroOre.Text);

    //  salva il file
    aDoc.Save();
    ////////////////// IMPORTANTE, CHIUDERE IL PROCESSO ////////////////////////////
    wordApp.Quit();
}
```

Tutto questo codice è inserito in un blocco try-catch.

Il metodo FindAndReplace(<Word.Application>, <pattern da modificare>, <testo da scrivere>) contiene il metodo Selection.Find.Execute() che cerca il testo da modificare e lo sostituisce con il testo inserito dall'utente nel form.

```
private void FindAndReplace(Word.Application wordApp, object findText, object replaceText)
{
    wordApp.Selection.Find.Execute(ref findText, ref matchCase,
        ref matchWholeWord, ref matchWildCards, ref matchSoundsLike,
        ref matchAllWordForms, ref forward, ref wrap, ref format,
        ref replaceText, ref replace, ref matchKashida,
                ref matchDiacritics,
        ref matchAlefHamza, ref matchControl);
}
```

A questo punto ho fatto dei test su quello che ho scritto fin'ora. Il primo test non è andato a buon fine (vedi prossimo capitolo), dal secondo il risultato è stato ciò che mi aspettavo.
Ho aggiunto un ulteriore modifica dopo che mi sono accorto che il modo in cui ho implementato il sistema di scrittura sul file risultava piuttosto scomodo. Infatti veniva modificato lo stesso file di template (di cui comunque avevo una copia) e ogni volta per provare dovevo ritornare alla versione originale. Quindi ho aggiunto qualche semplice riga di codice:

```
File.Copy(path, pathCopia, true);
```

Questo metodo esegue un copia del file (pathCopia al momento è una variabile statica, in seguito dovrò aggiungere una TextBox alla view che permette di selezionare anche la path del file da salvare).
Ovviamente il file da aprire non è più il template originale ma quello copiato:

```
object filename = pathCopia;
```

Anche questo metodo è funzionante ed è più comodo rispetto al precedente.
Ho quindi aggiunto questa funzionalità anche a DescrizioneView.

Tutta questa parte al momento la gestisco nel metodo OnClick, quindi direttamente nella View. Siccome alla fine del progetto la scrittura avverrà in un'altra view dopo che saranno stati inseriti tutti i campi nelle varie Views, dovrò spostare il codice (o magari inserirlo in un Binding di un pulsante e generare un ViewModel specifico per il salvataggio dei dati su un file Word). Per ora continuo a gestirla in questo modo.
Alla fine siccome mi rimaneva qualche minuto ho cominciato ad aggiornare la documentazione con quanto svolto nelle ultime lezioni.

Problemi riscontrati e soluzioni adottate

Durante il primo test di scrittura dei dati dal form sul file Word ho notato che venivano scritti solo il titolo e il cognome del formatore, mentre tutti gli altri campi rimanevano quelli di default. Questo perché avevo generato il file di Template .docx direttamente da un pdf e alcune parti non erano state scritte correttamente. Quindi ho cancellato quelle parti e le ho riscritte a mano. Dopo aver fatto ciò i test hanno funzionato.
Dopo aver copiato il codice anche in DescrizioneView mi sono accorto che in quella view non è definito il percorso del file di Template. Per ora l'ho scritto in modo automatico ma dovrò trovare un sistema di ricavarlo dalla view QdcView (probabilmente aggiungerò un campo al database in modo che il campo possa essere a disposizione anche delle altre View).

Punto della situazione rispetto alla pianificazione

In recupero.

Programma di massima per la prossima giornata di lavoro

Sistemare tutti i placeholder e aggiungere una TextBox a QdcView. Il resto del tempo lo userò per aggiornare la documentazione su quanto svolto nelle ultime due settimane.