תכנות בסביבת אינטרנט

מדריך לתלמיד
כולל דף הנחיות למורה

עבור סביבת ASPX

נבדק בסביבת ויז'ואל סטודיו 2015

משה שטיינר

moshe@steiners.co.il

"החקלאי" פרדס חנה

מרץ, 2019

הקדמה

מסמך זה מיועד לתלמידים המבצעים פרויקט בסביבת אינטרנט במסגרת היחידה השלישית במדעי המחשב.
מטרת המסמך להאיר ולהסביר את מהות התכנות בין צד לקוח לצד שרת, בצד עבודה עם מסד נתונים.
המסמך מיועד בעיקר לתלמידים, אם כי גם מורים יכולים להפיק ממנו תועלת בארגון הנושאים.
מסמך זה אינו מתווה את סדר ההוראה. לדוגמה, הטיפול בטופס במסמך זה מכסה גם צד לקוח, גם צד שרת, וגם פניה למסד הנתונים. לעומת זאת, בתהליך ההוראה ניתן לחלק את הטיפול בטופס ל-3 שלבים: צד לקוח בלבד. צד לקוח+שרת, וצד לקוח+שרת+מסד נתונים.

בנוסף כדאי לשים לב לכך שהמסמך אינו מסביר את עקרונות העבודה של כל מרכיב, אלא רק את אותם פרטים הנחוצים לעבודה.

בפרויקט בסביבת אינטרנט נלמד וניישם את עקרונות העבודה בין שרת ללקוח.
בחיים האמיתיים, בצד הלקוח נמצא מחשב אישי שולחני, לפטופ או טלפון חכם, ובצד השרת נמצא שרת המספק דפי אינטרנט וגישה למקורות מידע. דוגמאות לכך נמצאות לרוב, כמו שרת פייסבוק, שרת וואלה, שרת ynet, שרתי בנקים ועוד.
במקרה שלנו נשתמש בסביבת ויז'ואל סטודיו המספק אמולציה/מודל של שרת אינטרנט ושרת מסד נתונים, SQL. הלקוח יהיה נתון לבחירתנו: אחד הדפדפנים המותקנים במחשב.

הפרויקט כולל התנסות במספר "שפות". חלק מהשפות הינן שפת תגיות המתארות תכונות ולא באמת שפות תכנות. השפות בהן נשתמש:

1. HTML - עבור כתיבת דפים רגילים
2. CSS - עבור הגדרת סגנונות
3. Javascript - עבור הוספת דינמיקה לעמודים: בדיקות, סגנון משתנה
4. ASPX - עבור כתיבת דפים שניתן לכתוב עבורם קוד שיבוצע בצד שרת.
5. C# - כתיבת קוד צד שרת code behind
6. SQL - גישה למסד הנתונים

בתחילת הפרויקט, כל עוד נעסוק בצד לקוח בלבד, נעבוד בסביבת notepad++, על מנת להבין ולחוש טוב יותר את מבנה שפת ה-HTML. רק כשנעבור לטפל גם בצד השרת, נעבור לסביבת ויז'ואל סטודיו.

עבודה נעימה ובהצלחה

תוכן העניינים

הקדמה 2
דרישות מהפרויקט 4
בחירת נושא 7
Notepad++ 8
מבנה מסמך HTML בסיסי 9
מבנה דף HTML 9
הוספת קישורים 9
הוספת תמונות 11
אלמנטים מתקדמים ב-HTML 11
טבלה 12
תגיות נוספות 13
עיצוב - CSS 14
מודל הקופסה 16
מסגרות 16
פונטים 16
גודל 17
מיקום 17
מיקום יחסי 17
דף רישום משתמש חדש 19
javascript לשירות בדיקת הטופס 21
בדיקת נתונים 22
מעבר מ- notepad++ לויז'ואל סטודיו 25
עבודה עם שרת 25
עבודה עם דפי מאסטר - master page 25
עוברים לויז'ואל סטודיו 27
העבודה בצד השרת - CodeBehind 30
אובייקטים לשימוש השרת (Request, Response, Session, Application) 31
מסד הנתונים - database 34
סוגי השאילתות 35
בנית מסד הנתונים בסביבת ויז'ואל סטודיו 36
בנית שאילתה בצד השרת 42
DAL – והפניה למסד הנתונים 43
הכנסת פרטי משתמש חדש 44
כניסה ויציאה - login / logout 45
דף מנהל 46
דף עדכון נתוני משתמש 49
תיעוד הפרויקט - תיק פרויקט 50
נספח: DAL 52
מדריך למורה 54

דרישות מהפרויקט

את הדרישות מהפרויקט מומלץ לקרוא לפחות 3 פעמים:

- לפני תחילת העבודה
- במהלך העבודה
- לפני ההגשה הסופית

1. **נושא** - על האתר שלכם לשקף נושא כלשהו. לא תהיה התייחסות לנכונות התוכן, אבל כן תהיה התייחסות לאופן שבו אתם מגישים ומנגישים מידע וגישה למבקרים באתר.
2. **עמוד הבית** צריך להיות ייצוגי. לשקף את נושא האתר. רצוי גם שיכיל את שמכם. כאן הכל מתחיל...
3. בכל עמוד באתר, במקום קריא, יש לרשום **"שלום <משתמש>"**, כאשר <משתמש> הוא שמו האמיתי (ולא האימייל) של המבקר שנכנס לאתר שלכם, או המילה 'אורח' אם המשתמש אינו רשום או לא התחבר..
4. בעמוד הבית צריכה להיות גישה לכניסה/התחברות לאתר (login **או** sign-in). כלומר הכנסת אימייל וסיסמה. לאחר אימות הפרטים מול מסד הנתונים, המשתמש מקבל הרשאות של משתמש רשום. וגם שמו יופיע בכל הדפים.
5. בעמוד הבית צריך גם להופיע קישור לדף הרשמה לאתר. דף בו כל אחד יכול למלא פרטים ולהירשם.
6. קיימים 3 סוגי משתמשים: מזדמן/אורח, רשום, מנהל. מנהל הוא משתמש רשום עם זכויות של מנהל האתר. לכל אחד מסוגי המשתמשים יש זכויות שונות:
   a. מזדמן אינו יכול לראות את כל הדפים (רק את דף הבית, דף הרישום, אפשר גם להוסיף דף מידע "לגרות" את המשתמש).
   b. משתמש רשום יכול להגיע לדפי מידע נוספים.
   c. מנהל יכול להגיע לדף ניהול משתמשים. עבור משתמש מסוג מנהל תופיע אינדיקציה מנהל או admin ליד שמו של המשתמש בראש הדף.
7. **דף הרשמה** לאתר צריך לכלול טופס איסוף נתונים מהמשתמש. יש לבדוק ב-javascript נכונות ראשונית של הנתונים. בכל מקרה של אי עמידה בבדיקות, הטופס לא צריך להגיע לשרת להמשך טיפול.

   נכונות ראשונית כוללת:
   a. בדיקת הכנסת נתונים במקום שחובה להכניס
   b. בדיקת סיסמה, למשל האם ארוכה מספיק? האם כוללת אותיות וספרות.....
   c. בדיקת אימות סיסמה: האם האימות זהה להכנסה הראשונית.
   d. בדיקת אימייל: הכולל '@' ולפחות '.' (נקודה) אחת אחרי ה-'@'
8. בזמן רישום משתמש חדש יש לוודא שאין משתמש עם אימייל כזה במסד הנתונים.
9. **עדכון פרטי רישום** - משתמש שנכנס למערכת (login) יוכל לעדכן את פרטי הרישום שלו. לפחות אפשרות לעדכן את שמו ואת הסיסמה. מומלץ לא לאפשר שינוי אימייל. הקישור לדף עידכון הפרטים יהיה זמין רק למשתמש רשום.

   ניתן להחליף את הקישור להרשמה בקישור לעידכון פרטים.
10. לאחר שמנהל נכנס למערכת תהיה לו גישה לדף ניהול. דף הניהול יציג את טבלת כל המשתמשים במסד הנתונים. אסור להציג סיסמאות. למנהל תהיה יכולת לבצע את השינויים הבאים:
    a. מחיקת משתמש

b. שינוי מצב מנהל למשתמש
c. שינוי סיסמה למשתמש ששכח אותה (קביעת סיסמה אחידה, למשל 12345)

<u>שימו לב</u>: למנהל <u>אסור</u> למחוק את עצמו או לשנות את סטטוס הניהול של עצמו

10. **הגנה על דפים**. דפים המיועדים לסוג מסוים של משתמשים יוגנו בפני כניסה לא מאושרת.
11. יש לאפשר ביצוע של התנתקות מהמערכת (logout), הגורם לאיפוס ערכי ה-Session

דרישות נוספות

בנוסף למתואר למעלה, יש לדאוג לנושאים הבאים:

1. האתר חייב לעבוד!
   כל הקישורים; כל הקוד בסי-שארפ; הגישה והשליפה ממסד הנתונים. ללא התרסקות של הקוד.
2. לאורך כל הדפים, יש להראות שימוש נכון ומושכל בסגנונות (CSS) ובדף מאסטר (רשות).
   לא תהיה הערכה לאיכות הסגנון (צבע, צורה, טעם אישי). יש הערכה להבנה כיצד השימוש ב-CSS עובד.
3. תיק פרויקט כפי שהוסבר בנפרד.
4. שליטה מליאה בכל הפרויקט.
   שליטה כוללת הסבר איך דברים קורים, וגם היכולת לבצע שינויים בפרויקט על פי דרישה.

בונסים

כל נושא וכל מימוש מעבר למה שהוזכר למעלה יזכה בהערכת בונוס.

דוגמאות: דפי משחק. טריויה.

שימוש במסד הנתונים לצרכים נוספים מעבר לרישום המשתמשים. טיפול בתחביבים, והתייחסות אליהם. וככל העולה בדעתכם. בכל מקרה הציון המירבי לפרויקט הינו 100.

מחוון הערכה

פרויקט שאינו עובד - לכל היותר 50%

קריסות של הפרויקט בזמן ההגשה, שאינן מתוקנות במעמד ההגשה: -15%

נושא	ניקוד
תיק פרויקט - תיעוד הפרויקט	10
דפי תוכן: דף בית, דף לאורח, דפי משתמשים	10
כניסה, יציאה, "שלום אורח"	5
שימוש נכון באובייקטים: Response, Request, Session	10
שימוש נכון ב-database לצרכי רישום, זיהוי, ניהול, ועוד	10
הגנה על כניסה לדפים למורשים בלבד	10
קוד לבדיקת נכונות נתונים בצד לקוח (javascript או אחר)	10
עיצוב אחיד בעזרת שימוש ב-CSS. שימוש נכון ב-MasterPage	5
שליטה בפרויקט. הבנה של מה מתבצע והיכן	20
שינויים והרחבות במסגרת הגשת הפרויקט	10
בונוס: תוספת ייחודית ובעלת משמעות המשתמשת גם ב-database	10

בחירת נושא

נושא האתר הינו בחירה אישית. הנושא עצמו, תכולתו והדיוק בפרטים אינם עומדים למבחן. התלמידים יכולים לבחור כל נושא הקרוב לליבם ובתנאי שיהיה ראוי ולא פוגע בזולת.
מטרת בחירת הנושא לשוות לפרויקט האינטרנט נופך של סביבה אמיתית. במרבית המקרים התלמידים נהנים הרבה יותר כשבאתר מופיע תוכן שהם בחרו.
אופן הצגת הנושא, הפרטים, סגנון ההצגה, אסתטיקה – כל אלו אינם עומדים למבחן, אלא אך ורק למבחן השכל הישר. למשל: פונטים בגודל קריא, טקסטים שאינם "בורחים" מהמסך, וכיו"ב.
להלן מספר רעיונות לנושאים שתלמידים בחרו:

1. ענפי ספורט
2. סגנונות מוסיקה
3. אופנה
4. בישול
5. תנועת נוער
6. רכבים
7. אופנועים
8. אופניים
9. עשה במו ידיך (DIY)
10. קסמים
11. משחקי מחשב
12. גאדג'טים
13. הקוביה ההונגרית
14. סדרות טלויזיה

ועוד, ועוד.....

Notepad++

את תחילת הפרוייקט נבצע ב-++notepad. מומלץ להוריד למחשב הביתי את ++notepad (או בקצרה npp).
ההתקנה פשוטה ומהירה ואינה דורשת משאבים מיוחדים.
היתרון של npp שהוא "מכיר" את הסינתקס של השפות בהן נתעסק, ולכן יצבע לנו בצבעים מיוחדים מילות מפתח. בנוסף, ניתן להריץ דפי אינטרנט ישירות מ-npp.

בפעם הראשונה שנפתח מסמך חדש, נבחר באייקון השמאלי ביותר:

ועוד **לפני שנתחיל** לכתוב נשמור את הקובץ החדש על ידי בחירת: ...File->Save As

נא לבחור שם קובץ באנגלית **וללא רווחים**.

סיומת הקובץ חייבת להיות **html** או **css** בהתאם למה שנחוץ.

באופן זה, ++notepad יכיר את המילים השמורות, והמבנה של html.

מבנה מסמך HTML בסיסי

ברוכים הבאים לאתר שלי - יצאוני לדרך

מבנה דף HTML

שפת ה-HTML היא השפה בה כתובים דפי האינטרנט. זוהי שפת תגיות אותה מבינים הדפדפנים: כרום, אקספלורר, סאפרי, פיירפוקס ועוד...
שפת התגיות מכילה הנחיות לדפדפן כיצד להציג את תכולת הדף. קובץ HTML הוא קובץ ששמו מסתיים ב-.html או .htm והוא כתוב במבנה מסוים.

מבנה בסיסי של דף HTML:

```
<!DOCTYPE html>
<html>
<head>
        <title>כותרת הדף בדפדפן</title>
</head>
<body>
        <h1>Hello World Page</h1>
        <p>תוכן העמוד</p>
</body>
</html>
```

עוד על מבנה דף ה-HTML ניתן ללמוד במדריך ה-HTML
על מנת להריץ את הדף, יש לשמור את השינויים האחרונים וללחוץ על הרץ/Run בסרגל הכלים (יש אפשרות לבחור את הדפדפן האהוב עליכם).

> מרבית הדפדפנים מאד סלחניים לשגיאות תחביריות ב-HTML, וזאת על מנת לא לעצור את הצגת הדפים. ולכן רק לעיתים נדירות הם יעירו לנו על שגיאות בזמן ההרצה. התוצאה עלולה להיות בהצגת העמוד לא בדיוק כפי שרצינו.
> לכן: עלינו להיות מאד זהירים בזמן הכתיבה: לסגור תגיות, לבדוק ששם התגית מאוית נכון, וכיו"ב

הוספת קישורים

ישנם 3 סוגי קישורים עיקריים:
1. קישורים חיצוניים
2. קישורים מקומיים
3. קישורים פנימיים

קישורים חיצוניים

קישורים מהאתר שלי לאתר אחר. נדרשת כתובת אינטרנט מלאה. למשל:

YNET

אתגר: חפשו כיצד ניתן ללחוץ על קישור לאתר חיצוני שיפתח בחלון/טאב אחר

קישורים מקומיים

קישורים מדף לדף בתוך האתר שלי. הקישור יהיה יחסי למיקום של הדף הנוכחי.

1. כאשר שני דפים נמצאים באותה תיקייה, כל שנדרש הוא שם הדף אליו רוצים לעבור. נניח ו-first.html ו- second.html נמצאים באותה תיקייה, אזי קישור ב-first.html ל-second.html יראה כך:

לדף השני

2. נניח שבתיקיה מסוימת יש קובץ בשם first.html ובנוסף יש תת תיקייה בשם subfolder. בsubfolder יש דף בשם second.html, אז הקישור ב-first.html ל-second.html יראה כך:

לדף השני

3. נשתמש בדוגמה מסעיף 2 רק שהפעם אנחנו מעוניינים בקישור ב-second.html ל-first.html. במקרה זה הקישור יראה כך:

לדף הראשון

המשמעות של ".." היא "לעלות תיקייה אחת למעלה" בהיררכיה

קישורים פנימיים

קישור לאזור אחר בתוך הדף בו נמצאים.
יש לתת שם ל"אזור" אליו רוצים לקפוץ. האזור יכול להיות פסקה, כותרת, תמונה, או כל אובייקט אחר בדף.
השם ניתן על ידי התכונה id.
למשל נניח שיש לנו דף שבו סיפור עם חמישה פרקים ואנחנו מעוניינים ליצור תוכן עניינים בראש הדף שבו בלחיצה על שם הפרק נקפוץ לתחילת הפרק בעמוד הנוכחי. זה יעשה כך:

לכותרת של הפרק השלישי למשל נוסיף את התכונה id עם שם מתאים:

<h2 id=”ch3”>פרק שלישי</h2>

ואילו בראש הדף בתוכן העניינים יהיה קישור לפרק השלישי באופן הבא:

פרק שלישי

המשמעות של # היא קישור בתוך הדף.

הוספת תמונות

ניתן להוסיף תמונות על ידי שימוש בתגית img באופן הבא:

```
<img src="smiley.gif" alt="Smiley face" height="42px" width="42px">
```

כאשר src הוא שם הקובץ המכיל את התמונה להצגה (יש לשים לב לנתיב הקובץ יחסית למיקום הדף).
במידה ונגדיר את alt, אזי טקסט זה יוצג אם הקובץ אינו נמצא.
רוחב ו/או גובה הם תכתיבים לדפדפן לגבי רצוננו מגודל התמונה. השמטת מאפיינים אלו יגרמו לדפדפן להציג את התמונה בגודלה הטבעי/המקורי. אם נציין רק מימד אחד, המימד השני ייקבע כך פרופורצית התמונה תישמרנה. כלומר, ציון שני המימדים עלול לעוות את התמונה. ניתן להשתמש באחוזים במקום בגודל אבסולוטי.

אלמנטים מתקדמים ב-HTML

רשימות

ניתן להגדיר שני סוגי רשימות ב-HTML: ממוספרות וללא מספרים:

רשימות ממוספרות

התגית ol - ordered list מציינת רשימה ממוספרת. כל פריט ברשימה יצויין על ידי li - list item.

הדוגמה הבאה:

```
<ol>
<u>רשימה ממוספרת</u>
        <li>קפה</li>
        <li>תה</li>
        <li>חלב</li>
</ol>
```

תייצר את הפלט:

<u>רשימה ממוספרת</u>

1. קפה
2. תה
3. חלב

רשימות לא ממוספרות

התגית ul - unordered list מציינת רשימה ממוספרת. כל פריט ברשימה יצויין על ידי li - list item.
כלומר, כל ההבדל בתגית המגדירה את הרשימה. הדוגמה הבאה:

```
<ul>
<u>רשימה לא ממוספרת</u>
        <li>קפה</li>
        <li>תה</li>
        <li>חלב</li>
</ul>
```

תייצר את הפלט:

<u>רשימה לא ממוספרת</u>

- קפה
- תה
- חלב

<u>טבלה</u>

טבלה היא אחד הכלים הנוחים ב-HTML להצגת נתונים או טפסים בצורה מסודרת. בין אם נרצה להציג קווים מפרידים (מסגרות) ובין אם לאו, הטבלה מסייעת לנו לערוך עמודות באופן מסודר אחת מעל השניה מבלי שנצטרך "לחשב" את רוחב הטקסט המוצג. גודלי התאים בטבלה מחושבים באופן דינמי בהתאם לתא הרחב ביותר, אלא אם כן נבקש אחרת, ועל כך בפרק העיצוב.
טבלה ב-HTML מורכבת משורות ובכל שורה יש תאים. אין חובה שמספר התאים בכל שורה יהיה זהה! אבל: כל התאים הראשונים יוצגו זה מעל זה, וכך גם כל התאים השניים, וכך הלאה. כלומר, שורה שיש בה יותר תאים משורות אחרות, תאים אלו יוצגו בהמשך השורה.
הגדרת טבלה תיעשה באמצעות התגית table. שורה מסומנת על ידי tr - table row. תא בטבלה מסומן על ידי td - table data.
נסו לראות מה קורה עבור הקוד הבא:

```
<table>
        <tr>
                <th>עמודה 1</th>
                <th>עמודה 2</th>
                <th>עמודה 3</th>
        </tr>
        <tr>
                <td>111</td>
```

```
            <td>222</td>
            <td>333</td>
        </tr>
        <tr>
            <td>444</td>
            <td>555</td>
            <td>666</td>
        </tr>
</table>
```

תגיות נוספות

חשוב לזכור שב-HTML אין חשיבות לרווחים (כפולים) או לשורות חדשות. מעבר שורה ב-HTML אינו מתורגם למעבר שורה בתוצאה המוצגת. על מנת לאלץ מעבר שורה ניתן להשתמש בתגית
 או <p>. באופן דומה אין משמעות לרווחים בקובץ ה-HTML. הדפדפן "מצמצם" רווחים לרווח בודד. אם נרצה לאלץ רווחים, נשתמש בסימון: .

<h1></h1> - כותרת. קיימות כותרות עד לרמה 6: <h6>

<center></center> - מירכוז התוכן בחלון

<div></div> או **<section></section>** - בלוק "לוגי" שניתן לייחס לו תכונות עיצוב משותפות.

**
** - מעבר לשורה חדשה

<p></p> - פיסקה

** ** – הצגת רווח

<hr> - קו אופקי

משימה

הגדירו לפחות 4 דפי HTML. בכל דף הגדירו כותרת, וקישורים לכל יתרת הדפים. ניתן להוסיף הגדרות גם לקישור לאתרים חיצוניים. בכל דף הוסיפו לפחות תמונה אחת. את כל התמונות יש למקם במחיצה נפרדת מהדפים.

עיצוב - CSS

CSS הוא קיצור של Cascading Style Sheet. זוהי הדרך לעצב את העמוד כרצוננו, כולל, צבעים, מיקום, סוג פונטים, תמונות, רקעים ועוד. מגוון האפשרויות גדול מאד. בפרק זה נעמוד רק על כמה דוגמאות קטנות. ניתן למצוא הסברים ודוגמאות רבות ב-w3school.

ניתן לשלב הוראות CSSב-3 מקומות שונים, המהווים 3 רמות עדיפות שונה.

Inline – ניתן לשלב לכל תגית מאפיין מסוג style שיכיל הוראות סגנון. זו מחרוזת המופרדת על ידי ';' בין ערך לערך. לדוגמה:

```
<div style="background:cyan;border-style:solid;">
```

בתוך המחרוזת יש לנו זוגות של שם הפרמטר, אחריו נקודותיים (:) הערך, ולאחריו נקודה פסיק (;).

Internal – בתוך קובץ העמוד עצמו ניתן לשלב תגית בשם style ובתוכה פירוט הסגנונות. לדוגמה:

```
<style>
body  {
      background-color: powderblue;
}
h1    {
      color: blue;
}
p     {
      color: red;
}
</style>
```

המשמעות במקרה זה שעבור אותו עמוד, לכל איזור ה-body יש רקע בצבע מסוים. לכל הכותרות מסוג h1 יהיו צבע כחול.

External – קובץ חיצוני המכיל את כל ההגדרות באופן מאד דומה ל-internal. היתרון הוא שלאותו הקובץ ניתן להתייחס ממספר עמודים רב, ואין צורך לשנות את הסגנון בכל עמוד בנפרד.

הערות

1. כמו HTML וכמו javascript, שגיאות תחביריות בסגנון יגרמו להתעלמות והפסקת קריאת הסגנון באותו הקובץ, וכל זאת ללא כל הודעת שגיאה....
2. כל אובייקט "יורש" מהאובייקט ההורה את התכונות שלו, אלא אם כן הם שונו בהוראות CSS

קדימויות

אם עבור פרמטר מסוים התקבל יותר מערך אחד, האחרון קובע.

אם לתגית מסוימת יש התייחסות סגנונית ביותר מאופן אחד, יתקיימו סדרי העדיפות הבאים:

1. Inline
2. Internal
3. External

קישור לקובץ חיצוני

נהוג לרשום בחלק ה-head של הקובץ, והוא נראה כך (במודגש שם הקובץ שלכם):

```
<link type="text/css" rel="stylesheet" href="styles.css" />
```

שיוך סגנון על פי מזהה

ניתן להתייחס למזהה של כל אובייקט ב-HTML ולקבוע לו סגנונות, בהנחה שהם אפשריים עבורו. נשתמש בשם המזהה עם הקידומת סולמית, למשל #main :

```
<div id="main">
```

אזי בקובץ הסגנון נוכל להתייחס לבלוק זה בשני אופנים:

```
div {
	background-color: blue;
	color:white;
}
#main{
	background-color: red;
	font-size:20px;
	direction:rtl;
}
```

כלומר, התקבלו הוראות סגנון גם דרך הפיסקה div וגם רך הפיסקה #main. עבור תכונה שיש לה כפילות בערכים, האחרון קובע.

שיוך סגנון על פי class

ניתן להתייחס ל-class על מנת לאפיין מספר איזורים בעמוד עם אותו הסגנון. למשל אם נרצה בתוך טבלה סגנון מסוים לחלק מהתאים, וסגנון אחר לחלק אחר של התאים:

```
<td class="tdRed">……</td>
<td class="tdBlue"> ……</td>
```

בקובץ הסגנונות נראה את ה-class בפיסקה המתחילה בנקודה:

```
.tdRed{
	color:red;
}
tdBlue{
	color:blue;
}
```

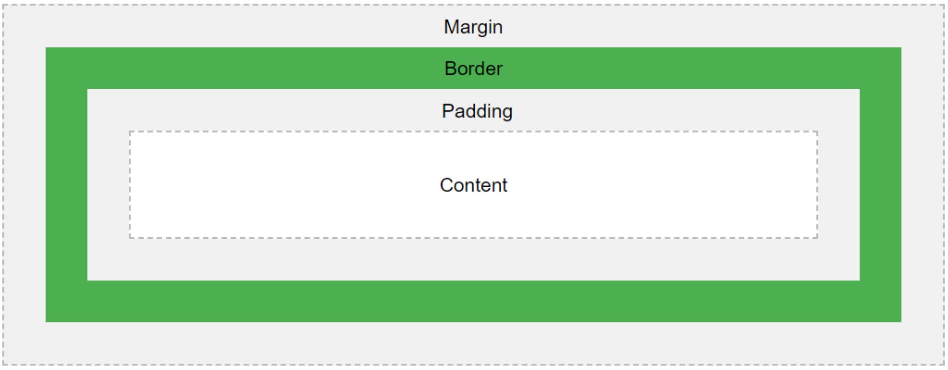
היתרון כאמור הוא בכך שניתן להשתמש ב-class ללא הגבלה.

ניתן להיות יותר ממוקדים ולאפיין תגית מסוג מסוים השייכת לבלוק מסוים לדוגמה, נניח שיש לנו בלוק שהמזהה שלו block1, ובתוכו יש תגיות מסוג a. נוכל לשנות את צבע הפונטים של תגיות a אלו בלבד, ולא את האחרות:

```
#block1 a{
	color:green;
}
```

מודל הקופסה

כל אובייקט נתון בתוך הקופסה וירטואלית. תוכלו לקרוא עוד על מודל הקופסה כאן.



על פי מודל הקופסה לכל אובייקט ניתן להגדיר:

1. מסגרת – border
2. שוליים פנימיים (בין התוכן למסגרת) – padding
3. שוליים חיצוניים (בין המסגרת לאובייקט הסמוך) – margin.

לכל אחת מהתכונות הנ"ל יש ערך ברירת מחדל, אותו ניתן לשנות.

ניתן להתייחס לכל ההיקף, או לכל צד בנפרד. למשל ;padding:15px מגדיר שוללים ברוחב אחיד מכל הכיוונים. לעומת ;padding-right:40px המגדיר ערך רק עבור צד ימין.

הדפדפן ישתדל למלא אחר כל ההנחיות אך זה אינו מובטח. למשל אם נקבע שגובהה של הקופסה 200, השוליים 10, והטקסט הוא של שורה בודדת, אזי ברור שבכיוון מטה לא נוכל לעמוד בדרישה, ויהיו שוליים גדולים יותר.

מסגרות

לכל אובייקט ב-HTML ניתן להגדיר מסגרת. למסגרת מספר תכונות:

צורת המסגרת – border-style: רציף – solid, מקוקו – dashed, נקודות – dotted, ללא קו – none, ועוד

צבע המסגרת – border-color

עובי המסגרת – border-width

רדיוס הפינה – border-radius

וכמובן שניתן להתייחס לכל צד בנפרד.

השימוש במסגרת נוח לא רק למטרות עיצוב, אלא גם כשמחפשים תקלות. המסגרת תראה לנו איפה נמצא הבלוק באופן יחסי למה שרצינו/חשבנו.

פונטים

נושא הפונטים הוא עדין. הדפדפן משתמש בפונטים הנמצאים במחשב הלקוח, אלא אם כן האתר/שרת שולח ללקוח הגדרת פונטים חדשה. ולכן לא מומלץ להוריד פונטים שאינם סטנדרטים. הם ייראו רק במחשב אליו הורדנו אותם.

לעומת זאת ניתן להשתמש בציון משפחת הפונטים. כל פונט משתייך למשפחה. שתי המשפחות הכי נפוצות הן:

```
font-family:serif;
font-family:sans-serif;
```

serif אלו פונטים מעוטרים, שלעיתים קשים לקריאה על גבי מסכים. למשל Times Roman

sans-serif אלו פונטים ללא עיטורים כדוגמת Arial.

ציון המשפחה יגרום לדפדפן לבחור פונט הכי קרוב למה שביקשנו.

גודל הפונט יכול להיקבע במילים או מספרים: small, large, x-large, xx-large **ועוד.**
במספרים יש שתי שיטות: מידה בפיקסלים (נקודה על המסך): 20px, 24px
או ביחידות מיוחדות לפונטים שהם מידות יחסיות: 1em, 2em, 2.5em
בשיטה האחרונה ניתן להשתמש גם בחלקי מספר: 1.75em.

גודל

לכל אובייקט (לקופסה שהוא מייצג) ניתן לקבוע גודל. ללא קביעת הגודל, הדפדפן יבחר לו לבד משהו שעלול להשתנות בהתאם לנסיבות.

גודל הקופסה: רוחב – width, וגובה – height, נמדדים בפיקסלים, או באחוזים מהמסך:

```
<div style="width:500px;">
<div style="width:60%;">
```

המדידה באחוזים תגרום לבלוק לשנות את גודלו בהתאם לגודל החלון על המסך.

מיקום

ברירת המחדל היא מיקום הבלוק במקום הבא הפנוי....

אם נרצה מיקום מסוים ברחבי העמוד, נצטרך להגדיר בעיצוב לפחות 2 פרמטרים: את העובדה שהמיקום מוחלט, ואת המרחק מלמעלה ומאחד הצדדים:

```
#block1{
	position:absolute;
	top:100px;
	left:100px;
	width:500px;
}
```

מיקום יחסי

אם נרצה למקם 3 בלוקים זה לצד זה כך שיחס הרוחב ביניהם ישאר קבוע, נוכל לבצע זאת באופן הבא:

```
<div id="top">
	<div id="a">
	</div>
	<div id="b">
	</div>
	<div id="c">
```

```
        </div>
</div>
```

בקובץ הסגנונות נרשום:

```
#top{
        display:flex;
        width:90%;
}
#a{
        flex:1;
}
#b{
        flex:2;
}
#c{
        flex:3;
}
```

המשמעות: רוחבו של הבלוק top הוא 90% מהחלון. הוא מכיל תת בלוקים במדיניות הצגה גמישה.
יחסי הרוחב בין הבלוקים a, b, c הוא: 1, 2, 3. כלומר אם ניקח את רוחבו של top ונחלק ל-6, אזי רוחבו של a יהיה 1/6, רוחבו של b יהיה 2/6 ושל c 3/6.

דף רישום משתמש חדש

אחת ממשימות הפרויקט היא לקלוט פרטי משתמשים שיירשמו לאתר שלנו.

דף רישום פרטי משתמש הוא דף שזו בדיוק מטרתו.

עד כה ראינו אמצעים שונים להצגת טקסטים ותמונות, וכאן אנו נדרשים לקלוט נתונים.

קליטת נתונים תיעשה בדרך כלל על ידי תגית <input>
התגית יוצרת לנו איזור בו ניתן יהיה להקליד נתונים (טקסט, מספרים וכיו"ב).
קיימים מספר סוגי קלטים, והשימושיים ביותר הם:

מיועדת לקליטת טקסט	<input type=”text”>
דומה לטקסט, אך תוכן הטקסט יוצא מוסווה	<input type=”password”>
מאפשר בחירה של ערך **אחד** מתוך כמה	<input type=”radio”>
מאפשר בחירה של מספר ערכים	<input type=”checkbox”>
מאפשר בחירת ערך אחד מתוך תפריט הנפתח עם לחיצה	<select> <option value="zero">אנא בחר יום</option> <option value="Sunday">ראשון</option> <option value="Monday">שני</option> </select>

תגית text/password
לכל תגיות ה-input מסוג טקסט או סיסמה יש להוסיף מזהה מסוג **id** וגם מזהה מסוג **name**. אפשר גם להוסיף ערך class. לדוגמה:
```
<input type=”text” id=”firstName” name=”firstName” class=”userData”>
```

שימו לב שה-id וה-name זהים בערכם. ה-id ישמש אותנו לגישה דרך ה-javascript. ה-name ישרת אותנו לגישה מהשרת.
אם מעוניינים לתת סגנון אחיד לכל התגיות, ניתן להוסיף class עם ערך זהה לכל התגיות.

תגית הרדיו
מכיוון שבתגית הרדיו רק ערך אחד נבחר:
```
<input type="radio" name="gender" value="male" />male
<input type="radio" name="gender" value="female" />female
```

הפעם אין לנו id. אין אפשרות לשני אובייקטים להיות עם אותו מזהה. תפקיד המשתנה name לשייך את כל תגיות הרדיו בעמוד לאותו המשתנה. כך שאם יש לנו מספר כפתורי רדיו, לדוגמה:
מין: זכר או נקבה
כיתה: י או יא או יב
ידע הדפדפן למי לשייך את התשובה.
המשתנה name, במקרה שלנו gender, יכיל את הערך של התשובה הנבחרת: male או female. ולא את הערך המוצג על המסך.

תגית רב ברירה – checkbox
תגית ה-select מאפשרת לבחור ערך אחד מתוך רשימה הנפתחת למטה drop down. בדומה לתגית הטקסט, יש צורך להצמיד לתגית זיהוי ושם משתנה:
```
<select id="myday" name=”myday”>
        <option value="zero">אנא בחר יום</option>
        <option value="Sunday">ראשון</option>
```

```
		<option value="Monday">שני</option>
		<option value="Tuesday">שלישי</option>
		<option value="Wednesday">רביעי</option>
		<option value="Thursday">חמישי</option>
		<option value="Friday">שישי</option>
		<option value="Shabat">שבת</option>
	</select>
```

יש עוד אפשרויות רבות, ניתן ללמוד עליהם ב- W3School.

מה נדרש בטופס הפרטים?

טופס פרטי משתמש חדש חייב לכלול לפחות את הפרטים הבאים:

1. שם
2. שם משפחה
3. אימייל
4. סיסמה
5. אימות סיסמה

בנוסף, כחלק מעולם התוכן של הפרוייקט שלכם יש להוסיף מידע נוסף. לדוגמה בלבד:

1. מין - כפתורי רדיו
2. גיל - תפריט גלילה, או טקסט, או נומרי
3. רשימת תחומי ענין / תחביבים - check box
4. כל מידע אחר שייחד את העבודה שלכם.

הטופס חייב לכלול שימוש בקלט מסוג:

טקסט

סיסמה

רדיו

checkbox

מומלץ מאד להשתמש בטבלה כמעטפת לטופס, וזאת על מנת שהטופס ייראה מיושר על נמסך.

בתחתית הטופס יש להוסיף שני כפתורים של שליחת הנתונים, ואיפוס המסך:

```
<input type=”submit” value=”send” />
<input type=”reset” value=”clear” />
```

את כל הטופס יש "לסגור" עם תגית טופס -<form>. פרטים נוספים בפרק על javascript

JAVASCRIPT לשירות בדיקת הטופס

javascript היא שפת תכנות דינמית מונחית־עצמים המותאמת לשילוב באתרי אינטרנט. היא רצה על ידי דפדפן האינטרנט בצד הלקוח. השפה מרחיבה את יכולות שפת התגיות הבסיסית HTML ומאפשרת בכך ליצור יישומי אינטרנט מתוחכמים יותר. למעשה, רוב אתרי האינטרנט המודרניים משלבים שפה זו. היא ידועה בעיקר כשפה המוטבעת בדפי HTML על מנת להציג דפים דינמיים, שמשולבת בהם תוכנה. קוד ה־JavaScript שמשולב בדף HTML מבוצע על ידי הדפדפן.

> בדומה ל-HTML, גם במקרה של javascript הדפדפן לא יודיע על שגיאות! וכאן הכוונה לשני סוגי השגיאות: תחביריות ושגיאות זמן ריצה. הקוד פשוט לא יבוצע....

את קוד ה-javascript נמקם בתוך תגית script, אותה ניתן למקם בכל מקום ברחבי המסמך. עדיפות שלי: בסוף המסמך, ממש לפני ש"סוגרים" את תגית ה-body. בנוסף אם יש לנו קוד משמעותי, ניתן לשמור אותו בקובץ נפרד עם סיומת js. את הקישור לקובץ נעשה באופן הבא:

```
<script src="myscripts.js"></script>
```

מספר טיפים חשובים לזכור:

1. קוד ה-javascript מאורגן ביחידות של פונקציות.
2. קוד שאיננו חלק מפונקציה, מבוצע עם טעינת הדף. סוג של איתחול.
3. משתנים גלובליים יוחזקו מחוץ לפונקציות.
4. משתנים ב-javascript הם חסרי טיפוס, ולכן כולם מסוג var. הטיפוס מתקבל עם ההצבה אליהם.
5. כל פונקציה יכולה להחזיר ערך. מכיון שאין סוגי משתנים, אין צורך להכריז על ההכרזה. ולכן כותרת הפונקציה תהיה: function functioName(param list)
   רשימת הפרמטרים, אפס או יותר, מופרדת בפסיקים וכמובן שאין צורך בהגדרת הטיפוס.
6. שגיאות תחביריות יגרמו לפונקציות המוגדרות בקטע ה-script לא להיות מוכרות
7. שגיאות זמן ריצה יגרמו לתופעות בלתי צפויות...
8. ההוראה debugger תגרום לעצירת הריצה breakpoint, אם הדפדפן פתוח במצב של בדיקה inspect.

לצורך הפרויקט נשתמש בסקריפט בעיקר לבדיקת נתונים, ולטיפול בכפתורי לחיצה.

מענה ללחצנים - **buttons**

אם נרשום בתגית הטופס את המאפיינים הבאים:

```
<form onsubmit="return check();"
    onreset="return confirm('Deleting all fields\n Continue?');"
    onclick="return clearAll();"
    method="post">
....
</form>
```

- לחיצה על כפתור מסוג submit יגרום לקריאה/זימון של פונקציה בשם check. הפונקציה מצופה להחזיר true או false. במקרה של true, יועברו הנתונים לצד השרת בשיטת post. אחרת, הטופס ישאר ללא שינוי בצד הלקוח.
- לחיצה על כפתור מסוג reset תקפיץ הודעה לאישור מחיקה. לחיצה על ok תחזיר true, וכל שדות הטופס ימחקו. אחרת, הטופס ישאר ללא שינוי בצד הלקוח.
- אם הפונקציה check כותבת הודעות שגיאה למסך, ונרצה "להעלים" את השגיאות לאחר לחיצה על submit עם תחילת הקלדת הנתונים, אזי הפונקציה המזומנת על ידי onclick תעזור לנו.

בדיקת נתונים

על מנת לגשת לנתונים בטופס, לצורכי בדיקה, נבדיל בין סוגי קלטים שונים: טקסט, תפריט יורד מטה (select), רדיו, רב-ברירה (checkbox). דוגמאות בסוף פרק זה.

טקסט - לכל תגית קלט מסוג טקסט מומלץ לתת שם וזיהוי זהים. הזיהוי נוח יותר ב-javascript, והשם נחוץ לצד השרת. לדוגמה:

```
<input type="text" id="firstname" name="firstname">
```

על מנת לגשת לערך שהוקלד נשתמש ב-id:

```
var name = document.getElementById("firstname").value;
```

שימו לב לא לשכוח את value...

עכשיו יש לנו את המחרוזת, עליה נוכל לבצע מגוון של בדיקות. למשל: אורך, תכולת תוים מסוימים ועוד...

תפריט יורד מטה (select) - למרות הויזואליות השונה, ההתנהגות זהה לטקסט. כלומר פניה לפי שם הזיהוי.

כפתורי רדיו - כאן המצב שונה. כפתור אחד יקבל ערך true וכל האחרים false. נניח שיש לנו 3 כפתורים, ונרצה לוודא שהמשתמש בחר ולא דילג. כפתורי הרדיו יראו כך:

```
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
<input type="radio" name="gender" value="other"> Other
```

נשים לב שבמקרה של רדיו, אין לנו מזהה, משום שלא ניתן להגדיר אותו המזהה למספר פריטים. כפתורי הרדיו יקובצו בהתאם לשם, ובמקרה שלנו - gender. אם יש בעמוד עוד כפתורי רדיו, ניתן להם שם אחר.

הקוד הבא יבדוק שהמשתמש בחר על אחד מהכפתורים:

```
var gender = document.getElementsByName("gender");
if(!gender[0].checked && !gender[1].checked && !gender[2].checked)
{
/* כאן נטפל בשגיאות */
}
```

דוגמאות

אם נרצה להוציא הודעת שגיאה על שם שלא הוקלד, או אימייל לא חוקי, נוסיף בטופס שלנו במקום בו נרצה להציג את השגיאה, קוד דומה לזה:

```
<td><input type="password" id="password" name="password"></td>
<td class="errorMessage" id="_password"></td>
```

שתי השורות הנ"ל מציגות שני תאים בטבלה. הראשונה מסוג קלט של סיסמה. השניה אינה מכילה דבר, אך מתוייגת בסגנון מסוג errorMessage. למשל פונטים אדומים.

אם נזהה שגיאה בפורמט הסיסמה, בזמן הקלט, נוכל להציג שגיאה באופן הבא:

```
document.getElementById("_password").innerHTML = "password too short";
```

בפונקציה clearAll, נעבור על כל שדות השגיאה ונדאג לרשום מחרוזת ריקה...

דוגמה לבדיקת אימייל

נרצה לוודא שהאימייל שהוקלד מכיל את התו '@', ולפחות נקודה '.' אחת אחריו. בידקו את הקוד הבא:

```
function checkEmail(emailId)
{
    var email = document.getElementById(emailId).value;
    var emailError = document.getElementById("_" + emailId).innerHTML;
    if(email.length == 0)
    {
        emailError = "please fill email";
        return false;
    }
    var firstAt = email.indexOf('@');
    var lastAt = email.lastIndexOf('@');
    if(firstAt != lastAt || firstAt == -1)
    {
        emailError = "bad email format";
        return false;
    }
    var dot = email.lastIndexOf('.');
    if(dot <= firstAt + 1)
    {
        emailError = "bad email format";
        return false;
    }
    return true;
}
```

בפונקציה check נבדוק את כל מה שדורש בדיקה לפני שליחה לשרת. נוכל להשתמש במשתנה עזר בוליאני שיאגור את תוצאות הבדיקה. לדוגמה:

```
function check ()
```

```
{
    var okay = true;
    okay = checkName ("firstname") && okay;
    okay = checkName ("lastname") && okay;
    okay = checkEmail ("email") && okay;
    …
    return okay;
}
```

מעבר מ- ++NOTEPAD לויז'ואל סטודיו

בשלב הזה כבר יש לנו מספר דפי html, קובץ סגנונות – css, וקטעי קוד ב-javascript המוכלים בתוך הדפים עצמם, או נמצאים בקבצים עצמאיים. זה הזמן להעביר את הפרויקט לסביבת ויז'ואל סטודיו, שהיא הסביבה בה יוגש הפרויקט הסופי.
לפני שנבצע את המעבר, חשוב להבין כיצד מתחלקת העבודה בין השרת ללקוח., וכן חשוב להבין את תפקיד דף המאסטר.
בפרק זה נסקור רק בקצרה את עקרונות העבודה עם שרת ועם דפי מאסטר. הפרק הבא מרחיב את הנושא.

עבודה עם שרת

באתר אינטרנט נמצא התוכן בשרת המיועד לטפל בפניות לקוחות לקבלת מידע, כלומר לשליפת דפים והצגתם בצד הלקוח. קיימים מספר סוגים של שרתי אינטרנט. בפרויקט זה נעסוק בשרת של מיקרוסופט המריץ קוד בסביבת סי-שארפ.
כאשר לקוח פונה לאתר אינטרנט באמצעות הדפדפן, הוא מבקש לקבל עמוד לתצוגה. שרת האינטרנט ישלח את העמוד ללקוח, ושם הוא יוצג על ידי הדפדפן.
קיימים שני סוגי דפים: html, ו-aspx.
עמודים בעלי סיומת html/htm לא עוברים כל עיבוד על ידי השרת, ונשלחים כפי שהם ללקוח.
עמודים בעלי סיומת aspx עוברים עיבוד בצד השרת, והתוצאה נשלחת ללקוח.
נשתדל שכל העמודים שלנו יהיו מסוג aspx, זה יאפשר לנו גמישות בהמשך להוסיף אלמנטים של עיצוב, עיבוד, הגנה ועוד.
הפרק הבא מסביר כיצד הדבר מתבצע.

עבודה עם דפי מאסטר - master page

העובדה שהדפים עוברים עיבוד ראשוני בצד השרת, מאפשרת לנו לבנות דפים עם מראה אחיד. לדוגמה:

בדף למעלה נמצא 4 איזורים:

1. כותרת הדף - header
2. תחתית הדף – footer
3. איזור התפריט – menu bar
4. גוף הדף – main
5. שימו לב שיזור ה-לוגו אינו ממוספר, הוא יופיע תמיד.

זוהי רק דוגמה, וכל פרויקט יכול להכיל מספר איזורים לבחירת התלמיד. שימו לב שבניהול נכון של האיזורים, באמצעות הסגנונות CSS, ניתן לגרום לאיזור הראשי להשתמש בפס גלילה כך שה-footer תמיד ישאר במקום.

אז איך עובד המאסטר?

נחליט על תבנית, למשל זו שבדוגמה ונבנה דף aspx מסוג מאסטר שיגדיר לנו את האיזורים שנרצה. זוהי רק התבנית!!! מותר לשים בתבנית כל דבר שנרצה שיופיע **<u>בכל הדפים</u>**. למשל "שלום אורח" המופיע בכותרת בצד שמאל. או איזור התפריט.

נחשוב על כל איזור כאילו הוא "מיכל" – container. נבנה דף שיש בו 4 מיכלים. לכל מיכל נדאג לעיצוב משלו.

כל עמוד שיתמש בדף המאסטר, ידאג לתוכן של המיכלים.

המיכל שלנו נקרא ContentPlaceHolder, ודרך השימוש בו תהיה כך:

```
<div id="footerContent">
    <asp:ContentPlaceHolder ID="Footer" runat="server">
    </asp:ContentPlaceHolder>
</div>
```

למיכל שלנו יש מזהה ייחודי, בדוגמה כאן: Footer. הבלוק שעוטף אותו הוא מסוג div וכל תפקידו לשמש כמזהה עבור הסגנון בקובץ הסגנונות.

שימו לב שכל מה שייכתב בתוך ה- ContentPlaceHolder, לא יוצג !!!!!

מכיוון שתוכל המיכל נשלט על ידי העמוד ולא על ידי דף המאסטר.

בדוגמה של העמוד עם ארבעת האיזורים, נקבל קוד בסגנון הבא:

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="SiteMaster.Master.cs" Inherits="MasterExample.MyMaster" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
    <link type="text/css" rel="stylesheet" href="styles.css" />
    <title>My Site</title>
</head>

<body>
    <img class="logo" src="images/logo-haklai.jpg" alt="haklai logo" />
```

על השורה הראשונה – בפרק הבא

קישור לקובץ הסגנונות

לוגו יופיע בכל עמוד

```
    <div id="titleContent">
        <asp:ContentPlaceHolder ID="Header" runat="server">
        </asp:ContentPlaceHolder>
    </div>

    <div id="menuContent">
        <nav class="sideNav">
            <a href="Home.aspx">דף הבית</a>
            <a href="Page1.aspx">דף ראשון</a>
            <a href="Form.aspx">דף רישום</a>
        </nav>
        <asp:ContentPlaceHolder ID="MenuContent" runat="server">
        </asp:ContentPlaceHolder>
    </div>

    <div id="mainContent">
        <asp:ContentPlaceHolder ID="MainContent" runat="server">
        </asp:ContentPlaceHolder>
    </div>
    <div id="footerContent">
        <asp:ContentPlaceHolder ID="Footer" runat="server">
        </asp:ContentPlaceHolder>
    </div>
</body>
</html>
```

מיכל הכותרת

חלק זה אינו בתוך מיכל, ולכן יוצג בכל עמוד. בנוסף – כל עמוד יכול להוסיף פריטים לתפריט

מיכל התפריט

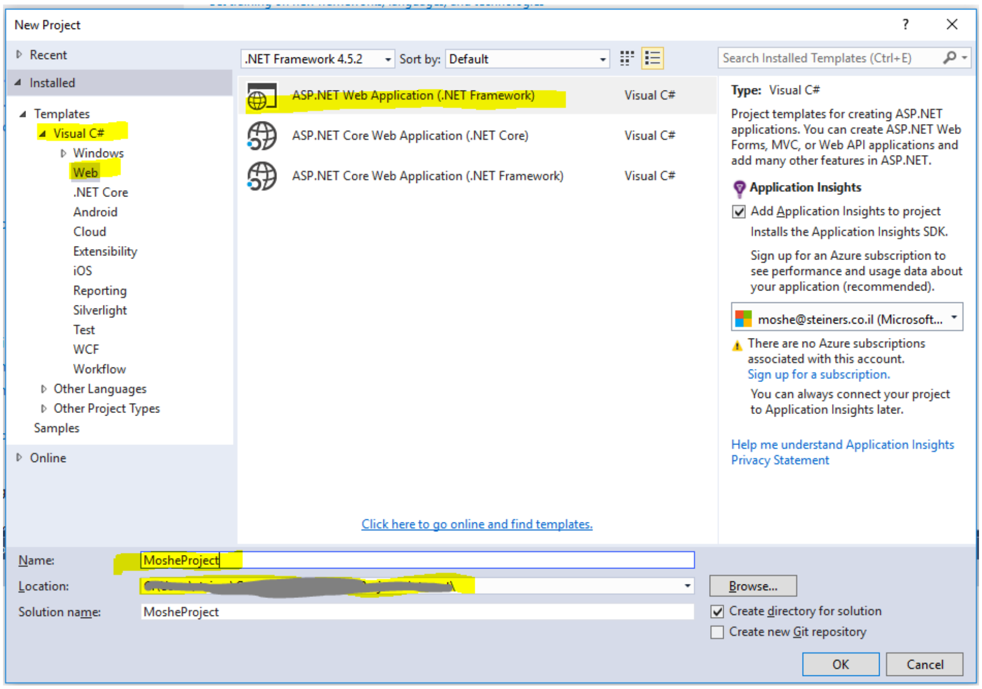
מיכל האיזור הראשי

מיכל תחתית העמוד

כלומר, דף המאסטר נראה כמו דף HTML ללא תכולה אמיתית. רק עם הגדרות מיכלים. לכל מיכל יש מזהה id, הניתן להתייחסות בקובץ הסגנונות: גודל, מיקום, צבעים וכו'.

עוברים לויז'ואל סטודיו

בויז'ואל סטודיו נבנה פרויקט חדש, ונבחר:

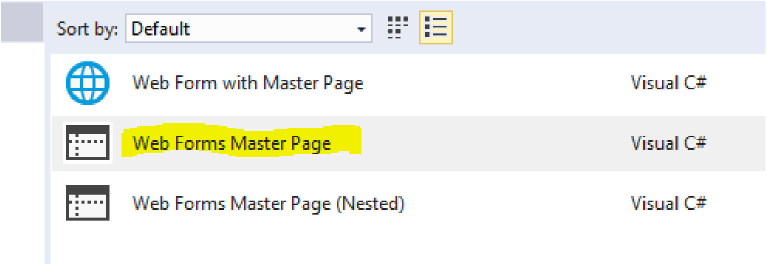


1. ויז'ואל C#
2. Web
3. ASP.NET Web application
4. שם פרויקט שיכלול את שם התלמיד !!
5. היכן יאוחסן הפרויקט במחשב
6. נאשר – okay

במסך הבא נבחר פרויקט ריק
לעיתים יופיע עוד מסך, ניתן ללחוץ על ביטול – cancel.

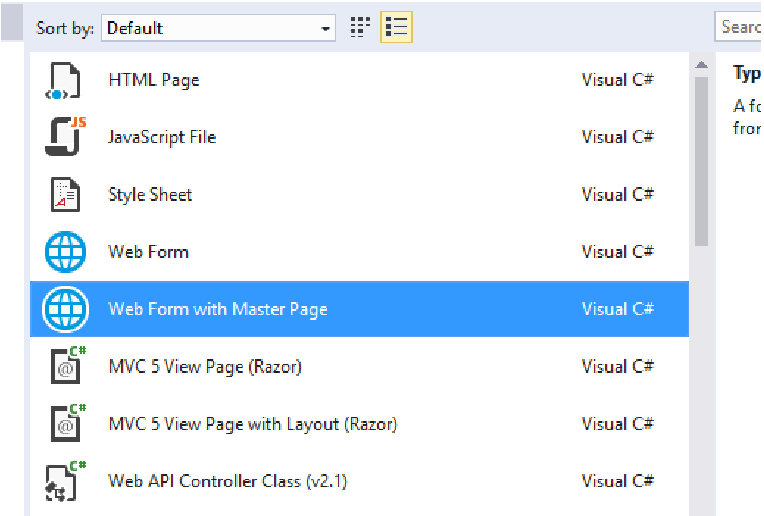
כעת הוקמה לנו סביבת העבודה.
נייבא את קובץ הסגנונות שבנינו עד כה. אם הקובץ היה במחיצה שונה מ"שורש" הפרויקט, אז כדאי לבנות מחיצה זו גם בפרוייקט על ידי לחיצה ימנית על שם הפרויקט (ב-solution explorer), הוסף -> מחיצה חדשה. את קובץ הסגנונות נייבא על ידי הוסף->קובץ קיים. סטודיו יעתיק את הקובץ לאיזור הפרויקט.

בנית עמוד המאסטר - נוסיף פריט חדש ונבחר web form master page:



לא לשכוח לתת לו שם משמעותי, אחרת שמו יהיה site1.
את עמוד המאסטר נבנה בהתאם להנחיות למעלה.
שימו לב: שבונים דף מאסטר התבנית הבסיסית שלו כוללת הגדרת טופס <form runat=”server”>. מומלץ למחוק את השורה הזו ואת השורה הסוגרת את התגית.

ייבוא העמודים שבנינו באמצעות ++notepad:
נוסיף פריט חדש ונבחר web form with master page



וניתן לו את השם של העמוד אותו אנחנו רוצים לייבא. לדוגמה, אם נרצה לייבא את העמוד clubs.html אז לעמוד החדש נקרא clubc.aspx.

לאחר שנאשר, יופיע חלון לבחירת דף מאסטר (מותר לנו להחזיק מספר דפי מאסטר שונים, ולכן יש לבחור במי להשתמש). נקבל את הדף הבא:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/SiteMaster.Master"
AutoEventWireup="true" CodeBehind="clubs.aspx.cs"
Inherits="MasterExample.WebForm1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="Header" runat="server">
          כאן נכניס את תוכן הכותרת
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MenuContent" runat="server">
          כאן נכניס תפריטים ספציפיים לעמוד
</asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="MainContent" runat="server">
          כאן נכניס את תוכן העמוד
</asp:Content>
<asp:Content ID="Content4" ContentPlaceHolderID="Footer" runat="server">
          כאן נכניס את תוכן תחתית הדף
</asp:Content>
```

דף זה **אינו** דומה לדף HTML שהיכרנו. אחרי השורה הראשונה מופיעים שמות המיכלים שהגדרנו במאסטר.

כעת נעתיק באופן מבוקר וזהיר את התוכן מהעמוד המקורי שלנו אל תוך המיכלים.

שימו לב שבקובץ זה לא תופענה התגיות של html, head, body כי הכל נמצא במאסטר.

גם ההפניה לקובץ הסגנונות נמצאת במאסטר.....

לא חובה למלא את כל המיכלים!

לפני שמתחילים למלא את המיכלים, רצוי להריץ את העמוד, ולקבל את התחושה של התנהגות המאסטר בלבד.

העבודה בצד השרת - CodeBehind

הרעיון המרכזי בעבודה עם שרת אינטרנט, היא העבודה שהדפים מאוחסנים בצד השרת. כאשר לקוח פונה לאתר אינטרנט, הוא בעצם יוצר בקשה לקבל עמוד לתצוגה. שרת האינטרנט ישלח את העמוד ללקוח, ושם הוא יוצג על ידי הדפדפן. לפני שליחת הדפים ללקוח השרת יכול לעדכן את הדפים. עמודים בעלי סיומת html/htm לא עוברים כל עיבוד על ידי השרת. עמודים שהסיומת שלהם aspx מיועדים לעיבוד על ידי השרת. לכל עמוד כזה קיים קובץ בשם הזהה לשם העמוד עם סיומת cs. לדוגמה:
בדף ה-aspx, בשורה הראשונה רשומות הנחיות שונות לשרת, כמו למשל שם הקובץ של הקוד עבור השרת - codeBehind, שם המאסטר – MasterPageFile, ועוד. מומלץ בחום **לא** לשנות דבר בשורה זו....
עבור העמוד homePage.aspx נוכל למצוא קובץ homePage.aspx.cs. לקובץ זה נהוג לקרוא code behind, והוא כתוב ב-סי שארפ, ומבוצע כולו על ידי השרת. ללקוח אין גישה אליו!
בנוסף, ניתן לכתוב קוד בסי שארפ עבור השרת בעמוד עצמו בתוך תגית מיוחדת:
<% קוד בסי שארפ המיועד לשרת %> .

סדר הביצוע הוא:

1. ביצוע הקוד בקובץ ה-cs
2. השרת עובר על העמוד, עבור כל שורה עם התגית המיועדת אליו היא תבוצע, ותוצאתה תחליף את התגית. למשל: אם בקובץ נגדיר משתנה: **rotected string name;**
ובמהלך עבודת השרת המשתנה יקבל את הערך "moshe". ובעמוד עצמו נמצא את התגית הבאה:
<% Response.Write(“hello “ + name);%>
אזי השרת יחליף את התגית, ובעמוד שנשלח ללקוח תתקבל השורה:
hello moshe
3. כעת עבר כל הקובץ לביצוע בצד הלקוח, והשרת סיים את תפקידו עבור עמוד זה.

העמודים המיועדים לעבודה עם השרת נקראים בשם הכללי טופס - form (גם אם אינם מכילים טופס), משום שבדרך כלל נשתמש בטופס על מנת להעביר נתונים מהלקוח לשרת.
הפעולה הראשונה המתבצעת בקוד על ידי השרת נקראת PageLoad. בפעולה זו נרצה להבחין האם זו הפעם הראשונה בו הגענו לקוד, כלומר רק טוענים את הדף, או שהגענו לאחר שהמשתמש לחץ על כפתור שליחה (submit) והגיע לשרת עם נתונים.

לסיכום :סדר הפעולות בטעינת דף מסוג aspx הוא:

1. השרת מחפש את העמוד המבוקש.
2. משורת הכותרת הוא "לומד" איך נקרא קובץ השרת, ואם קיים דף מאסטר.
3. השרת טוען ומבצע את הקוד בקובץ השרת
4. השרת טוען ומבצע את הקוד בקובץ המאסטר (על כך בהמשך)
5. השרת בונה את העמוד על ידי "איחוד" העמוד עם עמוד המאסטר
6. השרת עובר על העמוד שנוצר ומבצע את השורות המיועדות עבורו - <% %>
7. השרת שולח את התוצר המוגמר ללקוח.
8. יש דפים, כמו דף המנהל, שיתכן ותהיה תקשורת "פינג-פונג" בין הלקוח לשרת. כלומר השרת ישלח את

הדף הראשון, הלקוח יבצע עדכונים, השרת יבצע וישלח עמוד מעודכן, וכך הלאה.

אובייקטים לשימוש השרת (Request, Response, Session, Application)

בכל פעם שהשרת מבצע את הקוד הוא מקבל הפניות למספר אובייקטים:

אובייקט ה-Request

באובייקט זה נוכל לקבל את כל הנתונים המגיעים מצד הלקוח.
לדוגמה, אם בצד הלקוח קיים טופס, ובו שדה קלט בשם firstName, ושיטת העברת הנתונים היא post:

```
<input id="firstName" name="firstName" type="text">
```

אזי בצד השרת נוכל להגיע לערך אותו הכניס הלקוח.

```
if(Request.Form["firstName"] != null)
{
        name = Request.Form["firstName"].ToString();
}
```

אם נתונים מגיעים בשיטת get, כלומר העברת נתונים מבוצעת בשורת הכתובת למשל:
update.aspx?firstName=moshe&email=moshe@gmail.com

אזי השרת יוכל לקבל את הנתונים באופן הבא:

```
if(Request.QueryString["firstName"] != null)
{
        name = Request.QueryString["firstName"].ToString();
}
```

חובה לבדוק שהאובייקט קיים. אם קיים, יש להשתמש ב-ToString להמרה למחרוזת

אובייקט ה-Response

באובייקט זה נוכל להעביר נתונים ללקוח. שימושים עיקריים:
הצגת ערך בעמוד: Response.Write המקבלת מחרוזת כפרמטר, ומציגה אותה.
מעבר לעמוד אחר: Response.Redirect("nextPage.aspx")

אובייקט ה-Session

לאחר ביצוע הקוד מאחורי עמוד כלשהו, השרת "עוזב" את מרחב הקוד. כלומר אין יכולת להגיע למשתנים ולתוצאות שחושבו עד שלב זה. כלומר כל עמוד מבוצע כאילו הוא העמוד הראשון. ביצוע כזה נקרא חסר זיכרון

- stateless. ומטרתו לאפשר טיפול של שרת אחד בכמות גדולה של משתמשים.
על מנת שנוכל בכל זאת "לזכור" מספר פרטים בין העמודים של האתר, קיים אובייקט מיוחד שמלווה את התקשורת של הלקוח עם השרת. זה ה-Session. אובייקט זה מסוגל להכיל ערכים כרצוננו אותם נוכל "לראות" מכל העמודים. אם למשל נגדיר:

```
Session[“name”] = ”moshe”;
```

אז כל העמודים יוכלו לקבל את הערך הנכון.
ה-Session מתקיים עבור כל העמודים של אותו המשתמש.

אובייקט ה-Application

בדומה ל-Session, קיים אובייקט המאפשר תקשורת בין כל המשתמשים באותו האתר. זהו ה-Application.
השימוש בו דומה ל-Session. הוא משמש בעיקר למטרות כמו ספירת כמות המשתמשים ברגע נתון באתר.
מדידת התעניינות במוצר מסוים על ידי יותר ממשתמש אחד וכיו"ב.

תחילת הביצוע בצד השרת

אם הדף שנטען הינו דף אינופרמטיבי בלבד, כלומר – לא עוברים נתונים חזרה לשרת, הרי שהשרת יבצע את הקוד שלו פעם בלבד לפני שליחת העמוד ללקוח, ובכל פעם שתתבצע פעולת "ריענון" בצד הלקוח.
אך אם העמוד הוא למשל טופס רישום, אזי נגיע לשרת לפחות פעמיים: לפני טעינת הדף, ואחרי מילוי הפרטים.
לכן עלינו לברר באיזה מצב אנחנו נמצאים.... יש מספר שיטות לבצע זאת, נזכיר שתיים:

1. **מילוי אחד השדות**

   אם הדף המדובר הוא למשל טופס רישום, אזי נרצה לקבל את כל הפרטים לשרת. נוכל לבחור שדה כרצוננו, למשל email ולברר האם הוכנסו בו נתונים.... אם לא הוכנסו בו נתונים, אז כנראה שזו הפעם הראשונה לפני שהדף נשלח ללקוח, או שממילא נפלה טעות (חסר אימייל) והפרטים אינם מעניינים.
   את הבירור נבצע באופן הבא:

   ```
   If (Request.Form[“email”] != null && Request.Form[“email”].ToString() != “”)
   ```

   שימו לב: חשוב מאד שבטופס יוגדר שם השדה email ולא רק המזהה:

   ```
   <input type=”text” id=”email” name=”email”>
   ```

2. **שיטת המשלוח**

   דרך נוספת היא לברר באיזה אופן נשלחו אלי הנתונים. בפעם הראשונה שהדף נטען עדיין לא נשלחו נתונים, אז מוגדרת שיטת משלוח מסוג get. מתוך הנחה שאנו שולחים את הנתונים בשיטת post, נוכל לבדוק:

   ```
   If (Request.HttpMethod == “post”)
   ```

לאחר שבררנו שהנתונים זמינים, נזמן את הפעולה המבצעת את מטרת הדף.

> למרות שחלק/כל הנתונים נבדקו בצד לקוח, לא ניקח את הסיכון שמשהו השתבש.
>
> בצד לקוח הדפדפן מאד סובלני, ואם נפנה לאובייקט לא קיים (null) לא נרגיש בכך.
> פניה ל-null בצד השרת תגרום לקריסת הקוד.

לכן, לפני כל פניה לנתון, נברר אם הוא זמין:

```
If (Request.Form["email"] != null)
{
	email = Request.Form["email"].ToString();
}
else
{
	message = "missing email";
}
```

מסד הנתונים - DATABASE

מבוסס על המסמך של אביטל גרינולד על מסד נתונים.

אוסף נתונים גדול המאורגן בצורה כזו שניתן להגיע לכל נתון בצורה מהירה וקלה.
אחד האמצעים לשמירת מידע רב לאורך זמן הוא מערכת ממוחשבת לניהול מסד נתונים – Data Base Management System – DBMS.
מאוד נוח למקם ברשת האינטרנט באחד המחשבים בסיס נתונים ולאפשר לגולשים מכל מקום להתחבר אליו.
ומדוע נרצה להשתמש במסד נתונים? זו הדרך היעילה ביותר לשמור על כמויות גדולות מאד של מידע המגיע מהרבה מקורות ומשרת הרבה לקוחות.
ישנם מודלים אחדים לייצוג מסדי נתונים. הנפוץ ביותר הוא המודל הטבלאי.
במודל זה הנתונים מאורגנים בטבלה. כל שורה בטבלה מייצגת קבוצה של יישויות. שורה בטבלה נקראת רשומה - record. כל עמודה בטבלה נקראת שדה – field. בתוך תאי הטבלה נמצאים הנתונים.

דוגמה לטבלה המייצגת קבוצת ישויות – תלמידים

מספר זהות	שם	שכבה	כיתה	רחוב	עיר
1544	יוסי כהן	יב	1	נורדאו 5	תל-אביב
1614	זהבה לוי	יב	1	הרצל 14	רמת-גן
1620	חיים מצליח	יא	2	ויצמן 94	תל-אביב
1637	זהר אוריין	י	1	ז'בוטינסקי 9	תל-אביב
1712	חיים ביטון	י	2	חרל"פ 19	גבעתיים
1714	יונה קדוש	ט	1	זנד 9	תל-אביב

בטבלה זו יש 6 שדות ו 6 רשומות.
(*) לקוח מספר מבוא לתכנות אינטרנט בהוצאת מטה

שפת שאילתות - שפת SQL

על מנת לפנות למסד הנתונים ולבצע עליו פעולות, נשתמש בשפת שאילתות SQL – Structured Query Language. המיוחד בשפה זו שהיא מאפשרת ניהול מסד הנתונים בשרת מרוחק, וגישה אליו ממחשבים מרוחקים (לקוחות) שונים.
השפה מאפשרת בנית מסד נתונים, איחסון, שליפת מידע וסינון על פי קריטריונים.
שפת ה-SQL אינה רגישה לאותיות גדולות או קטנות.

במסד נתונים מלא נמצא בדרך כלל יותר מטבלה אחת, כאשר בין הטבלאות קיימים יחסי גומלין. בפרויקט שלנו נסתפק בטבלה אחת.
נהוג שלכל שורה בטבלה, לכל ישות, יש לפחות תכונה אחת ייחודית המבדילה את הישות/רשומה מישויות/רשומות אחרות. למשל תעודת זהות. בפרויקט שלנו זה יכול להיות אימייל. לתכונה כזו נקרא "מפתח".

ניתן להחזיק יותק ממפתח אחד לטבלה, ואז נגדיר את אחד מהם כמפתח ראשי.
הרעיון בהגדרת מפתח היא מניעת כפליויות. זה בסדר שיהיו שני תלמידים בעלי אותו שם פרטי ואפילו אותו שם משפחה, אבל תעודת הזהות שלהם חייבת להיות שונה.
ה-DBMS לא יאפשר לנו להכניס שורה חדשה עם מפתח קיים.

קיימים סוגים שונים של שאילתות, אנחנו נתמקד בארבעה סוגים. לצורך ההדגמה, ניקח את הטבלה שהוצגה קודם לכן, ונניח ש:

א. שם הטבלה: Students
ב. שם התכונות/עמודות: id, name, classLevel, classNumber, street, city
ג. תעודת הזהות של תלמיד משמשת כמפתח

סוגי השאילתות

1. **שליפת נתונים – SELECT**

הפורמט הבסיסי של השאילתה הוא:

```
Select * from Students
```

מילת המפתח select מגדירה שאילתת שליפת נתונים.
הסימון כוכבית, *, מבקש מה-DBMS לשלוף את כל השדות של הטבלה.
אם נרצה למשל רק שם ומספר זהות, נרשום:

```
Select name,id from Students
```

המילה from מציינת מאיזה טבלה נרצה לבצע את השליפה. במקרה שלנו: Students.

ניתן למיין את התוצאות לפי קריטריון:

```
Select * from Students order by name ASC
```

המילים order by מורות על מיון. אחריהן יופיעו שמות התכונות לפיהן נרצה למיין, ולבסוף נבקש האם יהיה זה סדר עולה – ASC או סדר יורד - DESC

2. **הכנסת רשומה חדשה – Insert**

הכנסת נתוני רשומה חדשה תתבצע באופן הבא:

```
Insert into Students (id, name, street, city, classNumber)
      values (‘12345’, ‘moshe’, ‘Haatzmaut’, ‘Zichron Yaacov’, 3)
```

שאילתת ההכנסה מגדירה לאיזו טבלה נרצה להכניס, לאחר מכן, בתוך סוגריים, מציינת את התכונות/העמודות אותם נרצה להכניס. אין חובה להכניס את כל העמודות. **אבל !!!** יש חובה להכניס את כל העמודות שהוגדרו כחובה. למשל המפתח...
אחרי פירוט התכונות תופיע המילה values ולאחריה כל הערכים של השדות באותו הסדר שפירטנו בחלק הראשון של השאילתה. בדוגמה: מספר זהות, ואחריו שם, ואחריו רחוב, וכך הלאה.

מחרוזות יוכנסו בין גרשיים (גרש בודד). מספרים יוכנסו ללא גרשיים.

3. **עדכון רשומה– Update**

אם נרצה לעדכן תכונה מסויימת, למשל אם התלמיד עבר כיתה. נשתמש בשאילתת עדכון:

```
Update Students set classNumber=4, street='Arlozorov'
```

הסינתקס של שאילתת עדכון שונה מהכנסה! הדוגמה מדברת בעד עצמה.

> איזו רשומה מתעדכנת בדוגמה למעלה?

השאילתה כפי שנכתבה למעלה, תעדכן את כל הרשומות!!! וזאת מכיוון שלא אמרנו לה איזו רשומה לעדכן.
הדרך לצמצם את הבחירה היא על ידי הוספת תנאי:

```
Update Students set classNumber=4 where id='12345'
```

ניתן להוסיף איזה תנאי שנרצה, אולם התייחסות למספר הזהות מבטיחה איתור מדויק: קיימת לכל היותר רשומה אחת כזו, כי מספר הזהות הוא מפתח.

4. **מחיקת רשומה– Delete**

מחיקת רשומה מתבצעת על ידי שאילתת מחיקה. וגם כאן!!! חובה להוסיף קריטריון מיון, אחרת נמחוק את כל הטבלה:

```
Delete from Students where id='12345'
```

> אחד האתרים השימושיים ביותר ללימוד, בירור פרטים, ניסויי הרצה הוא האתר w3school

בנית מסד הנתונים בסביבת ויז'ואל סטודיו

באופן עקרוני ניתן ליצור מסד נתונים וטבלאות על הוראות SQL. אולם בפרויקט שלנו, נייצר אותם "ידנית" באופן חד פעמי . סביר להניח שבהמשך נרצה להוסיף עמודות...
נרצה לבנות טבלת משתמשים אשר תכיל עמודות/תכונות בהתאם לפרטים שנאסף בטופס הרישום, עם 3 שינויים קלים:

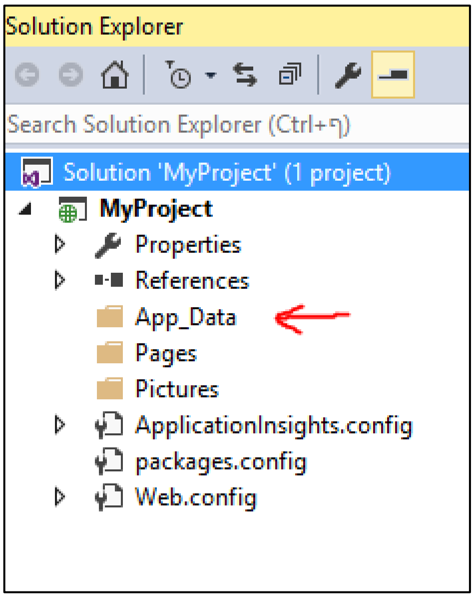
1. סיסמה נחזיק רק פעם אחת (אין צורך באימות סיסמה)
2. נוסיף עמודת מנהל - admin, המציינת האם המשתמש הינו בעל זכויות ניהול
3. עמודות המתארות checkbox יוחזקו כעמודה אחת.

הוספת מסד הנתונים

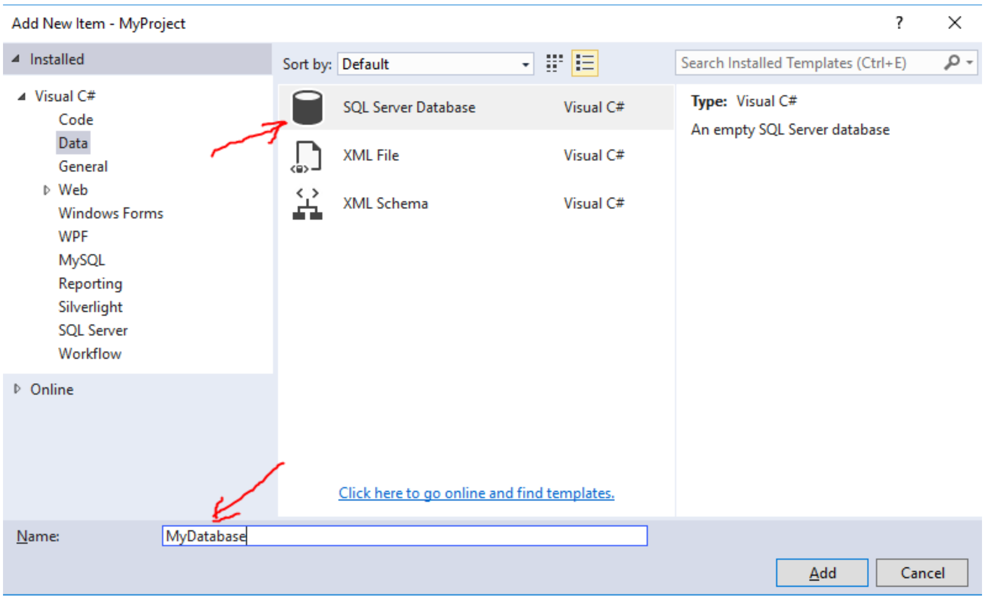
אחד היתרונות של השימוש בויז'ואל סטודיו הוא היכולת שלו למלא תפקיד של שרת אינטרנט **וגם** שרת SQL. איננו צריכים להקים סביבה נוספת.

שלב ראשון: הקמת מסד הנתונים

1. נפתח את הפרויקט שלנו בויז'ואל סטודיו.
2. בדרך כלל יופיע ה-solution explorer בצד ימין של המסך. אם לא, נפתח אותו דרך View->Solution Explorer
3. נלחץ עם הלחצן הימני של העכבר על שם הפרויקט המודגש (בדרך כלל יופיע בשורה השניה), ונבחר הוסף או Add מחיצה חדשה, או New Folder
4. תפתח לנו מחיצה, ותינתן לנו הזדמנות לשנות את השם. נשנה ל- App_Data. כך זה צריך להיראות:

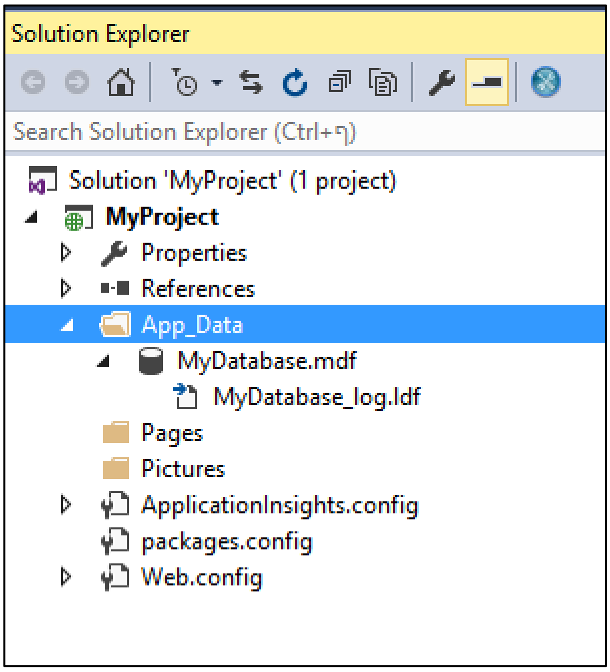


5. כעת נחזור על תהליך ההוספה כשנלחץ על App_Data. נבחר את Add New Item, ונקבל את המסך הבא, או דומה לו:



נבחר SQL Server Database, ולא נשכח לשנות את שם מסד הנתונים בתחתית הטופס.

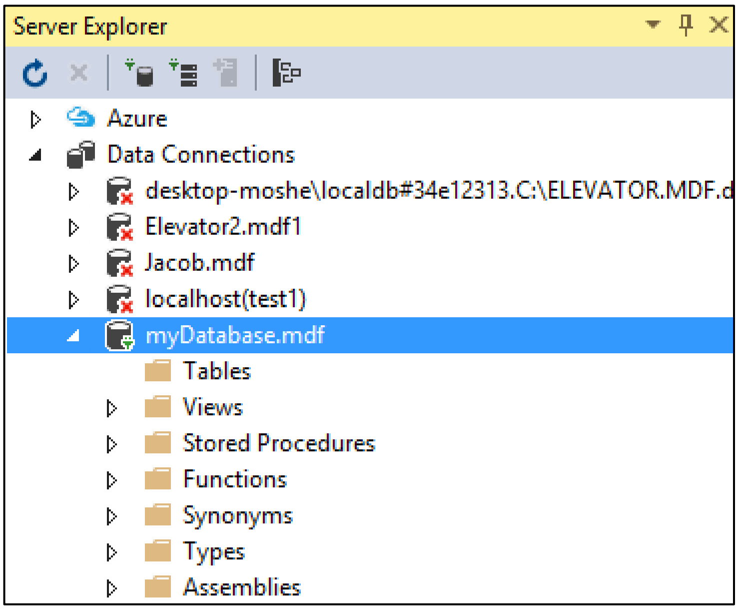
6. לאחר אישור, ה-solution explorer נראה כך:



מסד הנתונים התווסף למחיצה App_Data, ושם הקובץ מסתיים ב-mdf. הקובץ השני הוא לא לשימוש שלנו.

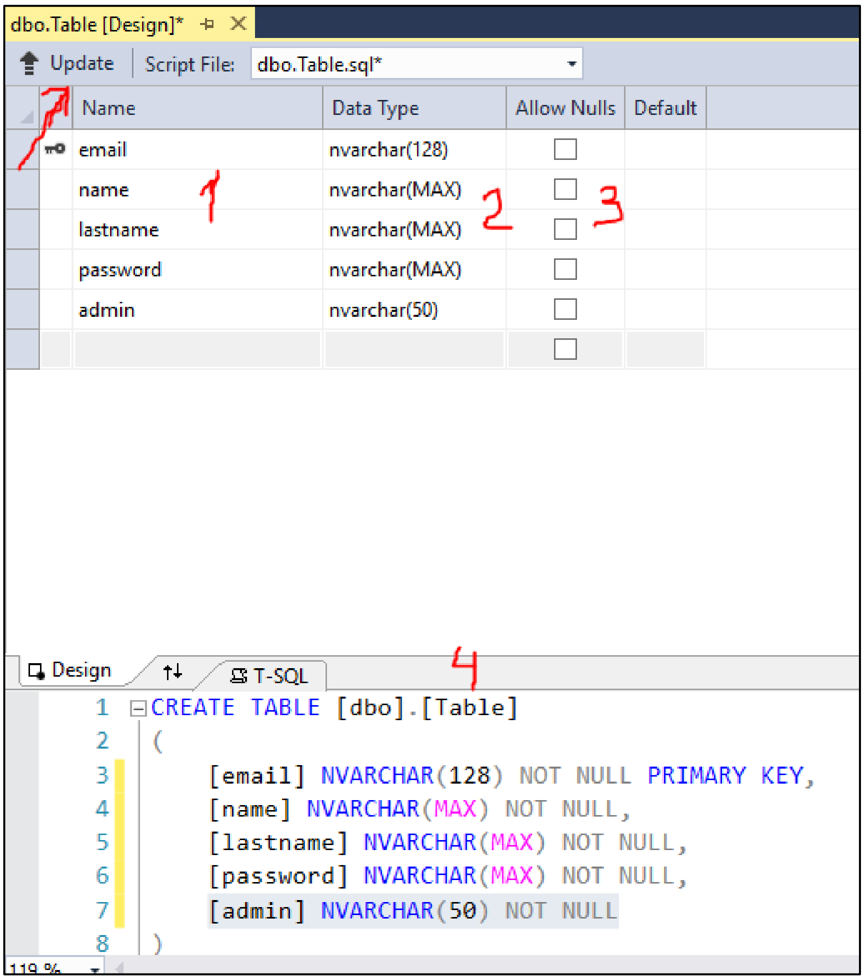
שלב שני: בנית טבלה

1. לחיצה/הקלקה כפולה על מסד הנתונים, ובצד שמאל של המסך יפתח חלון של server explorer.



אם מופיע X אדום, משמעותו שסטודיו לא "התחבר" למסד הנתונים. על מנת לחבר אותו, נלחץ על מסד הנתונים ונבחר – רענן – refresh.

2. מתחת למסד הנתונים שלנו נוכל למצוא תתי קטגוריות שונות. אותנו מעניינת קטגורית הטבלאות. כרגע היא ריקה. לחיצה על טבלאות עם עכבר ימני, ונבחר הוסף טבלה חדשה.



מסך בנית הטבלה מאפשר לנו להגדיר איזה עמודות, ומאיזה סוג נרצה בטבלה שלנו. נעבור כעת שלב אחר שלב בבנית הטבלה עצמה, לפי סדר המספרים המופיעים באדום.

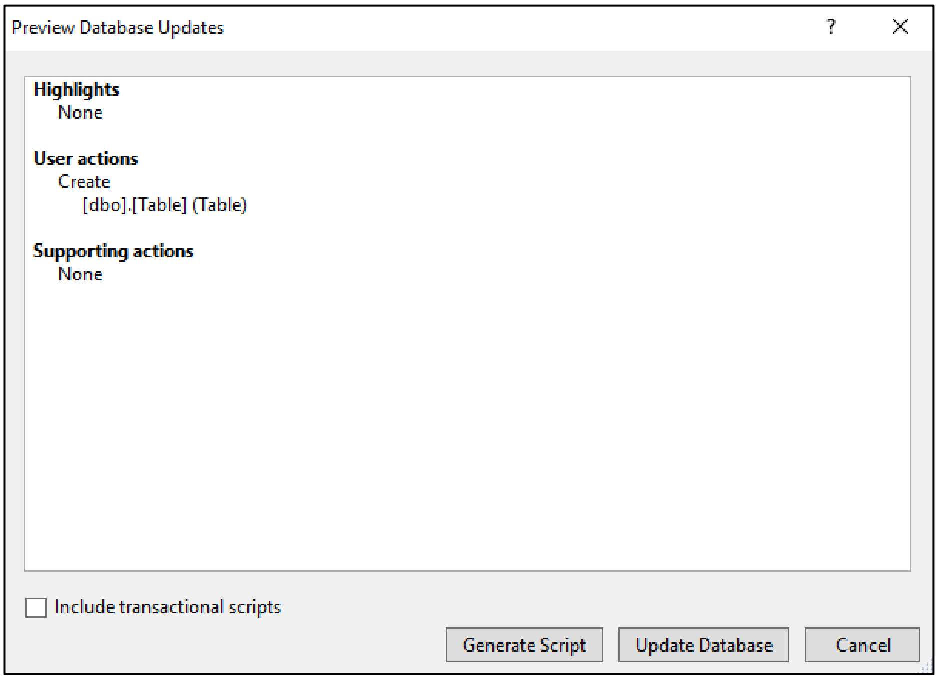
3. אדום 1: כשקיבלנו את מסך בנית הטבלה, הופיעה בו שורה אחת בלבד – id, ולידה ציור של מפתח. נשנה את השם id לשם email. עכשיו אימייל הוא שדה המפתח שלנו.

4. אדום 2: עבור כל משתנה/שדה מסוג טקסט נבחר את האפשרות של nvarchar. המספר המופיע בסוגריים הוא הכמות המקסימלית של תוים מותרים. לרוב נעדיף לבחור max. במקרה של שדה מפתח חובה להגביל את האורך. ובדוגמה השתמשתי ב-128. שדה ה-admin יכיל בדרך כלל yes/no, ולכן 50 זה יותר ממספיק.

5. אדום 3: נדאג למחוק את הסימון ב-Allow Nulls, כלומר אנחנו דורשים מה-DBMS שיקפיד השדות יקבלו ערך, ולא ישארו ריקים.

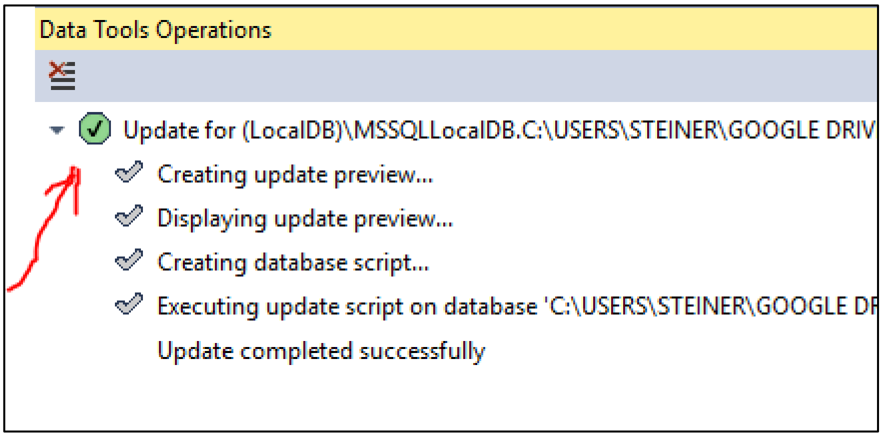
6. אדום 4: מסך הבניה בוחר לטבלה שם Table. נרצה להחליף אותו למשל ל-Users. בחלק התחתון של המסך, נחליף את המחרוזת [dbo].[Table] במחרוזת [dbo].[Users].

7. אדום 5: הטבלה שלנו עדין לא נוצרה, רק ההגדרות שלה מופיעות על המסך. לחיצה על כפתור Update יגרום להופעת מסך עידכון:



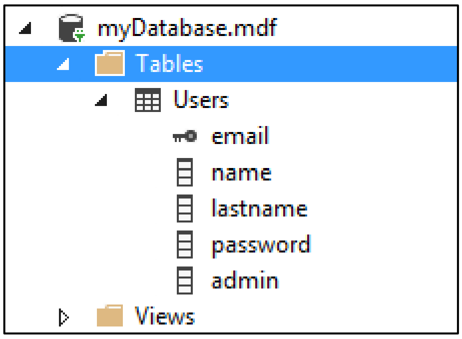
שימו לב שבמרכז החלון לא מופיעות שגיאות, כפי שנראה בדוגמה. אם המסך מכיל הודעות נוספות, כנראה שקיימת בעיה במסד הנתונים.

כעת נלחץ על הכפתור האמצעי למטה: Update Databse, ונמתין לסיום מוצלח:



עם סיום מוצלח יופיע עיגול בצבע ירוק בתחתי המסך, אחרת העיגול יהיה אדום.

8. אם נרענן את מסד הנתונים שלנו בצד שמאל של המסך, נקבל את התמונה הבאה:



הקטגוריה של טבלאות מכילה טבלה בשם Users ובה נמצאות התכונות/העמודות שהגדרנו.

9. לחיצה ימנית על שם הטבלה תחשוף לנו תפריט עם מספר אפשרויות, 3 מתוכן רלוונטיות לנו:
   א. New Query – תפתח חלון המאפשר לנו להריץ שאילתות SQL ישירות על הטבלה.
   הרצת השאילתה מתבצעת על ידי לחיצה על המשולש הירוק בכותרת מסך השאילתה בצד שמאל. התשובה לשאילתה מתקבלת בתחתית המסך.
   שימו לב: שאילתות הנכתבות בחלון זה אינן חלק מהפרויקט, ולכן לא נשמרות איתו. בכל פעם שנצא/נסגור את סטודיו, נקבל הודעה השואלת אותנו איך/איפה לשמור את השאילתה.
   אין חובה לשמור!

   ב. Open Table Definition – נקבל את חלון ההגדרות בדומה לתהליך הבניה

   ג. Show Table Data – תפתח לנו חלון הדומה לאקסל, ובו כל הנתונים הנמצאים ברגע זה בטבלה. בחלון זה ניתן לשנות ערכים, להוסיף או למחוק רשומות שלמות. בכל פעם שנכניס שורה חדשה מליאה, היא נשמרת מיידית במסד הנתונים. אין כפתור/תהליך שמירה נפרד.

   חלונות אלו אינם נשמרים כחלק מהפרויקט, ובכל פעם יש לפתוח אותם מחדש.
   מסד הנתונים – כן נשמר!

תהליך בנית מסד הנתונים הינו חד פעמי!
תהליך בנית הטבלה הוא גם חד פעמי באופן עקרוני, אך יתכן שנרצה להוסיף עמודות, לשנות את שמם או את הגדרתן במהלך פיתוח הפרויקט.

משימה

לאחר שבנינו טבלה, הכניסו לפחות 7 רשומות/שורות בטבלה באופן ידני (בחלון המתואר בסעיף ג' למעלה).
פיתחו חלון שאילתה, ובנו והריצו את השאילתות הבאות:
1. הצג את כל המשתמשים בטבלה
2. הצג פרטי משתמש שהאימייל שלו: user@gmail.com
3. הוסף משתמש חדש (בעזרת Insert), שימו לב מה מתקבל במסך התוצאות.
4. עדכן/שנה את הסיסמה של אחד המשתמשים
5. מחק את אחד המשתמשים

מסך השאילתות ישמש אותנו כ"פוסק" בעניין חוקיות השאילתות. כאשר נריץ את השאילתות מתוך הקוד, יהיו מצבים בהם נקבל שגיאות, ולא נבין למה. נוכל להעתיק את השאילתה שבנינו בקוד, ולהריץ בחלון השאילתות, ובצורה אינטראקטיבית לתקן את השאילתה. את התיקון ניישם בקוד.

בנית שאילתה בצד השרת

עד כה התנסינו בבנית שאילתות באופן ידני, והרצתן באופן ידני על מסד הנתונים. כעת משהבנו איך השאילתות בנויות מבחינה תחבירית, ומה אנו מצפים לקבל כתשובה, נרצה להריץ את השאילתות מתוך הקוד. כמובן שבצד השרת.

את השאילתות נבנה בתוך משתנה מסוג מחרוזת.לדוגמה, שאילתת שליפה פשוטה תיראה כך:

```
string sql = "select * from users" ;
```

אם נרצה לחפש משתמש מסוים על פי האימייל שלו:

```
string sql = "select * from users where email='" + email + "'" ;
```

המשתנה email מכיל את האימייל של המשתמש אותו אנו מחפשים. ערך המשתנה הגיע כנראה מצד לקוח.
שימו לב לגרש הבודד המוּאר בתכלת לפני ואחרי האימייל.

שאילתות מורכבות יותר דורשות יותר משתנים שנאספו בטופס. לדוגמה, לצורך רישום משתמש חדש נשתמש בשאילתת insert. מכיוון שמדובר בהרבה משתנים, יהיה לנו קל יותר להשתמש בפעולה string.Format. נניח שכל המשתנים כבר נאספו מהטופס:

```
string sql = "insert into Users (name, lastname, email, password, admin) ";
sql += " values ('{0}', '{1}', '{2}', '{3}', 'no' )";
sql = string.Format(sql, name, lastName, email, password);
```

בשלב הראשון נבנה את התבנית של השאילתה כפי שראינו קודם לכן. רשימת השדות הנמצאים בסוגריים אחרי שם הטבלה הם בדיוק שמות העמודות מהטבלה שלנו מופרדים בפסיקים. הסדר אינו חשוב. בסוגריים שאחרי המילה values יופיעו הערכים שנרצה להכניס לטבלה. סדר הערכים חייב להיות זהה לסדר השדות המופיעים בתחילת השאילתה. בדוגמה שלנו: הערך הראשון יוכנס לעמודה name, השני לעמודה lastname. כל הערכים נתונים בתוך גרש בודד ומופרדים בפסיקים.
במקום לשרשר למחרוזת הרבה משתנים והרבה '+' ביניהם, נשתמש ב-string.Format.

string.Format מקבלת כפרמטר ראשון מחרוזת בקרה (התבנית שלנו), ואחריה רשימת משתנים.
בתוך התבנית אנחנו רשאים לכלול סוגריים מסולסלים עם מספר ביניהם, כגון: {3}. הפעולה תחליף את הסוגריים המסולסלים עם המשתנה הרביעי (מתחילים לספור ב-0) אחרי מחרוזת הבקרה.
בדוגמה שלנו יש במחרוזת הבקרה הפנייה ל- משתנים מ-0 עד 3. כלומר string.Format צריכה לקבל 5 פרמטרים: בקרה + 4 ערכים.

DAL – והפניה למסד הנתונים

על מנת לפנות למסד הנתונים נוסיף לפרויקט שלנו את הקובץ המצורף dal.cs.
DAL – Database Access Layer הינו אובייקט המאפשר לנו גישה למסד הנתונים.
אחרי שצירפנו אותו לפרויקט (באמצעות הוסף קובץ קיים), נדאש לשנות לו את שם ה-namespace שיהיה זהה ליתר הפרויקט, אחרת נצטרך לפנות אליו בצורה "ארוכה" יותר.
להלן ממשק חלקי של DAL:

פעולה המקבלת מחרוזת עם נתיב מלא לקובץ מסד הנתונים, ובונה DAL	DAL (string db)
פעולה המקבלת מחרוזת sql מסוג select, מבצעת ומחזירה טבלת DataTable	DataTable GetDataTable (string sql)
מקבלת מחרוזת sql מסוג הוספה, מחיקה או עידכון, מבצעת ומחזירה את מספר הרשומות שהושפעו מהביצוע.	int UpdateDB (string sql)

קובץ מסד הנתונים נמצא במחיצה App_Data באיזור הפרויקט שלנו. אבל הפרויקט כולו יכול להיות ממוקם במקומות שונים במחשבים שונים. לכן נשתמש בפעולה MapPath בכדי לקבל את הנתיב המלא לקובץ שלנו:

string db = MapPath ("~/App_Data/myDatabase.mdf");

הסימון ~ (טילדה), מסמן את מיקום הפרוייקט ברחבי הדיסק. תפקידה של MapPath לתרגם את ~ למסלול המדויק. שימו לב להחליף את שם הקובץ המואר בתכלת בשם הקובץ שלכם....
אחרי ההוראה האחרונה נקבל לדוגמה:

db = " C:\Users\Moshe Steiner\Documents\Visual Studio 2015\Projects\myProject\myDatabase.mdf"

קוד לדוגמה לשליפת כל המשתמשים מהטבלה יהיה:

```
using System.Data;     //     נוסיף בראש הקובץ על מנת לקבל תמיכה בטבלאות

DAL dal;
DataTable dt;

string db = MapPath ( "~/App_Data/myDatabase.mdf" );
dal = new DAL ( db ) ;
string sql = "select * from Users" ;
dt = dal.GetDataTable ( sql );
```

המשתנה dt הוא מסוג DataTable. זהו אובייקט המכיל את תוצאת הרצת השאילתה.

dt.Rows.Count – יציין לנו כמה שורות קיבלנו כתשובה. 0 פירושו: לא נמצאו שורות מתאימות

אם נרצה לשלוף את שם המשתמש בשורה הראשונה נבצע: dt.Rows [0] ["name"].ToString()

במקרה של שאילתות עדכון (הוספה, מחיקה, עדכון), נקבל מספר שלם כתשובה. שמשמעותו על כמה שורות השפענו.... כמה שורות מחקנו, כמה שינינו. במקרה של הוספה נקבל 0 או 1.
0 מסמן שהפעולה נכשלה, אולי כי השתמשנו במפתח שכבר נמצא בשימוש.
1 מסמן שנוספה שורה אחת למסד הנתונים.

הכנסת פרטי משתמש חדש

את הדף המטפל בקליטת פרטי משתמש חדש כבר היכרנו בשלבים המוקדמים של הפרויקט.
בשלבים אלו דאגנו שכל הנתונים ייבדקו בצד הלקוח באמצעות javascript. עכשיו הגיעו הנתונים לצד השרת.
בשלב הראשון נקלוט את כל הפרטים למשתנים "רגילים".

> אסור להניח שהנתונים קיימים בגלל שכבר בדקנו בצד לקוח.
> כבר ראינו שהדפדפן סלחן, ולא יתריע על בעיה.
> בצד השרת שגיאה עלולה לדרוש לקריסה. לדוגמה, אם משתנה הטופס לא מולא כהלכה, או שמו השתנה, אנו עלולים לפנות דרך null.

לאחר שכל הפרטים בידינו, נשתמש ב-DAL על מנת לפנות למסד הנתונים ולברר האם קיים כבר משתמש עם מפתח כזה, ואם לא קיים, נכניס למסד הנתונים.

LOGIN / LOGOUT - כניסה ויציאה

כל משתמש שנרשם לאתר יכול לבצע כניסה לאתר – login. תהליך הכניסה כולל אימות שם וסיסמה מול מסד הנתונים, ועדכון אובייקט ה-Session.
מכיוון שהשרת אינו "זוכר" משתנים לאחר שהדף נטען (stateless, ראו פרק על השרת) הדרך להעביר מידע מעמוד לעמוד היא דרך האובייקט Session.
לשם כך, נדאג בעמוד הבית של הפרויקט, או בדף המאסטר, לאתחל את האוביקט באופן הבא:

```
If (Session["name"] == null)   // first time project is being loaded
{
        Session["name"] = "guest";
        Session["email"] = "";
        Session["admin"] = "no";
}
```

באופן זה, עמודי האתר יכילו את ההודעה Hello guest.

שימו לב:
תהליך יציאה, logout, יהיה זהה לחלוטין למתואר מעלה. כלומר – איתחול ערכי ה-Session.

במהלך תהליך הכניסה, נפנה למסד הנתונים לברר האם קיים משתמש כזה:

```
Select * from Users where email='<email>' and password='<password>'
```

המשתנים בצבע אדום הם הערכים שנקלטו במהלך תהליך הכניסה.
אם נקבל ממסד הנתונים תשובה עם אפס שורות, נדע שאין משתמש כזה. שורה אחת, פירושה: נמצא משתמש המתאים לפרטים.
במקרה זה נעדכן את משתני ה-Session בערכים מתוך מסד הנתונים. לדוגמה:

```
Session["admin"] = dataTable.Rows[0]["admin"].ToString( );
```

שאלה: מדוע פנינו ישירות לשורה 0 בטבלה?

דף מנהל

בדף המנהל תוצג טבלה של כל המשתמשים הרשומים.
כל משתמש בשורה נפרדת. את הנתונים נשלוף ממסד הנתונים בצד השרת.
בנוסף, בכל שורה תתווספנה 3 אפשרויות: מחיקת משתמש, שינוי מצב מנהל, ואיפוס סיסמה.

לשם כך הקוד בצד השרת יופעל מספר פעמים:

1. בקריאה הראשונה לעמוד: אז יפנה השרת למסד הנתונים לשליפת פרטי המשתמשים.
2. במקרה של מחיקת משתמש
3. במקרה של שינוי סיסמה
4. במקרה של שינוי סטטוס

כאשר רשמנו משתמש חדש, השתמשנו בטופס, אשר אחד המאפיינים שלו היו: method="post". המשמעות היא שכל נתוני הטופס עברו דרך האובייקט Request.Form.
דף המנהל מעביר פרמטרים לצד השרת באופן שונה. אין טופס. את הפרמטרים נעביר בשורת הכתובת. בפניה רגילה לדף אינטרנט פשוט מציינים את שמו המלא, למשל: adminPage.aspx.
לצורך העברת פרמטרים נשתמש בפורמט הבא:

adminPage.aspx?param1=value1&param2=value2

כלומר: אחרי שם הדף מופיע סימן שאלה המציין יש פרמטרים. הפרמטרים נרשמים בשמם, הסימן '=' וערכם.
בין הפרמטרים נשתמש ב-'&' כמפריד. הכל נרשם ללא רווחים וללא גרשיים.
כך שאם נרצה למחוק משתמש שהאימייל שלו: user@gmail.com, נפנה לעמוד המנהל באופן הבא:

adminPage.aspx?op=delete&email=user@gmail.com

פניה לעמוד אחר מתוך javascript נעשית באופן הבא:

```
function changePassword(email)
{
	window.location.replace( "AdminPage.aspx?op=password&user=" + email );
}
```

במקרה זה העברנו 2 פרמטרים: op=pass. החלטנו שהפרמטר הראשון שלנו נקרא op וערכו password.
השרת יבין שהכוונה לשינוי סיסמה. הפרמטר השני הוא האימייל של המשתמש עבורו נרצה לשנות סיסמה. ניתן להעביר דם את הסיסמה בתור פרמטר שלישי.

השרת יוכל לגשת לפרמטרים אלו על ידי האובייקט Response.QueryString באופן דומה לטיפול בטופס:

```
Request.QueryString["op"] != null
```

פירושו שנשלח לנו פרמטר בשם op. ערכו יהיה:

```
Request.QueryString[“op”].ToString()
```

כיצד כל התהליך מתבצע?

בדף המנהל קורים מספר תהליכים:

1. בדף המנהל, צד לקוח, נכין תשתית לטבלה. הטבלה תהיה ריקה, אבל תפנה לשרת שייצוק בה תוכן:

```
<table> <%Response.Write(table);%> </table>
```

כלומר, בדף השרת קיים משתנה בשם table שיכיל את תכולת הטבלה.

2. בדף המנהל נכין 3 פונקציות javascript המקבלות אימייל כפרמטר:
    a. deleteUser(email)
    b. changePassword(email)
    c. changeAdmin(email)

פונקציות אלו יתנו מענה לשלושת הפעולות שהוזכרו בהתחלה. הפרמטר email יגרום לכך שהפעולות שנבצע במסד הנתונים יקרו אך ורק למשתמש אחד אותו בחרנו.

3. השרת פונה למסד הנתונים וקורא/שולף את נתוני כל המשתמשים מהטבלה.
הדבר מתבצע על ידי שאילתת select לא מותנית: select * from users.

4. את המשתנה table נאתחל בשורת הכותרת של הטבלה. למשל:

```
<tr> <td>name</td> <td>last name</td> <td>email</td>......</tr>
```

שימו לב: לא נהוג להציג את הסיסמה.

5. עבור כל שורה בטבלה נייצר קוד HTML, כלומר נשרשר למשתנה table קוד בסגנון הבא:

```
<tr>
<td> ערך מתוך טבלת המשתמשים </td>
<td> ערך מתוך טבלת המשתמשים </td>
......
</tr>
```

לערך בתוך טבלת המשתמשים נגיע באופן הבא: נניח ששם המשתנה המחזיק את טבלת ה-SQL הוא dataTable, אזי נבצע לולאה על כל השורות. את האימייל למשל נוכל לקבל:

```
dataTable.Rows[ i ][“email”].ToString()
```

כאשר i הוא מספר השורה בלולאה.

6. בכל שורה בטבלה נרצה להוסיף 3 כפתורים בהתאם לדרישות.
זה יתבצע על ידי הוספת קוד לכפתור לחיצה, button:

```
"<td><button onclick=\"deleteUser('" + email + "');\">delete</button></td>";
```

הסבר: זוהי בעצם שורת HTML הנוצרת בצד השרת.
הכיתוב בדוגמה הנ"ל הוא delete, וההוראה המתבצעת היא קריאה ל-javaScript לפונקציה בשם deleteUser עם פרמטר שהוא האימייל המבוקש למחיקה.
שימו לב: בצבע תכלת מודגשים הסימנים \", שמשמעותם להשתמש בתו גרשיים כפי שהוא, ולא בתור סימון לסוף מחרוזת. ובצבע כתום מודגש התו גרש בודד.

7. לסיכום, תחילת הפעולה בדף המנהל צד שרת היא כעת מורכבת יותר. ואלו הצעדים הכללים שנראה בפעולה PageLoad:
   a. בדיקת הגנה: האם המשתמש הינו בעל הרשאת ניהול. אם אין, טיפול בכניסה לא מורשית.
   b. "פתיחת" גישה למסד הנתונים, על ידי בניית DAL. כל הפעולות הבאות יזדקקו לגישה זו.
   c. האם הגיעה בקשה מיוחדת? בדוגמה שלנו האם הגיע פרמטר op ב-QueryString?
      i. אם הגיע, נוודא שהגיע גם אימייל
      ii. אם הגיע אימייל, נברר מה התבקשנו לבצע: מחיקה, שינוי סטטוס או שינוי סיסמה
      iii. זימון הפעולה שמבצעת את הפעולה הדרושה
   d. הצגת טבלת המשתמשים.

שימו לב: מכיון שהצגת המשתמשים אינה מותנית בסעיף הקודם, הרי שכל שינוי בטבלה משתקף מיד.

למנהל אסור לבצע שינויים על עצמו: לא למחוק, ולא לשנות סטטוס.

חישבו איך למנוע מהמנהל לבצע פעולות אלו על עצמו.
זיכרו שהמנהל יכול כמו כל משתמש להיכנס לדף עידכון פרטים ולבצע שינוי סיסמה או כל פרט אחר, באופן הזהה למשתמשים אחרים.

דף עדכון נתוני משתמש

דף זה דומה מאד לדף רישום נתונים, אם כי מהלך העבודה קצת שונה.
הזכות להגיע לדף זה היא למי שביצע תהליך כניסה מוצלח. כלומר פרטיו ידועים לנו.
במעבר לדף עידכון הפרטים נשלוף את הפרטים מתוך מסד הנתונים, ונעדכן משתנים בערכים המתאימים.

בדף עצמו שורת קלט תיראה באופן הדומה לזה:

```
<input type="text" id="fName" name="fName" value="<%Response.Write(firstName);%>" />
```

כך שלאחר שליפת הנתונים המשתנה firstName יכיל את השם הפרטי של המשתמש, ויוצג כערך ראשוני בשדה הקלט. למשתמש תהיה אפשרות לשנות אותו.

בדף העידכון לא נציג סיסמאות באופן גלוי. וגם הפעם נדאג לאפשרות של אימות סיסמה.
את שם האימייל נציג אך לא נאפשר לשנות.

שדה Select

ה-Select מורכב ממספר שורות, שכל אחת מציעה ערך נוסף לרשימה. ובנוסף יש שורה המציינת מה יוצג בהתחלה, למשל:

```
<option value="" selected>בחר</option>
```

במקרה זה נשנה את המילה **בחר** בביטוי <%Response.Write(userValue);%>

שדה Radio

אנו יכולים לבחור איזה מהאפשרויות תיבחרנה בזמן העלאת הטופס. וזאת על ידי הוספת התכונה: checked="checked"
לתגית input. אבל אם נוסיף לכל השורות, הרי לא נצפה שכל האפשרויות "תידלקנה".
לכן הדרך לבצע זאת על ידי הוספת Response.Write שונה בכל שורה. למשל, עבור האופציה הראשונה נציג את opt1 ועבור השניה את opt2. בזמן השליפה ממסד הנתונים נדאג שהמשתנה התאים יכיל את "checked"=checked, והשני מחרוזת ריקה.

שדה CheckBox

זהה לחלוטין לשדה רדיו.

עם קבלת הטופס

ראשית בררו לעצמכם איך אתם יודעים שנכנסתם לקוד בפעם השניה (בראשונה קראתם נתונים מתוך מסד הנתונים)......איך עושים זאת? ובכן יש מספר דרכים.
עד היום ניגשנו לאחד משדות הטופס ובררנו האם הוקלד בו ערך
ניתן להשתמש במשתנה IsPostBack, אבל השימוש בו יותר מורכב.
הדרך הפשוטה לברר "If (Request.Method) == "POST בפעם הראשונה ערך המשתנה יהיה GET, ובפעם השניה (והשלישית) באם הגדרנו בטופס, יתקבל הערך POST

עכשיו אנחנו יודעים שהגיע הטופס עם הערכים המעודכנים.
נבנה שאילתת SQL מסוג UPDATE. המעדכנת אך ורק את המשתמש הרצוי.
שימו לב **לא** לאפס את הסיסמה של המשתמש אם הוא לא עידכן אותה....

תיעוד הפרויקט - תיק פרויקט

הגשת הפרויקט תהיה מלווה במסמך תיעוד – תיק פרויקט. התיעוד יכיל את כל הפרטים החשובים להבנת הפרויקט ולמבנה שלו:

1. עמוד פתיחה
   בעמוד הפתיחה יופיעו הפרטים הבאים:
   a. שם הפרויקט, או שם האתר.
   b. שם התלמיד המגיש, כולל תעודת זהות
   c. שם בית הספר
   d. שם המורה המלווה
   e. שנת הלימודים
   f. תמונת העמוד הראשי

2. הרחבה על נושא האתר
   הרחבה על הנושא בו עוסק האתר, איזה סוגי מידע ניתן למצוא בו (הצהרת כוונות), למי מיועד האתר, איזה סוגי קהל יעד. לדוגמה: חובבי רכיבת סוסים בסגנון אנגלי.
   מדוע בחרתי בנושא זה לפרויקט שלי?

3. רשימת הדפים באתר
   טבלה המכילה את רשימת כל הדפים באתר. הטבלה תכלול: שם הקובץ, כותרת העמוד, תיאור קצרצר של מטרת הדף, ולמי יש גישה לדף זה. לדוגמה:

מס'	שם קובץ	כותרת הדף	תאור	למי יש גישה
1	Update.aspx	עדכון פרטים	דף לעידכון פרטי המשתמש	משתמש רשום

4. רשימת שפות וסביבות עבודה
   כל השפות בהן משתמש הפרויקט: HTML, CSS, javascript, SQL, C#
   סביבת עבודה – ויז'ואל סטודיו 2015
   מטרת סעיף זה לפרט מה נדרש מהמחשב שיריץ את האתר. לדוגמה, האם מותקן עליו ויז'ואל או לא.
   האם השתמשנו ביכולות נוספות כמו TrueType או Jason

5. הפעלת האתר
   במסד הנתונים של האתר המוגש חייבים להיות לפחות 2 משתמשים: מנהל ומשתמש רגיל. בסעיף זה יש לרשום את שם המשתמש (אימייל) והסיסמה של כל אחד מהם, על מנת לאפשר לקורא המסמך כניסה ראשונית לאתר.

6. דף רישום

לדף זה חשיבות רבה בגלל בדיקות האימות המתבצעות ב-javascript. בסעיף זה נציין את כל השדות אותם אנו קולטים, את סוג הקלט (טקסט, סיסמה, כפתור רדיו, וכו'). יש לפרט את בדיקות האימות המתבצעות ב-javascript.

7. מסד הנתונים – database
בסעיף זה נתאר את מסד הנתונים שלנו. עבור כל טבלה במסד הנתונים שנשתמש נפערוך טבלה עם הפרטים הבאים: שם הטבלה, שם השדה / עמודה, סוג הנתונים (מחרוזת, מספר שלם, וכיו"ב), תיאור קצר של השדה.

8. שאילתות SQL
רשימת כל השאילתות שאנו משתמשים בהם בפרויקט. ברשימה יש לכלול את שם הדף, השאילתה בה אנחנו משתמשים, ומה מטרתה. לדוגמה:

שם הדף	השאילתה	מטרת השאילתה
Register.aspx	Select * from Users where email='<email>'	בדיקה האם קיים משתמש שזה האימייל שלו. אם כן, השאילתה תשלוף את כל הנתונים שלו.

נספח: DAL

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace <your project namespace>
{
public class DAL
   {
      private string dbPath;
      private string ConnectionString;
      private string connectionStr = @"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={0};Integrated Security=True";
      private SqlConnection sqlConnection;
      private SqlCommand command;
      private SqlDataAdapter adapter;

      public DAL(string dbPath)
      {
         this.dbPath = dbPath;
         ConnectionString = string.Format(connectionStr, this.dbPath);
         sqlConnection = new SqlConnection(ConnectionString);
      }

      public DataTable GetDataTable(string sql)
      {
         DataTable results = new DataTable();

         using (sqlConnection = new SqlConnection(ConnectionString))
         using (command = new SqlCommand(sql, sqlConnection))
         using (adapter = new SqlDataAdapter(command))
            adapter.Fill(results);
         return results;
      }

      public int UpdateDB(string sql)
      {
         int rowsEffected;
         sqlConnection = new SqlConnection(ConnectionString);
         command = new SqlCommand(sql, sqlConnection);
         adapter = new SqlDataAdapter(command);
         command.CommandText = sql;
         sqlConnection.Open();
         rowsEffected = command.ExecuteNonQuery();
         sqlConnection.Close();
```

```
            return rowsEffected;
        }

        public DataSet GetDataSet(string sql)
        {
            DataSet ds = new DataSet();

            command = new SqlCommand(sql, sqlConnection);
            adapter = new SqlDataAdapter(command);
            command.CommandText = sql;
            adapter.SelectCommand = command;
            adapter.Fill(ds);
            return ds;
        }
    }
}
```

מדריך למורה

סדר הוראה, טיפים ודגשים

1. מבוא – מהי תקשורת האינטרנט? לקוח-שרת, סוגי דפדפנים.
2. מה מצפה לנו בפרויקט? תכולה, שפות תכנות
3. הדגמת פרויקטים משנים קודמות.
4. הנחית התלמידים "להתביית" על נושא לפרויקט. להתחיל לאסוף רעיונות וחומרים.
5. מהי שפת תגיות?
6. היכרות ראשונה עם HTML דרך ++notepad.
7. כותרות, בלוקים, ועוד תגיות שימושיות
8. תמונות
9. קישורים <a…..>
10. **תרגיל ראשון** בבנית לפחות 4 עמודים עם קישוריות. בכל עמוד יש כותרת שונה, ולפחות תמונה אחת. מכל עמוד ניתן להגיע לכל עמוד אחר. התמונות נמצאות במחיצה שונה מהדפים.
11. עקרונות עיצוב CSS
12. רשימות
13. טבלה, טופס, כפתור submit
14. קלט – תגית input
15. **תרגיל שני** בנית טופס רישום. שימוש בטבלה + שדות קלט
16. Javascript – סדר הרצה, משתני var, פונקציות, החזרת ערך, פונקציות בנויות: alert, confirm, prompt, וכיו"ב
17. גישה לשדות בדף getElementById(“name”).value
18. **תרגיל שלישי** – הרחבת תרגיל שני – הוספת עמודה בטבלה כך ששם התא זהה לשם תא הקלט עם "_" לפניו. כלומר: אם שדה הקלט הוא: <input id=”email”>, אז העמודה הבאה בטבלה תהיה: <td id=”_email”>. מטרת התרגיל: כתיבת פונקציה פשוטה ב-javascript המעתיקה את הערכים שנקלטו לעמודה על ידם.
19. בדיקות javascript – בדיקות חוקיות: שם שאיננו ריק, אימייל חוקי (@ ונקודה), סיסמה עם לפחות שני כללים לבדיקה: למשל אורך, ספרות, אותיות גדולות קטנות, ועוד.
20. **תרגיל רביעי** – פיתוח על תרגיל שלישי – הפעלת הבדיקות והוצאת שגיאות לעמודה הנוספת. בדוגמה הקודמת לתא ”_email“. יש להוציא את השגיאות במרוכז ולא בנפרד.
21. מעבר לסביבת סטודיו
22. בנית מאסטר
23. הפעלת כל הדפים שנוצרו עד עכשיו בסביבת aspx
24. Session. הוספת "שלום אורח" לכל הדפים.
25. **תרגיל חמישי** – בנית עמוד כניסה, או שילוב יכולת כניסה בעמוד הבית. בקוד השרת יש לשמור שני משתנים עם השם והסיסמה, ולבדוק האם הכניסה חוקית או לא. כניסה לא חוקית תוציא הודעה מתאימה. כניסה מוצלחת תשנה את ערכי אובייקט ה-Session, ותציג "שלום <משתמש>". מימוש יציאה.....
26. מבוא למסד נתונים

27. בנית טבלת תירגול והרצת שאילתות: טבלה של תלמידים והישגי ספורט או ציונים. או טבלת רכבים ותצרוכת דלק, זיהום אויר. וכיו"ב. המטרה – שימוש אינטראקטיבי בשאילתות.
28. בנית טבלת המשתמשים בהתאם לנתוני טופס הרישום. הזמנות טובה לשיעור במחרוזות: string.Format
29. בנית קוד השרת לטופס הרישום – ללא חיבור
30. DAL.
31. **תרגיל שישי -** הצגת שתי שאילתות SQL בטופס הרישום: שאילתת שליפת המשתמש (לבדוק שלא קיים) ושאילתת הכנסת משתמש חדש. התלמידים צריכים להעתיק עם העכבר את השאילתות ולהריץ באופן ידני בסטודיו על מנת לוודא שאין שגיאות תחביריות, והכל מבוצע כפי שרצו.
32. תלמיד שהצליח לקבל שאילתות תקינות מתחבר למסד הנתונים באמצעות DAL בעזרת המורה.
33. בנית דף מנהל. שלב ראשון – הצגה בלתי מותנית של כל המשתמשים. בשיעור יציג המורה את אופן שליפת הנתונים מהטבלה DataTable המגיעה לאחר הרצת השאילתה. כיצד היא מגיעה לדף הלקוח.
34. דף מנהל שלב שני – הוספת 3 כפתורים בכל שורה: מחיקה, שינוי סטטוס מנהל, איפוס סיסמה. המורה ידגים את השילוב בין הקוד שיותר את הכפתור, הקוד ב-javascript המזמן עמוד לביצוע, והעברת פרמטרים בשיטת GET.
35. למנהל אסור לבצע שינויים על עצמו, אלא דרך דף עידכון פרטים
36. דף עידכון פרטים
37. הגנות כניסה: בפני משתמשים לא רשומים, ובפני מי שאינו מנהל לדף המנהל.
38. **הגשה סופית** – התלמיד מציג את הפרויקט על תפקודיו השונים וצריך להפגין ידע ובקיאות עד כדי ביצוע משימת שינוי/עידכון/תיקון במהלך ההגשה.

בהצלחה