**פיתוח אפליקציה עם ממשק משתמש ב-WPF**

הגענו לשלב האחרון בפרויקט טריוויה,  
עד כה פיתחתם את מסד הנתונים ששומר את השאלות, התשובות, החדרים, ההתמודדויות ועוד, את צד שרת באמצעותו חושפים את המידע, את צד לקוח שצורך מהשרת את המידע וכל מה שנותר הוא לפתח את ממשק המשתמש.  
בטריוויה משחק המשתמש יהיה גרפי, מבוסס GUI (Graphics User Interface).  
ישנן הרבה מאוד ספריות גרפיות: Windows Forms הוותיקה, DirectX ו- OpenGL התלת ממדיות, qt שהיא Cross Platform ועוד.  
אנחנו מבנה את ממשק המשתמש בטכנולוגיה מעניינת שנקראת WPF Windows Presentation) Foundation).

החשיבות של ממשק המשתמש היא קריטית להצלחת מערכות תוכנה ובוודאי שגם למשחק מחשב.

במהלך הלימוד נתרגל WPF על ידי כתיבת אפליקציה פשוטה שתסייע לנו לזכור ימי הולדת של חברים, מכרים ומשפחה.  
התרגול יהיה בשלבים, לאחר כל נושא שנלמד נפתח את החלקים הרלבנטיים באפליקציית ימי ההולדת שלנו.

לאחר שתסיימו לפתח את אפליקצית ימי ההולדת, תגשו לפתח את ה-GUI של אפליקציית לקוח הטריוויה. אפליקציית לקוח הטריוויה מומשה בטכנולוגיה ישנה ל-GUI בשם Win Forms, ואתם תממשו אותה בטכנולוגיה החדשה של פיתוח אפליקציות עם ממשק משתמש בחלונות – WPF.

**פתיחה**

****

מערכות תוכנה בעלות פונקציונליות מדהימה נכשלות,  
מערכות אחרות שאינן מתפקדות כראוי לעיתים מצליחות מעל ומעבר,  
יש בזה היגיון?

התשובה היא כן, הרבה תלוי במילה אחת – "ידידותיות".  
המשתמש הסופי תמיד יעדיף מערכות שיהיה לו קל להשתמש בהן, מערכות שנעימות לעין, שלא צריך להשקיע הרבה בשביל ללמוד כיצד להשתמש בהן, מערכות שמגיבות ולא נתקעות.  
כל אלו תלויות בממשק המשתמש של המערכת.

שכבת ממשק המשתמש (UI- User Interface)   
שכבת התצוגה מורכבת מחלונות ופקדים באפליקציות חלונאיות או דפיHTML  באתרי אינטרנט.  
השכבה מקשרת בין המשתמש לבין שאר השכבות של המערכת.



לעיתים שכבת ממשק המשתמש מכונה "שכבה טיפשה",  
היא מכונה כך משום שאין בה כמעט אלגוריתמיקה (זה קיים ב- BL), היא לא מטפלת במסד הנתונים (זה מתרחש ב- DAL), כל תפקידה הוא לקשר בין המשתמש לבין המערכת, לאפשר למשתמש לבצע פעולות ולקבל תגובות.

קיימות הרבה תשתיות לפיתוח ממשקי משתמש, אנחנו נתמקד בתשתית הנקראת WPF (Windows Presentation Foundation).

**מבוא לתכנות חלונאי**

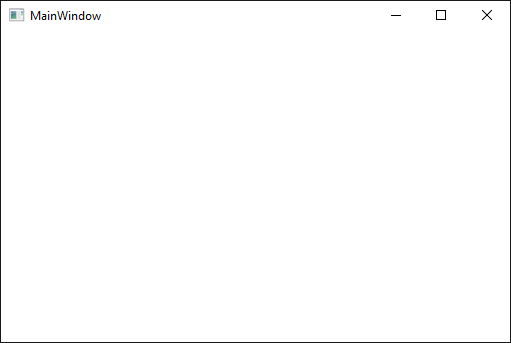
אולם, רגע לפני שנתמודד עם יכולת ממשק המשתמש שמציעה מערכת ההפעלה Windows עלינו להכיר את מודל התכנות החלונאי, אשר שונה במהותו מהמודל המוכר מאפליקציות Console Application.

האוריינטציה הגרפית (GUI - Graphic User Interface) של תוכנית הבנויה ל- Windows מכתיבה מודל תכנותי מבוסס אירועים הנקרא - Event Driven.

זהו מודל שונה מאוד במהותו מתכניות Console שבהם האפליקציה מתבצעת Top Down, שונהמאוד מהמודל שבו תוכנית ה- Console היא זו אשר יוזמת את הפנייה למערכת ההפעלה בעיקר באמצעות פונקציות הפלט/קלט.

המערכת הגרפית של Windows מכתיבה מודל תכנותי שונה במהותו, בו מערכת ההפעלה היא זו היוזמת את האינטראקציה עם התוכנית.   
האינטראקציה מתבצעת באמצעות שליחת הודעות (Messages) לתוכנית.   
ההודעות יכולות להישלח כתגובה לפעילות המשתמש (עכבר, מקלדת, סגירת חלון וכו') או כתוצאה מיוזמה של מערכת ההפעלה (טעינת חלון, הצגת החלון, שינויי פוקוס, שחרור החלון הריסת חלון וכו').

פעילות משתמש: עכבר, מקלדת או פעילות של מערכת ההפעלה: פתיחת חלון, סגירת חלון וכו'



התוכנית מטפלת באירוע או לחילופין מתעלמת ממנו

מערכת ההפעלה מתרגמת את הפעילות להודעה.

**Event Driven**

כאשר מריצים תוכנית Windows, מתבצעות מספר פעילויות אתחול שתפקידן להגדיר, לטעון לזיכרון ולהציג את החלון .

לאחר אותן פעולות אתחול התוכנית נכנסת למצב המתנה המכונה IDLE, במצב זה התוכנית אינה עושה דבר ובו היא נשארת עד אשר מתקבל אירוע (Event).

האירוע מעורר את התוכנית אשר מטפלת באירוע וחוזרת חזרה למצב IDLE עד להופעת האירוע הבא אשר יעורר אותה וחוזר חלילה, עד אשר התוכנית מסתיימת.

על מנת להבין כיצד מנגנון זה מתפקד יש להכיר מספר מושגים חדשים:

**אירועים – Events**

פעילויות המתרחשות בתוכנית או המיועדות לתוכנית. הפעילויות הללו יכולות להיות ביוזמת מערכת ההפעלה או המשתמש.

האירועים אשר מערכת ההפעלה יכולה ליזום הם טעינת חלון לזיכרון, הצגת חלון, רענון חלון, סגירת חלון, הזזת חלון, שינוי גודל החלון וכו'.

אירועי משתמש יכולים להיות אירועי מקלדת שונים, אירועי עכבר וכו'.

התוכנית תקבל הודעה (Message) על התרחשות אירוע זה או אחר ותפעל בהתאם.

התוכנית אינה חייבת לטפל בכל האירועים המוזרמים אליה ממערכת ההפעלה.

**שפת C#**

לרשותכם מצורף המאמר "יסודות שפת C#". תוכלו לעבור עליו עכשיו, או להשתמש בו כאשר תצטרכו אותו.

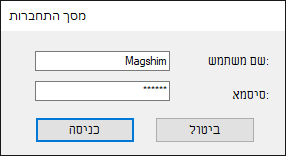
**WPF – אפליקציית ימי הולדת - תכנית עבודה**

על מנת ללמוד תכנות GUI לחלונות בעזרת WPF תממשו **אפליקציה לשמירת ימי הולדת של חבריכם בלוח שנה**.

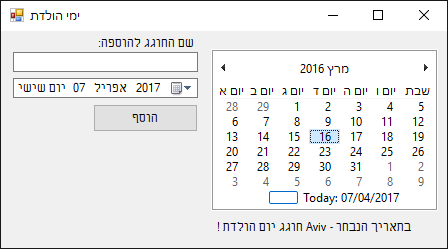
באמצעות האפליקציה תוכלו להזין שם של חבר ואת תאריך הולדתו. כשאר תבחרו תאריך מסויים תוכלו לראות מי מחבריכם חוגג יום הולדת.

האפליקציה תכיל שתי חלונות בלבד:

1. **חלון התחברות:**



1. **חלון ראשי:**



בצד ימין של החלון ניתן לבחור תאריך והמערכת תציג מי חוגג יום הולדת בתאריך הנבחר, צד שמאל של החלון נועד להוסיף תאריכי ימי הולדת נוספים.

לכל משתמש רשימת ימי הולדת חברים נפרדת בקובץ TXT שנושא את שם המשתמש.

כאמור, כדי לממש את האפליקציה תשתמשו בטכנולוגיית WPF שתוכלו לקרוא עליה באופן כללי כאן:

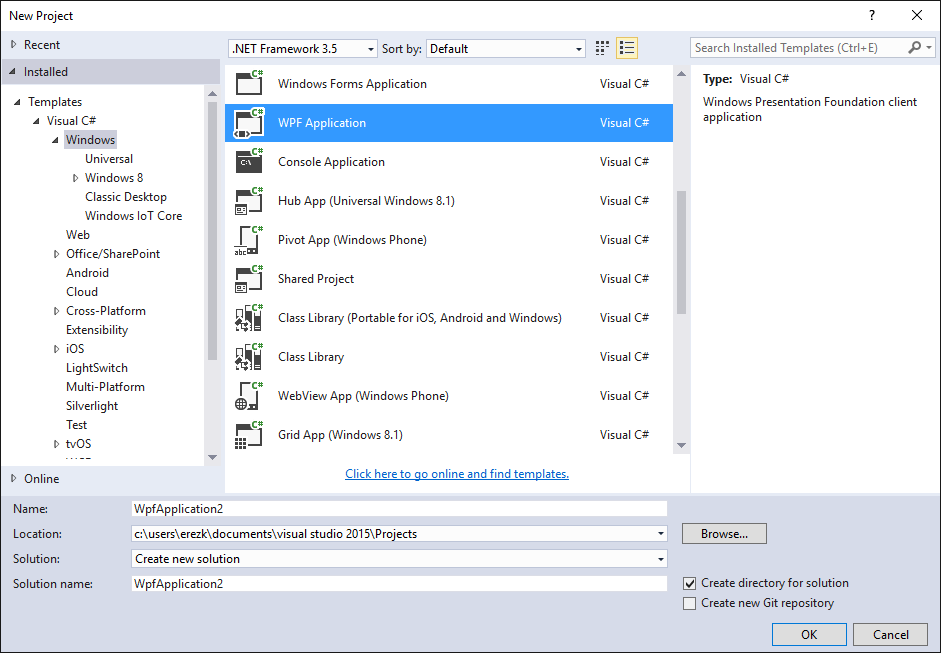
[מה זה WPF?](http://www.wpf-tutorial.com/about-wpf/what-is-wpf/)

כעת, תפתחו פרויקט WPF חדש, ותלמדו על איך מפתחים בעזרת הטכנולוגיה בפועל.

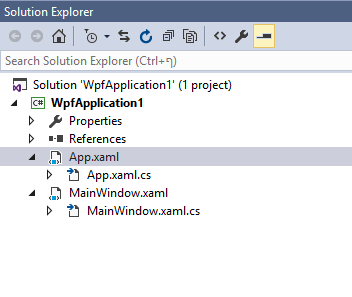
**פתיחת פרויקט WPF חדש**

בחלונית New Project של Visual Sudio נבחר:

Visual C# -> Windows -> WPF Aplication



**מבנה הפרויקט**



הקבצים **App.xaml** ו- **App.xaml.cs** הם קבצים ברמת האפליקציה, לדוגמה הם מגדירים מהו החלון הראשי שיוצג כאשר האפליקציה מוצגת, ניתן לטפל בהם בפרמטרים נוספים ברמת האפליקציה כמו במשאבים (Resources), סגנונות עיצוביים וכדומה.  
בפרויקט טריויה לא נעשה שינויים בשני קבצים אלו.

הקבצים **MainWindows.xaml ו-MainWindow.xaml.cs** הם הקבצים של החלון הראשי,

כל חלון ב- WPF מחולק לשניים:

* העיצוב הגרפי מבוסס בפורמט XML הנקרא XAML
* Code Behind בו נטפל באירועי החלון

בראשון מוגדר העיצוב הגרפי של החלון, בשני מוגדרים האירועים בהם נטפל.

תחילה נלמד XAML ונעצב באמצעותו את החלון, לאחר מכן נלמד כיצד לטפל באירועי החלון ב- Code Behind.

**קצת על XAML**

XAML (מבוטא זאמל)- קיצור של Extensible Application Markup Language היא שפת תגיות מבוססתXML שפותחה על ידי מיקרוסופט כשפה לייצוג, עיצוב והגדרה של אלמנטים גראפיים בממשק המשתמש של WPF.

כמו כל קבצי ה- XML, גם XAML היא היררכית וכך כמובן גם האובייקטים המוגדרים בה.

השימוש בשפה הורחב גם לטכנולוגיות אחרות מבית מיקרוסופט, כגון: Silverlight,  
 Windows 10 App, Windows Phone 10 APP.

היא מתבססת על DirectX כמנוע הגרפי שלה ויש לה יכולות גראפיות עשירות (אפקטים, אנימציה...)

היא מצמצמת את כמות הקוד הנדרש בהשוואה לטכנולוגיות UI אחרות

קיימת הפרדה ברורה בין העיצוב הגראפי של החלון לבין הלוגיקה של התוכנית.  
מה שמאפשר הפרדה ברורה בין תפקידי המתכנת לתפקידי הגרפיקאי.

כל אלמנט ב- XAML מתורגם לאובייקט ב- .NET Framework המכיל מאפיינים, מתודות ואירועים.

קובץ XAML פשוט ובסיסי לדוגמה

|  |
| --- |
| <Window x:Class="FirstWpfApplication2.MainWindow"        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"        Title="MainWindow"        Height="350"        Width="525">   <StackPanel>     <Label Content="Hello WPF" />     <Button Content="Hello World"            Width="130"            Height="30" />   </StackPanel> </Window> |

בקובץ XAML כל אלמנט מייצג מופע של מחלקה ב -NET. ,   
לדוגמא האלמנט Window הוא מופע של המחלקה System.Windows.Window, והאלמנט Button הוא מופע של המחלקה System.Windows.Controls.Button.

קביעת ערכים לAttribute של אלמנט קובעת את הProperty המתאים על המופע של המחלקה.

**קצת על Code Behind**

שפת XAML מאפשרת הפרדה טובה בין הלוגיקה של האפליקציה להגדרת ממשק המשתמש (UI).   
בדרך כלל, ליד קובץ XAML ישנו קובץ קוד בשפת #C, שמכיל קוד שנדרש ע"י קובץ הXAML, לדוגמא פונקציות שיקראו בעת לחיצה על כפתור. קובץ הקוד הנ"ל נקרא לעיתים code-behind file.

אם לקובץ הXAML שלנו קוראים MainWindow.XAML אזי קובץ הcode-behind יקרא: MainWindow.XAML.cs. ניתן לראות קובץ זה ב Visual Studio ע"י לחיצה על הסימן שמשמאל לקובץ הXAML

נניח ונרצה בקובץ ה code-behind לבצע שינויים על אלמנטים בUI, לדוגמא לשנות את גודל הפונט של הכפתור ל16. כיצד נוכל לעשות זאת?

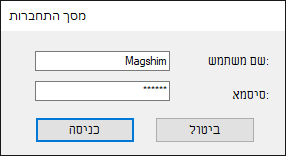
### הוספת שם לאלמנט

לצורך כך ראשית יש לתת שם לכפתור המבוקש. נעשה זאת ע"י הוספה של התכונה **x:Name** לכפתור באופן  הבא:

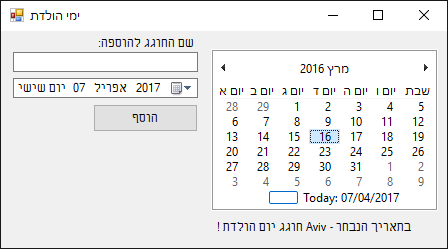
**משימה ראשונה – עיצוב החלונות**

**בנו את שני המסכים הבאים באמצעות XAML.**

**חלון התחברות:**



**חלון ראשי:**



בצד ימין של החלון ניתן לבחור תאריך והמערכת תציג מי חוגג יום הולדת בתאריך הנבחר, צד שמאל של החלון נועד להוסיף תאריכי ימי הולדת נוספים.

**מקורות מידע מומלצים (לפי הסדר):**

1. שלושת המאמרים הבאים על מנת להכיר את XAML:

* [What is XAML?](http://www.wpf-tutorial.com/xaml/what-is-xaml/)
* [Basic XAML](http://www.wpf-tutorial.com/xaml/basic-xaml/)
* [Events in XAML](http://www.wpf-tutorial.com/xaml/events-in-xaml/)

1. **פיתוח מודרך של אפליקציית ה-WPF הראשונה שלי!:** [Walkthrough: My First WPF Desktop Application](https://msdn.microsoft.com/en-us/library/ms752299(v=vs.110).aspx) (רק את החלק של העיצוב, עד " Adding Code to Handle Event" )
2. עימוד ועיצוב באמצעות Panels (צריכים להכיר רק את Grid ואת StackPanel)

## **ה זה Layout?**

המונח **Layout**מתייחס למדידה וסידור פקדים על גבי חלון האפליקציה שלנו. כלומר שליטה במיקום המדויק של כל פקד ופקד. זה כולל טיפול בהזזת הפקדים למיקומם הנכון (Move), הצמדה שלהם (Dock), טיפול בשינוי גודל החלון (Resize) ויישור (Align).

ב WPF יש טיפול נוח, פשוט ומשוכלל לטיפול ראוי ב Layout. ע"י התבססות על רעיונות מתוך עולם הHTML, מיקרוסופט הצליחה להפיק מערכת Layout חזקה מאוד ועם זאת, נוחה לתפעול.

פאנל (Panel) הוא פקד שיכול להכיל בתוכו מספר פקדים ויודע לסדר את הפקדים בצורה מסוימת. בפרקים הבאים במדריך נלמד על הפאנלים השונים הקיימים בWPF.

1. [Introduction to WPF Panels](http://www.wpf-tutorial.com/panels/introduction-to-wpf-panels/)
2. [The StackPanel](http://www.wpf-tutorial.com/panels/stackpanel/)
3. [The Grid](http://www.wpf-tutorial.com/panels/grid/)
4. [The Grid - Rows & Columns](http://www.wpf-tutorial.com/panels/grid-rows-and-columns/)
5. [The Grid - Units](http://www.wpf-tutorial.com/panels/grid-units/)
6. [The Grid - Spanning](http://www.wpf-tutorial.com/panels/grid-column-row-span/)
7. [The Grid - GridSplitter](http://www.wpf-tutorial.com/panels/gridsplitter/)
8. מבוא לפקדים (TextBlock, TextBox, Button, ListBox, Image)

WPF מכילה מספר רב של פקדים מוכנים מהקופסא. בפרקים הבאים נדבר על הפקדים הבסיסיים המסופקים בWPF. פקדים אלו הם יחידות הבסיס לכל תוכנה בסביבת Windows ועל כן חשובה מאוד הכרות טובה עימם.

1. [The Button Control](https://www.dotnetperls.com/button-wpf)
2. [The TextBlock control](http://www.wpf-tutorial.com/basic-controls/the-textblock-control/)
3. [The TextBlock control - Inline formatting](http://www.wpf-tutorial.com/basic-controls/the-textblock-control-inline-formatting/)
4. [The TextBox control](http://www.wpf-tutorial.com/basic-controls/the-textbox-control/)

[The ListBox control](http://www.wpf-tutorial.com/list-controls/listbox-control/)

**משימה שנייה – טיפול באירועים ב- Code Behind**

**הוסיפו ל-XAML ול-Code Behind  
של חלון ההתחברות – LoginScreen  
ושל המסך הראשי – MainWindow  
את האירועים הנדרשים  
הן של פקדי הלחצן  
והן של פקדי התאריך.**

**מקורות מידע מומלצים (לפי הסדר):**

1. [Delegates and Events](http://www.completecsharptutorial.com/basic/c-delegate-tutorial-with-easy-example/)
2. [WPF Events](http://www.wpf-tutorial.com/xaml/events-in-xaml/)
3. טפסים ומעבר ביניהם (כולל העברת מידע בין חלונות)
4. [The MessageBox](http://www.wpf-tutorial.com/dialogs/the-messagebox/)
5. [The Window](http://www.wpf-tutorial.com/wpf-application/the-window/)
6. מעבר בין חלונות

|  |
| --- |
| SecondWindow wnd = new SecondWindow();  wnd.ShowDialog(); |

1. העברת מידע בין חלונות – חפשו מידע על הדרכים בהם ניתן להעביר מידע בין חלונות באתר [StackOverflow](http://stackoverflow.com/search?q=passing+data+between+windows+in+wpf+)

**משימה שלישית - שמירה וטעינה של ימי הולדת בקבצי טקסט**

**ממשו את  
הלוגיקה של חלון ההתחברות (בעזרת Users.txt) ואת הלוגיקה של החלון הראשי  
(בעזרת <user>DB.txt)**

הפרויקט יכיל שני קבצי טקסט:

* Users.txt – שמות המשתמשים וסיסמה.
* <user>DB.txt – קובץ נפרד עבור כל משתמש עם שמות החברים וימי ההולדת שלו.

1. תככנתו את ה- Code Behind של חלון ההתחברות - LoginScreen:
   1. האירוע Click של הפקד "התחברות" בדקו האם המשתמש רשום, פרטי המשתמשים נמצאים בקובץ Users.txt, במידה וכן האפליקציה תציג את החלון הראשי – MainWindow במידה ולא תוצד הודעת שגיאה ב-MessageBox.
   2. הקובץ המצורף "Users.txt" מכיל רשימה של שמות משתמשים וסיסמאות. כל שורה בקובץ מכילה את המידע הבא: שם משתמש, סיסמא.

**לדוגמא**: Magshim,123456.  
כך ש"Magshim" הוא שם המשתמש ו"123456" היא הסיסמא. אפשר להניח שאין שני משתמשים עם שם זהה.

במידה והקובץ קיים כבר נטען אותו, במידה ולא ניצור קובץ חדש.  
כל שורה בקובץ נראית מהצורה: שם חוגג, תאריך,   
**לדוגמא**: Avi,01/02/03

**הערה**: בכל תאריך מסוים רק בן אדם אחד חוגג יום הולדת. אפשר להניח שהקובץ תקין ואין צורך לבדוק אותו.  
נקרא את הנתונים מתוך הקובץ אל תוך מבנה נתונים לבחירתנו ונציג אותם באמצעות הפקד המתאים.

* 1. האירוע Click של הפקד "ביטול" יסגור את האפליקציה.

1. תכנתו את ה- Code Behind של החלון הראשי - MainWindow:
   1. בעת פתיחת החלון (initialization) יתבצע קוד שיחפש את קובץ הנתונים המתאים עבור המשתמש שאליו התחברנו. לכל משתמש יהיה קובץ מיוחד שיקרא בשם - user\_name>BD.txt>

כך שלפי הדוגמא הקודמת עבור המשתמש "Magshim" נפתח את הקובץ ""MagshimBD.txt

* 1. הוספת חבר ויום הולדת: הקלידו שם של חבר ובחרו עבורו את תאריך יום הולדתו, האירוע Click של הלחצן "הוסף" יוסיף את הפרטים הנ"ל לקובץ MagshimimDB.txt.
  2. בחירה בתאריך בפקד Calendar תציג בתחתית המסך את החבר שחוגג יום הולדת בתאריך שנבחר (אם יש כמה חברים החוגגים באותו התאריך יוצג החבר הראשון שהוזן).  
     במידה ואף אחד לא חוגג הטקסט בתווית יהיה – "בתאריך הנבחר – אף אחד לא חוגג יום הולדת"

**משימות בונוס**

1. הוספת ימי הולדת:

נאפשר הוספת נתונים לקובץ. בחלק השני של החלון ניתן את האפשרות להזין שם חוגג ולבחור עבורו תאריך יום הולדת מתאים כך שכשנלחץ על כפתור "הוסף" תתווסף שורה חדשה בקובץ שפתחנו קודם עם הנתונים של יום ההולדת שהכנסנו. לאחר הלחיצה על כפתור ה"הוסף" ננקה את הפקדים מתוכן, ז"א נאפס את התאריך ונמחק את מה שהיה כתוב בתיבת הטקסט של שם החוגג להוספה.

לפני ביצוע ההוספה יש לבדוק שאין אף מישהו שכבר חוגג יום הולדת באותו תאריך. במידה ויש נציג הודעת שגיאה מתאימה ע"י MessageBox. כמו כן יש לשים לב שהשם נכתב באנגלית בלבד!

הערה:

יש לשים לב שתאריכי ימי ההולדת בקבצים מסודרים כך : M/D/YYYY

כך שהחודש מופיע לפני היום (לפי הגרסא הבינלאומית) יש לשים לב ששומרים על הפורמט הזה לכל אורך התרגיל. כדי לוודא שהעבודה לא תהיה תלויה בשפת מערכת ההפעלה תוכלו להשתמש לדוגמא בספריה:

using System.Globalization;

ובערך:

CultureInfo.InvariantCulture

כאשר אתם מפרשים את טקסט התאריך.

2. הוסיפו אפשרות להצגה ושמירה של יותר מחוגג אחד בכל יום.

3. הוסיפו אפשרות להוספת אירועים נוספים מלבד ימי הולדת.

4. שנו את קבצי הטקסט והקוד כך שהנתונים ישמרו בתבנית XML.

**נושאים נוספים ב-WPF לפרויקט טריוויה**

**על מנת לפתח את ה-GUI של לקוח פרויקט טריוויה, יש עוד שני נושאים שתצטרכו ללמוד. תוכלו ללמוד אותם כאשר תתכנתו את הלקוח.**

**נושא 1 - משאבים**

לעיתים קרובות האפליקציה צריכה להכיל קבצי מולטימדיה, סרטונים, אודיו.

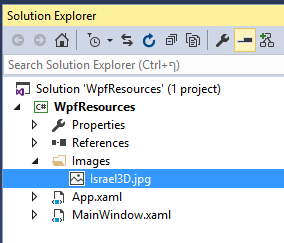
אפשרות ראשונה היא לשמור אותם כקבצים חיצוניים בספריית ההתקנה או בתת ספריה או בכל מקום אחר במחשב, אולם במקרה זה אנחנו מסתכנים שהמשתמש עלול למחוק אותם.

האפשרות המועדפת היא להוסיף אותם כחלק מובנה של האפליקציה, אפשרות זאת מכונה "משאבים" (Resources).

הסרטון הבא מציג כיצד מוסיפים משאבים לאפליקציית WPF:

[Adding Image to WPF C# application](https://youtu.be/affbgRZoeGc?list=PLS1QulWo1RIbTiFrgKi6CO5jVNDBNwHAG)

**הרחבה לסרטון, הצגת משאבים ב-Code Behind:**



XAML:

|  |
| --- |
| <StackPanel>  <Image Name="img" Width="400" Height="300" ></Image>  <Button Content="Button1" Name="btn" Click="btn\_Click"></Button>  </StackPanel> |

Code Behind:

|  |
| --- |
| public partial class MainWindow : Window  {  public MainWindow()  הצגת תמונה מ-Resource  {  InitializeComponent();  }  private void btn\_Click(object sender, RoutedEventArgs e)  {  img.Source=new BitmapImage(new Uri("/Images/Israel3D.jpg", UriKind.Relative));  }  } |

**נושא 2 – הוספה דינאמית של פקדים**

במקרים רבים למראה החלון משתנה במהלך הריצה, לעיתים נוסף מידע שרוצים להציגו, לעיתים נוספות אפשרויות למשתמש שנרצה שיוצגו, החלון יכול להיות דינאמי הן במראה והן בפונקציונליות,  
לדוגמה:  
בחלון המציג שאלון למשתמש, יתכן ובחלק מהשאלות המשתמש יצטרך לבחור תשובה אחת מבין 4, לעיתים יהיה עליו לבחור תשובה אחת מבין 5 ולעיתים עליו לסמן מספר משפטים שלדעתו מייצגים מתוך 10 משפטים שמוצגים לו.  
יהיו פקדים מסוגים שונים ובכמות שונה.

חלק מפקדי ה-Layout הכרנו בפרק המבוא ל-WPF (Grid. StackPanel).  
לפקדי ה-Layout יש מאפיין בשם Children, המאפיין Children הוא אוסף שיכול להכיל פקדים.  
כל הפקדים שנוסיף ב-XAML יתווספו אוטומטית למאפיין Children של פקד האב שלהם, לדוגמה:

|  |
| --- |
| <StackPanel>  <Image Name="img" Width="400" Height="300" ></Image>  <Button Content="Button1" Name="btn" Click="btn\_Click"></Button>  </StackPanel> |

המאפיין Children של ה-StackPanel יכיל את שני פקדי הבן: img ו-btn.

ב- WPF ניתן לייצר, להסיר ולשנות פקדים ואירועים ב-Code Behind במהלך ריצת התוכנית.

**XAML:**

|  |
| --- |
| <StackPanel Name="mainPnl">  <Button Content="Click Me" Name="btn" Click="btn\_Click"></Button>  </StackPanel> |

הוספה דינאמית של פקד ראשון

**Code Behind:**

|  |
| --- |
| private void btn\_Click(object sender, RoutedEventArgs e)  {  TextBlock txt = new TextBlock();  txt.Text = "Israel in 3D";  txt.FontSize = 30;  הוספה דינאמית של פקד שני  txt.Foreground = new SolidColorBrush(Colors.Red);  mainPnl.Children.Add(txt);  Image img = new Image();  img.Width = 400;  img.Height = 300;  img.Source=new BitmapImage(new Uri("/Images/Israel3D.jpg", UriKind.Relative));  mainPnl.Children.Add(img);  } |