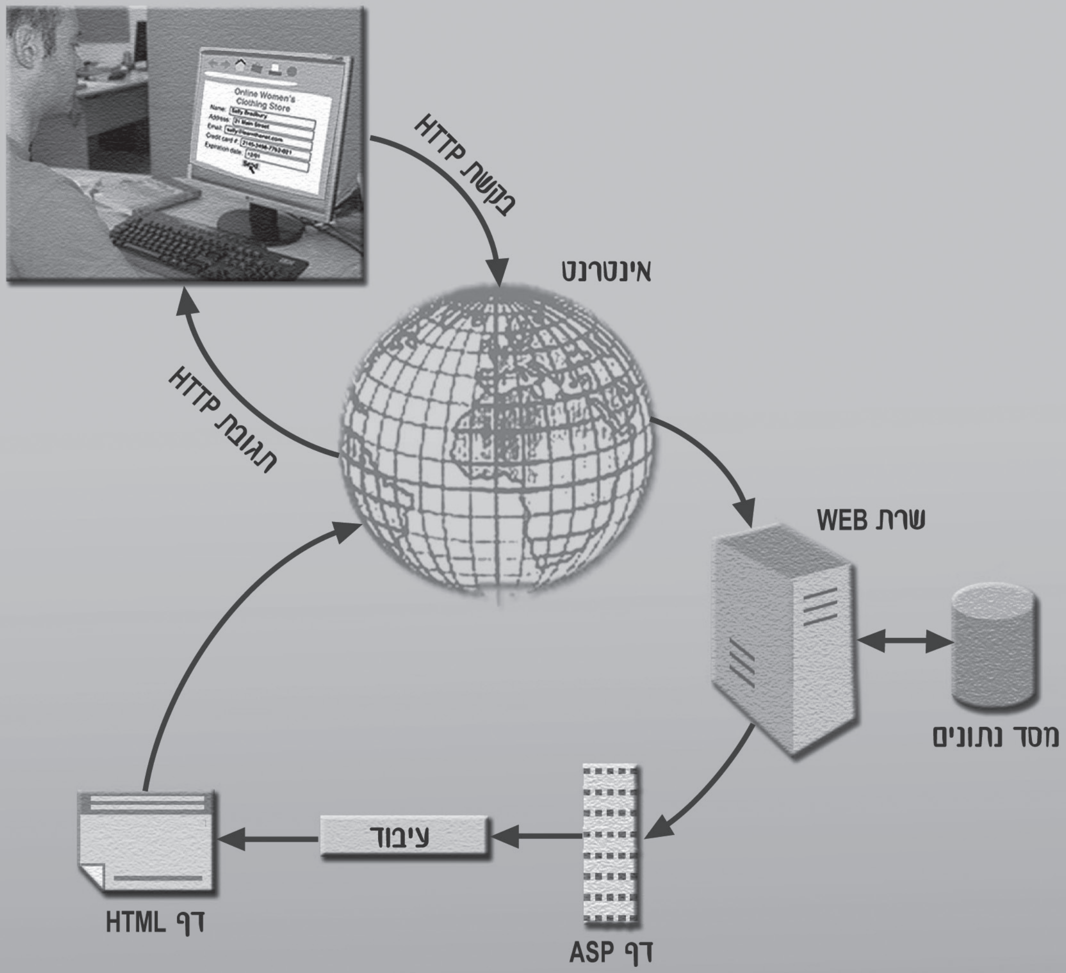
מבוא לתכנות
בסביבת האינטרנט בשפת C#



מטח
המרכז לטכנולוגיה חינוכית

האוניברסיטה הפתוחה
בית הספר לטכנולוגיה

המרכז הישראלי לחינוך
מדעי-טכנולוגי
ע"ש עמוס דה-שליט

משרד החינוך
האגף לתכנון ולפיתוח תכניות לימודים

גף
אישור
ספרי
לימוד
באישור
משרד החינוך
אישור מס׳ 4267

כתיבה
איריס צור ברגורי (פרקים 1-5)
אקת׳ים חאג׳ יחיא (פרק 6)

ייעוץ אקדמי
ד״ר צביקה פירסט ז״ל

עריכה פדגוגית
שרה פולק

ייעוץ וקריאה
ד״ר איל שפרוני
אקת׳ים חאג׳ יחיא
איזבלה טבלין

עריכה לשונית
לימור וייסמן רביד
אראלה ברילנט

עריכה גרפית והפקה
אביבה אבידן

עיצוב עטיפה
אבי חתם

האוניברסיטה הפתוחה
בית הספר לטכנולוגיה

מטח
המרכז לטכנולוגיה חינוכית

משרד החינוך
האגף לתכנון ולפיתוח תכניות לימודים

המרכז הישראלי לחינוך
מדעי-טכנולוגי
ע"ש עמוס דה-שליט

©תשע״א – 2011. כל הזכויות שמורות למשרד החינוך.
בית ההוצאה לאור של המרכז לטכנולוגיה חינוכית, קרית משה רואו, רח׳ קלאוזנר 16, רמת-אביב, ת״ד 39513, תל-אביב 61394.
The Centre For Educational Technology, 16 Klausner St., Ramat-Aviv, P.O.Box 39513, Tel-Aviv, 61394. Printed in Israel.

זכויות הקניין הרוחני, לרבות זכויות היוצרים והזכות המוסרית של היוצר/ים בספר זה מוגנות. אין לשכפל, להעתיק, לסכם, לצלם, להקליט, לתרגם, לאחסן במאגר מידע, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי, מכני או אחר, כל חלק שהוא מספר זה. כמו כן, אין לעשות שימוש מסחרי כלשהו בספר זה, בכולו או בחלקים ממנו, אלא אך ורק לאחר קבלת רשות מפורשת בכתב ממשרד החינוך.

תוכן העניינים

פרק 1 מודל שרת-לקוח 7

1.1 הקדמה 7

1.2 ה-Web ו-HTTP 9

1.3 המבנה של יישומי רשת ושל פרוטוקולים 11

1.4 כתובות באינטרנט 17

1.5 שימוש במאתר משאבים אחיד לזיהוי משאב 23

1.6 תהליך התקשרות שרת-לקוח בפרוטוקול HTTP 25

1.7 יישום שרת-לקוח בטכנולוגיית ASP 27

פרק 2 יצירת דפי HTML ו-CSS 33

2.1 מבוא לשפת HTML 33

2.2 שימוש בתגים לכתיבת קובצי HTML 36

2.3 פנייה לדף סטטי בשרת 56

2.4 גיליונות סגנון מדורגים (CSS) 57

2.5 פעילות מסכמת – בניית דף HTML 62

נספח א׳ – יצירת מסמך HTML והרצתו בסביבת Microsoft Visual Studio 66

נספח ב׳ – סיכום תגי HTML 70

פרק 3 תכנות שרת – יצירת דף דינמי ותכנות "חסר מצב" 73

3.1 יצירת דף דינמי ושליחת נתונים משרת ללקוח 73

3.2 העברת נתונים בין שרת ולקוח 90

3.3 פניה לדף חדש 116

3.4 תכנות חסר מצב 120

3.5 כניסה מאובטחת ושימוש בסיסמה 147

3.6 פעילות מסכמת – בניית אתר למשחק מכונת המזל 158

נספח ג׳ – יצירת קובץ aspx בסביבת Visual Studio 163

פרק 4 תכנות שרת – שימוש במסדי נתונים 171

4.1 מבוא 171

4.2 המודל הטבלאי 172

4.3 מבוא לשפת שאילתות מובנות (SQL) 175

4.4 עבודה עם מסד נתונים 185

4.5 תהליך אחזור ועיבוד נתונים בגישה הלא-מקושרת 188

4.6 תהליך אחזור ועיבוד נתונים בגישה המקושרת 227

4.7 פעילות מסכמת – בניית אתר למשחק 'מכונת המזל' 238

נספח ד' – שימוש במסד הנתונים SQL Server 241

פרק 5 תכנות צד לקוח 255

5.1 שילוב פעולות בצד לקוח 255

5.2 התחביר של שפת **JavaScript** 259

5.3 מבני בקרה 265

5.4 פונקציות 276

5.5 שימוש בעצמים מוגדרים מראש 279

5.6 טיפול באירועים 285

5.7 כתיבת תסריט לבדיקת תקינות נתוני טופס 296

5.8 ייבוא קבצי **JavaScript** 312

5.9 תרגיל משימה מלווה 313

פרק 6 עקרונות XML 315

6.1 מבוא 315

6.2 מבנה מסמך **XML** 322

6.3 סכימת **XML (XSD)** 331

6.4 שמירה ואחזור נתונים בקבצי **XML** 341

6.5 הצגת מסמכי **XML** כ-**HTML** 346

6.6 סידרות **(Serialization)** 350

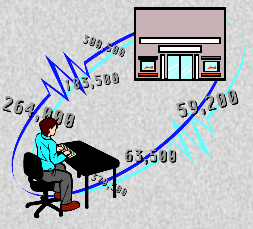
פתח דבר

ספר לימוד זה מתאים לתוכנית הלימודים החדשה – ***'תכנות בסביבת האינטרנט'*** (היחידה השלישית במקצוע **'מדעי המחשב'**). מטרת הספר היא לסייע לתלמידים ולמורים בלימוד התכנית הלימודים החדשה המדגישה את **עקרונות העבודה בסביבת שרת/לקוח ותכנות בצד השרת ובצד לקוח** (בשפת C#). הנושאים בספר מאורגנים סביב פרויקט לדוגמה שמתאים בתכניו לפרויקט הסיום של המקצוע ***'תכנות בסביבת האינטרנט'***.

הספר מכיל מבוא עיוני, דוגמאות ותרגילים בנושאים הבאים:

- עקרונות העבודה במודל שרת/לקוח באינטרנט, אתר דינאמי, פרוטוקול HTTP, טכנולוגית ASP;
- יצירת דפי HTML ו-CSS;
- פעולות בסיסיות בשרת: קבלת בקשה, הכנת דף ושליחת תגובה ושימוש בעצמים שמאפשרים התמודדות עם תכנות חסר-מצב (Stateless programming);
- עבודה עם מסד נתונים בגישה הלא-מקושרת ובגישה המקושרת;
- תכנות של צד הלקוח בשפת תסריטים JavaScript;
- עקרונות XML.

ברצוננו להודות לגב' איזבלה טבלין ותלמידיה מבית-הספר "קוגל" בחולון ומבית-הספר "שבח-מופת" בתל אביב על שהשקיעו מזמנם וליוו את פיתוח הספר בהארותיהם, ובהערותיהם שתרמו רבות לשיפורו של הספר.



פרק 1

מודל שרת-לקוח

1.1 הקדמה

רשת מחשבים מוגדרת כאוסף של מחשבים המקושרים ביניהם ומטרתה לאפשר העברת נתונים בין מחשב למחשב. המילה אינטרנט **(internet)** היא צירוף של שתי מילים: inter-ו net , שפירושן רשת שמורכבת מרשתות רבות המחוברות ביניהן לרשת עולמית הנקראת רשת האינטרנט. מרבית יישומי האינטרנט בנויים לפי המודל של שרת-לקוח[1]. יישומים הבנויים לפי מודל זה מורכבים משני חלקים: ה**לקוח** **(client application)** וה**שרת** **(server application)**. הלקוח והשרת מתקשרים כפי שמוגדר **בפרוטוקול (אוסף הנהלים)**. המחשבים של משתמשי הקצה מריצים את יישום הלקוח, ומחשב אחר, למשל מחשב של ארגון שנותן שירות, מריץ בדרך-כלל את יישום השרת. לדוגמה, כאשר אתם מעוניינים לרכוש ספר, אתם עשויים לגלוש לאתר של חנות הספרים הידועה amazon.com. כאשר אתם עושים זאת, אתם בבחינת משתמשי קצה שמפעילים יישום לקוח (דפדפן), שמתקשר ליישום שרת שרץ במחשב של חברת amazon ומקבל ממנו מידע על הספרים שהחנות מוכרת. דוגמאות נוספות הן גלישה לדף הבית של עיתון לצורך קריאת חדשות; גלישה לאתר של תיאטרון כדי לקבל פרטים על הצגות; גלישה לאתר של חנות מוזיקה כדי לקבל פרטים על שירים וזמרים וכדי לראות ולשמוע קטעי וידאו של זמרים ולהקות.

בכל המקרים הללו אנו משתמשים בדפדפן כדי לגלוש לאתרים הרצויים. הדפדפן מתקשר עם יישום שרת שמורץ במחשב אחר (לדוגמה במערכת העיתון), והמידע מועבר מהשרת אל הלקוח ומוצג לפניו באמצעות הדפדפן. ישנם יישומים שבהם אנו נדרשים להכניס נתונים מתאימים. לדוגמה, כאשר אנו רוצים לרכוש תקליטור באמצעות האינטרנט, אנו גולשים באמצעות הדפדפן לאתר הבית של החנות. באתר החנות מוצג לפנינו מידע על סוגי

---

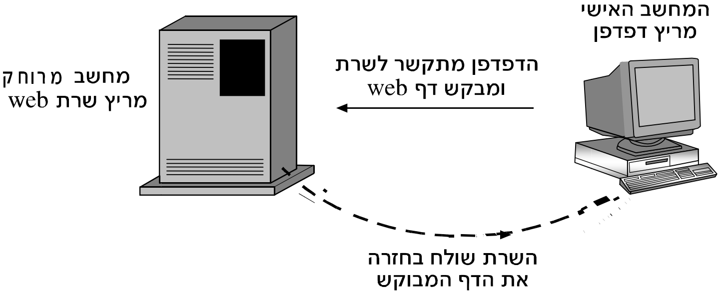
[1] ישנם מודלים נוספים למימוש יישומי רשת מלבד מודל שרת-לקוח, לדוגמה המודל peer-to-peer. בספר זה נעסוק רק במודל שרת-לקוח.

התקליטורים שניתן לקנות בחנות. אנו יכולים לבחור באפשרויות מסוימות באמצעות תפריטים או להכניס נתונים בתיבות טקסט. בהתאם לבחירות שביצענו, יישום השרת מאחזר מידע מתאים ושולח אותו לתצוגה בדפדפן.

שאלה 1.1

הביאו דוגמה לשני יישומי אינטרנט שאתם מכירים. הגדירו מיהו הלקוח ומיהו השרת בכל יישום כזה.

במודל שרת-לקוח יישום הלקוח יוזם את הקשר ויישום השרת מאזין באופן פסיבי וממתין ליצירת קשר. כאשר לקוח מעוניין לקבל שירות הוא שולח בקשה ליצירת קשר ליישום השרת המתאים. יישום השרת מקבל את הבקשה ומשיב עליה. כך נוצר הקשר בין יישום הלקוח ובין יישום השרת. בספר זה נשתמש בקיצור במונחים לקוח (במקום יישום לקוח) ושרת (במקום יישום שרת).



איור 1-1
התקשרות של לקוח עם שרת Web

אפשר לדַמות את ההתקשרות של לקוח עם שרת לשיחת טלפון בין אדם המבקש מידע ובין נציג שירות לקוחות בארגון כלשהו. הנציג צריך לטפל בפניות של לקוחות, לכן הוא ממתין לצלצול הטלפון. כאשר לקוח רוצה לקבל מידע כלשהו, הוא מתקשר לשירות הלקוחות. לאחר שנוצר הקשר הוא מעביר את הבקשה, מקבל את המידע ומנתק את ההתקשרות.

יישום שרת יכול לטפל בו-זמנית במספר רב של לקוחות. למעשה, אפשר לדמות את השרת לקבוצה של נציגי שירות לקוחות בארגון כלשהו. כאשר לקוח מתקשר לשירות הלקוחות, השיחה מועברת לנציג הפנוי. כל הנציגים מספקים את אותו סוג מידע, אך כל אחד מהם מטפל בכל רגע בבקשה של לקוח אחד.

לאחר יצירת הקשר בין השרת ובין הלקוח, הקשר הוא דו-סטרי וסימטרי. השרת שולח מידע ללקוח, והלקוח שולח מידע לשרת. הלקוח יעביר נתונים שמתייחסים לבקשתו, והשרת יספק את המידע המבוקש. מחשב משמש כשרת כאשר רצה בו תוכנת שרת, המקבלת פניות ממחשבים אחרים. תוכנת השרת רצה כל הזמן. לעומת זאת, תוכנת הלקוח מופעלת על-ידי משתמש. כדי להקל על המשתמש לתפעל את תוכנת הלקוח, תוכנות לקוח מכילות, על-פי רוב, ממשק משתמש גרפי וחלונאי. ממשק זה נועד להציג את המידע באופן ברור ומושך בפני המשתמש. ממשק גרפי זה נקרא **Graphic User Interface – GUI**. כאמור, את תוכנת השרת לא מפעיל משתמש, ועל כן, לרוב היא לא מכילה ממשק משתמש גרפי (GUI).

1.2 ה-Web ו-HTTP

האינטרנט מכיל יישומי רשת שונים, ביניהם יישומי דואר אלקטרוני, הורדת קבצים, הודעות מיידיות, קבוצות דיון ועוד. יישום Web הוא מיישומי האינטרנט המלהיבים ביותר. כגולשי אינטרנט שאלתם את עצמכם בוודאי לא פעם את השאלות האלה: כיצד מגיע המידע אל חלון הדפדפן שלי? כיצד מתקשר המחשב שלי אל מחשבים אחרים המחוברים לרשת? בסעיף הזה נענה על השאלות האלה.

הַמַּאֲרָג הַכְּלַל-עוֹלָמִי, ה-**World Wide Web (WWW)**, היא מאגר מידע עצום ומקוון שמאפשר למשתמשים לדלות מידע באמצעות תכנית הנקראת, כידוע, **דפדפן** (**browser**). הדפדפן הוא תוכנת לקוח המאפשרת למשתמש לגלוש בדפים המקושרים זה לזה באמצעות מערכת היפר-מדיה.

מערכת היפר-מדיה היא הרחבה של מערכת **היפר-טקסט (hypertext)**. במערכת היפר-טקסט המידע אגור בקבוצה של מסמכים (documents); כל מסמך יכול לכלול קישורים

למסמכים האחרים בקבוצה. כל קישור מיוצג על-ידי פריט במסמך הניתן לבחירה. בחירת הפריט, על-ידי הקלקת עכבר (mouse click), גורמת להצגת המסמך הקשור. ההבדל בין היפר-מדיה להיפר-טקסט הוא שבמערכת היפר-טקסט המסמכים כוללים טקסט בלבד, בעוד שבמערכת היפר-מדיה מסמך יכול לכלול טקסט, תמונות (images), קטעי קול (audio clips), קטעי הנפשה (animations) ועוד.

ה-Web, כמו כל יישום רשת, הוא מערכת **מבוזרת** (**distributed**), כלומר, מערכת המפוזרת במספר שרתים, במספר מחשבים ובמקרה של ה-Web – גם במספר ארצות. יתרונות הביזור הם שכאשר שרת אחד נופל, המערכת כולה אינה נופלת; כמו-כן, המערכת איננה נשלטת על-ידי גורם אחד. עם זאת, לביזור יש גם חיסרון: המערכת אינה מסוגלת להבטיח את התוקף של קישורים. כאשר מנהל אתר מוחק מסמך כלשהו או מעביר אותו למקום אחר – למדריך אחר או למחשב אחר, פג תוקפו של הקישור למסמך הזה. כאשר בוחרים בקישור שפג תוקפו, מקבלים הודעה כמו: "דף זה אינו ניתן להצגה" (This page can not be displayed).

מסמך ב-Web נקרא גם **דַּף הַמַּאֲרָג**, או בקיצור **דף** (**page**). המונח **אתר** (**site**) מתייחס לקבוצה של דפי Web ששייכים לארגון. הדף הראשי של האתר נקרא **דף הבית** (**home page**). כאשר נכנסים לאתר (כלומר, יוצרים קשר עם השרת המתאים) מוצג דף הבית של האתר (אלא אם ביקשנו דף אחר).

לסיכום, יישומי רשת הם בראש ובראשונה יישומים **מבוזרים**, כלומר, כל יישום רשת מורכב משני יישומים שמורצים במחשבים שונים. נתבונן למשל ב-Web; זהו יישום רשת המורכב מיישום לקוח – דפדפן שרץ במחשב של המשתמש, ומיישום שרת – שרץ במחשב אחר. שני היישומים האלו רצים בו-זמנית ומתקשרים ביניהם על-ידי החלפת **הודעות** (**messages**) באמצעות רשת התקשורת.

ייצוג המסמכים

דף Web יכול להיות מורכב מכמה קבצים. לרוב קיים קובץ טקסט בסיסי וכמה קבצים הקשורים אליו. הקבצים המקושרים יכולים להיות תמונות בפורמט כמו JPEG או GIF, יישומוני ג'אווה (Java applets), קובצי קול (audio files) ועוד.

כדי שגורמים רבים ברחבי העולם יוכלו לשתף פעולה, יש לקבוע תקן או שפה מוסכמת לייצוג מסמכי Web; כך יוכלו דפדפנים, המיוצרים על-ידי חברות שונות, להציג מסמכים בצורה דומה. השפה צריכה לאפשר להציג פריטי טקסט, גרפיקה, תמונות, קישורים ועוד. כמו-כן היא צריכה לקבוע את אופן פריסת הטקסט בתצוגה ואת סדר הצגת הפריטים. בפרק הבא נעסוק בשפת ה-**HTML (Hyper Text Markup Language)**, הנקראת גם **שפת סימון של היפר-טקסט**. שפת ה-HTML מאפשרת להציג את דפי ה-Web ומשמשת לקביעת עיצוב (format) של מסמכי היפר-טקסט. שפה זו מאפשרת למשל לקבוע את הגודל, את הצבע ואת סוג הגופן (font) של האותיות, מאפשרת לקבוע מעברי שורות או פסקאות וכו'. כמו-כן נציג בפרק הבא תקן שנקרא **CSS (Cascading Style Sheets)**, או **עיצוב סגנונות מדורגים**. תקן זה מאפשר קיום הפרדה מוחלטת בין התוכן לעיצוב ובכך מאפשר לשלוט על העיצוב הכללי של האתר.

רוב התקנים (הסטנדרטים) המשמשים באינטרנט עוצבו על-ידי ארגון בשם **התאחדות הרשת הכלל עולמית**" או בקיצור **W3C**[2]. הארגון מהווה מעין פורום פתוח בעבור חברות וארגונים בעלי עניין באינטרנט; חברי הארגון קובעים תקנים כלל-עולמיים המאפשרים לקיים תקשורת בין שרתים לתוכנות-לקוח בצורה אחידה, וללא תלות בתוכנה שרצה על כל אחד מהמחשבים, בסוג המחשב או בסוג הצג.

1.3 המבנה של יישומי רשת ושל פרוטוקולים

מאחר שיישום השרת ויישום הלקוח עשויים לרוץ בשני מחשבים מרוחקים, יש לקבוע כללים להתקשרות ביניהם בין המחשבים, כלומר, יש לקבוע **פרוטוקול**. הפרוטוקול קובע את המבנה של ההודעות, את סדר שליחתן ואת הפעולות שיש לנקוט כתגובה לכל סוג של הודעה. בהתאם, גם יישום ווב הבנוי לפי מודל של שרת-לקוח מורכב ממספר רכיבים: יישום השרת, יישום הלקוח ופרוטוקול. יישום Web טיפוסי מורכב מהרכיבים האלה:

- דפדפנים (יישום הלקוח);
- שרתי הַמַּאֲרָג או שרתי Web (יישום השרת);

[2] כתובת האתר של התאחדות הרשת הכלל–עולמית: http://www.w3c.org/

- פרוטוקול העברת תמליל-על הוא פרוטוקול HTTP ‏(Hyper Text Transfer Protocol). פרוטוקול זה מגדיר את המבנה ואת הסדרה של ההודעות שמועברות בין הדפדפן לשרת ה-Web;
- תקן להצגת מסמכים (שנקרא HTML).

יישום אינטרנט נוסף הבנוי לפי המודל שרת-לקוח הוא יישום דוא״ל (email). יישום דוא״ל טיפוסי מורכב מהרכיבים האלה:

- יישומי לקוחות דוא״ל שמאפשרים למשתמשים לקרוא ולשלוח הודעות (למשל outlook או gmail);
- יישומי שרתי דוא״ל שמאחסנים את תיבות הדוא״ל של המשתמשים ומטפלים בהעברת הדוא״ל;
- פרוטוקולים שמגדירים את המבנה של הודעת הדוא״ל; כיצד מועברות הודעות בין שני שרתי דוא״ל, כיצד מועברות הודעות בין לקוח דוא״ל לשרת דוא״ל וכיצד יש לפרש את התוכן של ההודעות.

שאלה 1.2

כאשר תכנית מחשב מורצת במחשב כלשהו, ההוראות והנתונים שהתכנית עושה בהם שימוש נשמרים בזיכרון של המחשב. פרטו היכן נשמרים ההוראות והנתונים ביישום רשת המממש מודל שרת-לקוח.

קיימים פרוטוקולים רבים למטרות שונות וכל פרוטוקול מותאם לביצוע פעולות ותפקידים שונים. הפרוטוקול המפורסם ביותר הוא פרוטוקול HTTP שהזכרנו קודם לכן . פרוטוקול נפוץ נוסף הוא פרוטוקול **ftp ‏(File Transfer Protocol)** שהוא **פרוטוקול להעברת קבצים** הפועל גם כן במודל שרת-לקוח. פרוטוקול זה מה מגדיר את מבנה ההודעות השונות שאפשר לשלוח בפרוטוקול זה, לדוגמה הודעה להתחברות לשרת, הודעה להעלאת קבצים או הורדת קבצים ועוד.

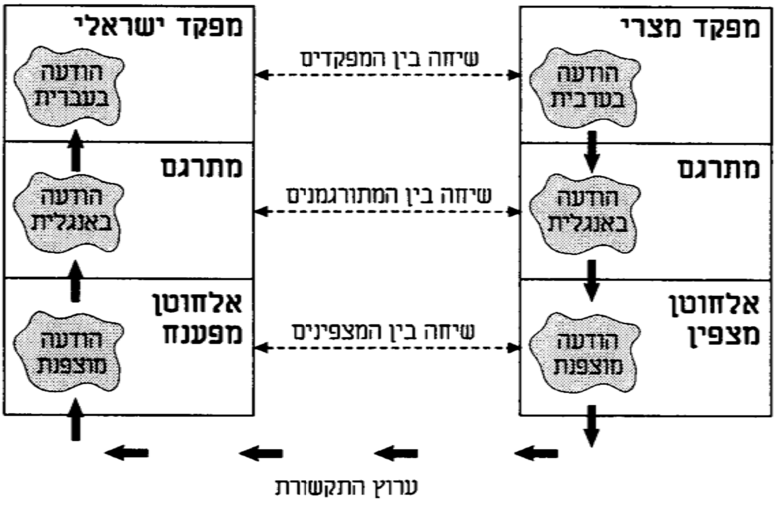
מושג חשוב נוסף הקשור לפרוטוקול הוא המִפתח. מערכות הפעלה מודרניות תומכות בריבוי משימות (multi tasking), כלומר הן מאפשרות למשתמש להפעיל בו-זמנית כמה יישומי תקשורת (וגם יישומים שונים). כדי לנתב את המידע אל היישום המתאים, נקבע

מִפתח (**port**) שדרכו מועבר המידע ליישום הנכון. למשל, שרתי Web (הפועלים בפרוטוקול HTTP) משתמשים במִפתח 80. בסעיף הבא נרחיב אל המושג מִפתח.

כיצד מועברות הודעות ברשת תקשורת

עד כה תארנו מרכיבי יישום במודל שרת לקוח, אבל עדיין לא הסברנו כיצד פועלת רשת תקשורת וכיצד ההודעות מועברות בה. מערכות תקשורת הן מורכבות ביותר. כדי להפחית את הסיבוך הקיים בעיצוב, בבנייה ובתחזוקה של מערכות תקשורת, נהוג לחלקן לשכבות (layers). כל שכבה ממלאת תפקיד מסוים במערכת התקשורת ומספקת שירותי תקשורת מסוימים לשכבות שמעליה. שכבה n במחשב אחד מנהלת שיחה עם שכבה n במחשב אחר. הכללים והמוסכמות המאפשרים לנהל שיחה כזו נקראים פרוטוקול.

נציג להלן דוגמה להמחשת המושגים 'פרוטוקול' ו'שכבה'. נניח ששני מפקדי כוח רב-לאומי שהופקד על שמירת השלום במזרח התיכון, מעוניינים לשוחח כדי לתאם את פעולותיהם. לרשות המפקדים עומדים מכשירי קשר, אך עליהם להתגבר על שתי בעיות: ראשית, מפקד אחד הוא מצרי שאינו דובר עברית, והשני ישראלי שאינו דובר ערבית. שנית, את ההודעות שעוברות בקשר יש להצפין כדי שגורמים עוינים לא יוכלו לצותת להם. איור 1-2 מתאר כיצד אפשר לפתור בעיות אלה בעזרת מתורגמנים ומצפינים.



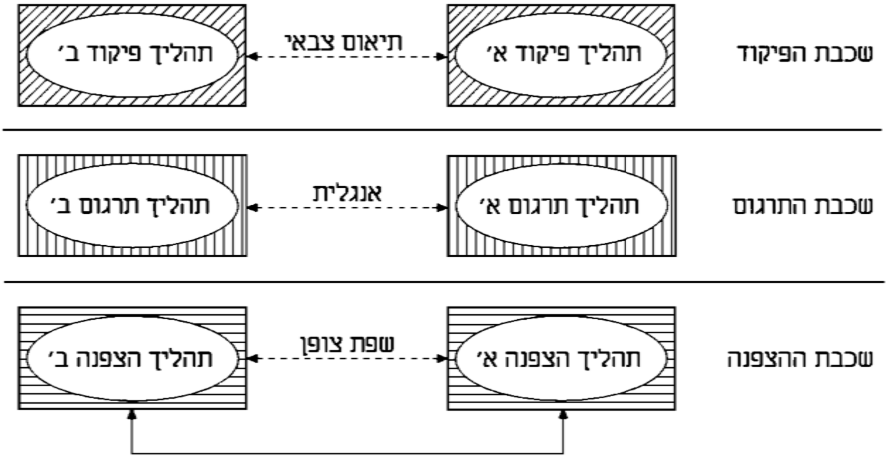
איור 1-2
מערכת תקשורת להעברת הודעות בכוח רב-לאומי

נניח שהמפקד המצרי רוצה להעביר מסר למפקד הישראלי. לשם כך הוא מעביר הודעה בערבית למתרגם שלו, המתרגם מתרגם את ההודעה מערבית לאנגלית ומעביר אותה לאלחוטן המצפין. האלחוטן מצפין את ההודעה ומשדר אותה. ההודעה נקלטת על-ידי האלחוטן הישראלי שמפענח אותה. לאחר הפענוח מתקבלת הודעה באנגלית שמועברת למתרגם הישראלי. הוא מתרגם את ההודעה מאנגלית לעברית, ומוסר אותה למפקד הישראלי.

במערכת תקשורת זו יש שלושה סוגי תהליכים: תהליכים פיקודיים, תהליכי תרגום ותהליכי הצפנה. לכן נוח לחשוב על מערכת זו כעל מערכת המורכבת משלוש שכבות, כפי שמתואר באיור 1-3. באיור מתוארות השכבות האלה: שכבת הפיקוד הכוללת את שני המפקדים, שכבת התרגום הכוללת את המתורגמנים ושכבת ההצפנה הכוללת את האלחוטנים המצפינים. השיחה בין הגורמים שבכל שכבה נעשית לפי פרוטוקול מתאים:

- המפקדים משוחחים בז׳רגון צבאי; הפרוטוקול שלהם כולל מונחים מתחום זה.
- המתורגמנים משוחחים ביניהם באנגלית, לכן הפרוטוקול המוסכם עליהם הוא השפה האנגלית.
- הפרוטוקול של האלחוטנים הוא שפת הצופן, שבה הם משתמשים להצפנת ההודעות.

כל פרוטוקול גם קובע כיצד מוקם קשר בין הצדדים וכיצד הוא מנותק.



איור 1-3
חלוקת המערכת לשלוש שכבות

מה היתרון בחלוקת המערכת לשכבות?

הבעיה הסבוכה נחלקת לכמה בעיות קטנות יותר, וכל שכבה מתמודדת עם חלק מהן. השכבות הנמוכות פוטרות את השכבות הגבוהות מלעסוק בפרטים שכבר טופלו. נתבונן בשכבת הפיקוד בדוגמה שלפנינו.

שני המפקדים אינם צריכים לדעת דבר על תהליכי התרגום וההצפנה. רק חשוב להם שההודעות יועברו באמינות (כלומר ללא שיבושים או השמטות). הם אינם צריכים להכיר את פרוטוקול שכבת התרגום (כלומר את השפה המשותפת למתורגמנים) או את פרוטוקול שכבת ההצפנה (שפת הצופן). אם שני המתורגמנים יחליטו להחליף את השפה האנגלית בשפה אחרת, למשל צרפתית, הם לא יצטרכו להודיע על כך למפקדים. השירות שהמפקדים מקבלים לא ישתנה כלל (כמובן בהנחה שהמתורגמנים אכן יודעים צרפתית). באופן דומה, פרוטוקול ההצפנה לא צריך להיות ידוע למתורגמנים או למפקדים, והחלפת הפרוטוקול לא תפריע לפעולתם.

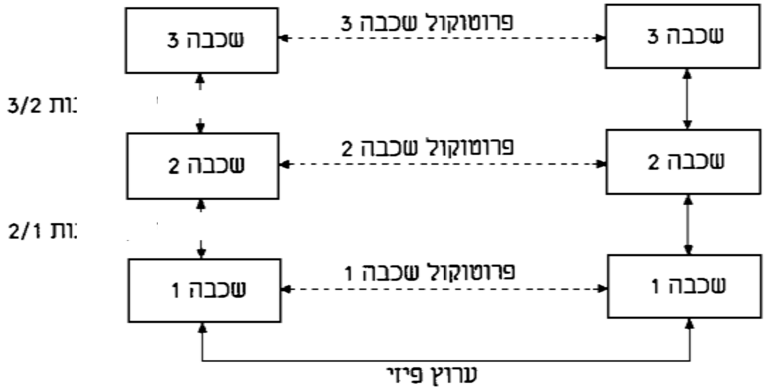
במערכת המחולקת לשכבות, המידע לא מועבר ישירות בין תהליכים עמיתים (כלומר תהליכים באותה שכבה), אלא זורם אנכית: בצד השולח המידע זורם כלפי מטה, ממפקד למתרגם וממתרגם לאלחוטן; בצד המקבל המידע זורם כלפי מעלה, מהאלחוטן למתרגם ומהמתרגם למפקד. רק בין האלחוטנים קיימת העברה פיזית של מידע בערוץ הפיזי (האופקי) המקשר ביניהם.

עם זה, כל מפקד יכול להניח שהוא משוחח ישירות עם עמיתו, שהרי דבריו מכוונים אל המפקד העמית ולא אל המתרגם: הוא פונה אל המפקד העמית ומצפה לקבל את תגובתו. לכן נוח לחשוב כאילו קיים ערוץ אופקי המקשר ישירות בין המפקדים העמיתים. ערוץ זה לא קיים במציאות, אך המפקדים יכולים **לדמות** כאילו הוא קיים. לכן הוא נקרא **ערוץ מדומה** (**virtual channel**). באופן דומה, שני המתורגמנים משוחחים זה עם זה באנגלית. באופן פיזי, המידע מועבר גם במקרה זה, באמצעות האלחוטנים, אך למרות זאת, המתורגמנים יכולים לדמות שהם משוחחים ביניהם ישירות באמצעות ערוץ מדומה.

כל שכבה מוסיפה להודעה מידע בקרה, המאפשר תיאום בין תהליכים עמיתים של אותה השכבה. מידע הבקרה אינו מועבר כלפי מעלה לשכבה שמעל. לדוגמה: נניח שהאלחוטנים צריכים לשנות מדי פעם את פרוטוקול ההצפנה שלהם, כדי להקשות את חשיפתו. כאשר האלחוטן הישראלי ייזום החלפה של פרוטוקול הצפנה, הוא יוסיף מידע מתאים להודעה

שבה הוא מטפל, למשל, את שם הצופן שאליו יש לעבור. מידע זה מיועד לאלחוטן המצרי והוא לא יועבר על-ידו כלפי מעלה, למתרגם המצרי.

האיור הבא מביא תמונה מופשטת יותר של האיורים הקודמים; מוצגים בו שני המושגים ***שכבה*** ו**פרוטוקול** שהצגנו לעיל.



איור 4-1
מודל שכבות למימוש רשת אינטרנט

תחילה נזכיר כי האינטרנט מחבר קבוצה גדולה של רשתות פרטיות או ציבוריות לרשת עולמית אחת. כל אחת מהרשתות המרכיבות את האינטרנט מתוכננת תחת אילוצים וצרכים נתונים ולכן עשויה להיות שונה מהרשתות האחרות מבחינת תקני החומרה והתוכנה. התוכנה של האינטרנט צריכה לאפשר להעביר הודעות בין הרשתות הללו גם כאשר הן אינן תואמות. פתרון לבעיה זו הוא השימוש בפרוטוקול. במקום לנסות לכפות שפה אחידה, אפשר לאמץ נוהל תקשורת שיאפשר להעביר הודעות, בלי להתייחס לתוכנן. למשל, דובר צרפתית שגר בסנגל יכול לשלוח מכתב לידידו בפריז. במכתב יכולים לטפל אנשים שאינם דוברי צרפתית, בתנאי שיש מוסכמה בין-לאומית אחידה של כתובות. לכן אין צורך שכל בני האדם ידברו אותה שפה. בשביל לאפשר לאנשים להחליף הודעות, דרושות מוסכמות בנוגע לאופן העברת ההודעות ולמבנה הכתובות.

לפתרון הזה בדיוק הגיעו מתכנני האינטרנט, וליתר דיוק, מתכנני פרוטוקול האינטרנט שנקרא פרוטוקול **IP** (**Internet Protocol**). פרוטוקול זה הוא ה'דבק' שמחבר את הרשתות השונות לכלל רשת עולמית אחת. כחלק מפרוטוקול זה מוגדר כיצד קובעים כתובות

למחשבים ברשת. לכל חיבור של מחשב לרשת יש כתובת ייחודית שנקראת **כתובת IP** או **IP address**, עליהן נרחיב בסעיף הבא. יחידות הנתונים, או המנות של הפרוטוקול, נקראות **מברקי-IP** (**IP-datagram**). הפתרון של IP לבעיית האחידות הוא פשוט: במקום לנסות לאכוף מבנה מנות אחיד לכל סוגי הרשתות באינטרנט, IP מכניס את המנות לתוך מברקי IP, מוסיף כתובת IP ושולח את המברקים ליעדם. הדבר דומה לשירות מברקים בדואר – מערכת הדואר אינה מתעניינת בתוכן המברקים, אלא בכתובת בלבד.

כל רשת שמחוברת לאינטרנט צריכה להכיר את פרוטוקול IP ולהיות מותאמת למוסכמות של כתובות ושל שמות המקובלים באינטרנט. אולם פרט לאילוצים אלו המנהל של כל רשת יכול להריץ את הרשת שלו לפי שיקוליו באופן חופשי.

פרוטוקול חשוב נוסף שנמצא בשימוש באינטרנט הוא פרוטוקול שכבת התובלה שנקרא **TCP** (פרוטוקול בקרת התעבורה **Transmission Control Protocol**). תפקידו להכין מנות לאפליקציה מסוימת במחשב והוא לשם כך משתמש במפתח.

פרוטוקול TCP נוצר כדי להתגבר על בעיות אלו. פרוטוקול TCP מבטיח אמינות. הוא מבטיח שכל מנה שנשלחה תגיע ליעדה. TCP בודק את המנות שמתקבלות ומסדר אותן על-פי סדר שליחתן. כמו-כן, בגלל שידורים חוזרים של מנות, ייתכן שיגיעו ליעד עותקים כפולים של אותה מנה. TCP בודק גם זאת, ומוודא שכל מנה תתקבל ביעד בדיוק פעם אחת. המחשבים באינטרנט כוללים גם תוכנת TCP והם משתמשים בה כדי להבטיח אמינות.

נסכם אפוא: הפרוטוקולים TCP ו-IP פועלים יחד כדי לאפשר העברת אמינה של נתונים באינטרנט. IP מאפשר לשלוח מנות, או מברקי-IP מהמקור ליעד, אך אינו מבטיח אמינות. TCP מטפל בבעיות האמינות שאינן מטופלות על-ידי IP. מתייחסים לשני פרוטוקולים אלו בשם הכולל: **TCP/IP**.

1.4 כתובות באינטרנט

כאשר שני מחשבים, שמורצים בהם יישומי אינטרנט, מתקשרים זה עם זה, הם שולחים הודעות זה לזה. שולח ההודעה נקרא **מקור** ומקבל ההודעה נקרא **יעד**. כדי שההודעה תגיע

אל היעד, על המקור לציין את כתובת היעד. כתובת היעד משמשת לזיהוי של מחשב היעד. הדבר דומה לאדם ששולח מכתב אל רעהו. הדוור מזהה את הנמען של מכתב כלשהו באמצעות הכתובת הרשומה על המעטפה. זיהוי מחשב היעד נעשה באמצעות שני מרכיבים:

- הכתובת של מחשב היעד;
- מזהה של היישום במחשב היעד – מרכיב זה דרוש משום שבכל מחשב יכולים לרוץ בו-זמנית כמה יישומים. נסביר זאת בהמשך.

לכל מחשב שמחובר לאינטרנט יש כתובת ייחודית המאפשרת לזהות אותו באופן ייחודי, כפי שמספר זהות מאפשר לזהות אזרח באופן ייחודי. ישנן שתי דרכים לייצג כתובת כזו. הדרך הראשונה, היא **כתובת IP** (**IP address**). כתובת IP היא מספר, שלפי התקן הנוכחי מורכב מ-32 סיביות.[3] נהוג לייצג כתובות IP באמצעות ארבעה בתים – כל בית מורכב מ-8 סיביות ומיוצג על-ידי מספר עשרוני בתחום 0 - 255. מייצגים את הכתובת על-ידי רישום ארבעת הבתים, מופרדים על-ידי נקודה. הנה למשל רישום של כמה כתובות IP: 126.17.54.110, 218.60.104.117, 127.0.0.1. כתובת IP משמשת לזיהוי מחשב היעד.

שאלה 1.3

א. האם המספר 23.400.112.35 יכול לייצג כתובת IP? נמקו את תשובתכם.

ב. מהו המספר המרבי של מחשבים שאפשר לחבר לרשת ולזהות באופן ייחודי באמצעות כתובות IP? נמקו את תשובתכם.

ג. נניח שרוצים לאפשר לכל אדם בתבל להתחבר לרשת ישירות ממחשבו האישי או מהטלפון הנייד שלו, באמצעות כתובת IP קבועה שתוקצה בעבורו. האם הדבר אפשרי? הניחו כי מספר כתובות IP שניתן ליצור עם 32 סיביות הוא 2^{32}?

שאלה 1.4

מצאו מהי כתובת ה-IP של מחשבכם בשעה שהוא מחובר לאינטרנט. לשם כך בצעו את הפעולות האלה:

א. לחצו על הכפתור **התחל** (**Start**) ואז על **הפעל...** (**Run...**);

---

[3] סיבית (Binary Digit או בקיצור bit) היא יחידת הנתונים הקטנה ביותר במחשב המיוצגת בבסיס 2. סיבית אחת יכולה להיות בעלת ערך שהוא 0 או 1.

ב. בתיבת הטקסט הקלידו את הפקודה **cmd** ואשרו באמצעות הכפתור **OK**;
ג. בחלון שייפתח הקלידו את הפקודה: **ipconfig** ;
ד. העתיקו מן הרשימה שתתקבל את כתובת ה-IP (IP address).

מערכות מחשב עושות שימוש בכתובות המורכבות ממספרים, אולם לנו, בני-האדם, נוח יותר להשתמש בכתובות המורכבות משמות בעלי משמעות. תארו לעצמכם שהיינו משתמשים במספרי זיהוי כדי לפנות לאנשים ("שלום, 47889520, אני 66521937, מה נשמע?") – אין ספק שזה מוזר ולא נוח. מסיבה זו האינטרנט מאפשר לזהות מחשבים גם באמצעות שמות שנקראים **שמות תחומים** **(domain names)**. שמות תחומים מאפשרים לזהות מחשבים באופן ייחודי. לכל שם תחום יש כתובת IP ייחודית. שם תחום מורכב מכמה מילים המופרדות זו מזו על-ידי נקודה. הנה למשל שם תחום: www.police.gov.il. משתמש שרוצה להתקשר לשרת ווב, עושה זאת על-ידי כתיבת שם התחום (למשל, הקלדת שם התחום בשדה הכתובת של הדפדפן). לסיכום, המשתמש יכול לגלוש לאתר על-ידי שימוש בשם התחום שלו או בכתובת ה-IP שלו.

שאלה 1.5

א. השתמשו בדפדפן כדי לגלוש לאתרים בדרכים האלה:

1) רשמו בתיבת הכתובת את שם התחום הזה: www.police.gov.il.
מהו שם האתר שהגעתם אליו?

2) רשמו בתיבת הכתובת את כתובת ה-IP הזאת: 147.237.72.66. מהו שם האתר שהגעתם אליו?

ב. כדי למצוא כתובת IP של אתר, שאתם יודעים את שם התחום שלו, השתמשו בפקודה ping. הפקודה ping שולחת מידע לאתר זה, ואז מציגה את פרטי ההתקשרות, כולל את כתובת ה-IP של האתר.

1) לחצו על הכפתור **התחל** **(Start)** ואז על **הפעל...** **(Run...)**;
2) בתיבת הטקסט הקלידו את הפקודה **cmd** ואשרו;
3) בחלון שייפתח הקלידו את הפקודה:

ping   www.police.gov.il

4) רשמו את כתובת ה-IP של האתר.

כמו כתובת דואר רגילה (פיזית) שכוללת את שם הארץ, את שם היישוב, את שם הרחוב וכו', כך גם שמות התחומים בנויים מכמה חלקים, כמפורט להלן:

שם הארץ; שם המגזר; שם הארגון; שם המחשב

למשל, כדי לגלוש לאתר של משטרת ישראל נשתמש בשם התחום www.police.gov.il. שם התחום הזה מציין מחשב שנקרא www, השייך לארגון ששמו police (ובעברית משטרה), השייך לממשלה (gov הוא קיצור של goverment), במדינת ישראל (il מייצג את מדינת ישראל).

כפי שאפשר לראות, שמות התחומים מסודרים במבנה הגיוני שמקל את זכירתם. קיימות מוסכמות לגבי שמות אלה. מקובל ששם מחשב (שאינו נמצא בארצות-הברית) יסתיים בשם הארץ שבה הוא נמצא; לכל ארץ נקבעו שתי אותיות המזהות אותה. המרכיב השני מימין מייצג את המגזר של הארגון (למשל gov לעיל). להלן כמה קיצורים מקובלים למגזרים:

ac – אקדמיה, gov – ממשלה, mil – צבא, co – ארגון עסקי, org – ארגון לא עסקי, k12 – בתי-ספר. שמות של אוניברסיטאות בארצות-הברית מסתיימים ב-edu ושל חברות עסקיות ב-com (למשל, השם של חנות ספרים ידועה הוא www.amazon.com).

כאשר אדם או ארגון רוצים להקים אתר ברשת, עליהם לבחור לעצמם בשם תחום. אם שם התחום הרצוי כבר תפוס, יש לבחור בשם אחר. רישום שמות התחומים נעשה על-ידי ועדה מרכזית שעוסקת בכך כדי להבטיח את ייחודיות השם, כלומר, כדי שלא יהיו שני אתרים בעלי אותו שם תחום.

השימוש בשם www

השם www מופיע לעיתים קרובות כשם של מחשב שמריץ שרת Web. אולם, שרת Web יכול לרוץ גם על-גבי מחשב ששמו אינו מתחיל ב-www. כדי להקל את זכירת שם האתר, רוב הארגונים בוחרים בשם www בעבור המחשב שמריץ את שרת ה-Web שלהם. למעשה, מערכת שמות התחומים של האינטרנט היא גמישה ומאפשרת לתת למחשב יותר משם אחד. כדי להדגים זאת נניח שבאוניברסיטה הפתוחה יש מחשב ששמו tavor.openu.ac.il.

אם רוצים להתקין על מחשב זה שרת Web, אפשר להוסיף למחשב את השם הזה: www.openu.ac.il. כעת, אם משתמש מכניס את השם הזה:

http://www.openu.ac.il

הדפדפן יתקשר לשרת שאליו היה מתקשר כאילו המשתמש היה מכניס את הכתובת:

http://tavor.openu.ac.il

שירות שמות תחומים (domain name system)

שאלה למחשבה

אם מחשבים המקושרים לאינטרנט מזוהים באמצעות כתובות IP ומשתמשי הרשת מזהים מחשבים באמצעות שמות תחומים, כיצד מתרגמים את שם התחום לכתובת IP?

באינטרנט קיים יישום רשת שנקרא **DNS** (שירות שמות תחומים – **Domain Name System**). יישום זה כולל שרתי שמות רבים שתפקידם לתרגם שמות תחומים לכתובות IP. כאשר משתמש מפעיל תוכנת לקוח, כמו דפדפן או תוכנת דוא״ל, הוא כותב את שם התחום של מחשב היעד, ותוכנת הלקוח פונה לשרת ה-DNS ומקבלת ממנו את כתובת ה-IP המתאימה. תפקיד שרתי ה-DNS להיות מעודכנים באופן שוטף משום שהאינטרנט הוא רשת דינמית, ובכל רגע עשויים להתווסף לאינטרנט מחשבים נוספים, ובעקבות כך – שמות תחומים חדשים.

מִפתחים (ports)

כבר ציינו שמערכות הפעלה תומכות **בריבוי משימות** (**multi tasking**), ואמנם, משתמשים נוהגים להפעיל כמה יישומים במקביל, כגון דפדפן, מעבד תמלילים, תוכנת דוא״ל ועוד. בעת שכמה יישומי רשת פועלים במקביל, מערכת ההפעלה של המחשב צריכה לנתב את המידע המגיע אל המחשב באופן נכון אל כל אחד מהיישומים האלה.

מאחר שלכל יישומי הרשת המורצים באותו מחשב יש אותה כתובת IP (או אותו שם תחום), קובעים לכל יישום כזה מספר נוסף שנקרא **מִפתח (port)**. המִפתח מאפשר למערכת ההפעלה לנתב את המידע המגיע מהאינטרנט אל היישום המתאים. הדבר דומה לקו טלפון אחד שמקושרות אליו מספר שלוחות/מנויים. כדי להתקשר למנוי מסוים, יש לחייג את מספר הטלפון (כתובת ה-IP) ואת מספר השלוחה (מספר מִפתח). אסור שבאותו מחשב יפעלו שני יישומי תקשורת בעלי אותו מִפתח. ליישומים ידועים יש מספרי מִפתח ידועים (well-known ports). למשל, שרתי Web (פרוטוקול HTTP) מאזינים למִפתח 80; שרתי דוא"ל (פרוטוקול SMTP) מאזינים למִפתח 25. רשימת המִפתחים של כל השירותים התקניים באינטרנט מצויה באתר – http://www.iana.org.

בדרך-כלל משתמשים ביישומים שמספקים שירותים תקניים אינם צריכים לדעת מהו מִפתח של היישום. תוכנת הלקוח היא שיודעת מהו מספר המִפתח של השרת, משום שלשירותי הרשת התקניים יש מספרי מִפתח ידועים. כל מְפַתח דפדפן (או לקוח Web אחר) יודע שעליו ליצור קשר עם השרת במִפתח מספר 80. באופן דומה, כל מְפַתח של יישום שרת דוא"ל, המממש את פרוטוקול SMTP, יודע שעליו להשתמש במִפתח 25.

כאשר כותבים יישום פרטי יש לקבוע מספרי מִפתח בעבור השרת ובעבור הלקוח. נקבע שמִפתחים 0 עד 1023 שמורים לשירותים התקניים של האינטרנט; לכן יישומי תקשורת פרטיים צריכים להשתמש במִפתחים גדולים מ- 1023. לרוב, לשרת וללקוח אין אותו מספר מִפתח. כדי ליזום קשר, על הלקוח לדעת את כתובת ה-IP ואת המִפתח של השרת. כאשר מדובר בשירות תקני, אפשר לברר בקלות מהו המפתח; כאשר מדובר ביישום פרטי, השרת צריך לפרסם את המפתח שהוא מאזין לו, כך שיהיה אפשר לכתוב תוכנות לקוח שייצרו קשר עם מפתח זה. מספר המפתח של הלקוח יכול להיות מוקצה אוטומטית על-ידי התוכנה. הלקוח מעביר לשרת את כתובת ה-IP שלו ואת המפתח שלו כחלק מן הבקשה ליצירת הקשר המזוהה על-ידי:

<server-IP, server port; client-IP, client port>

כלומר, כתובת ומספר מפתח של השרת, כתובת ומספר מפתח של הלקוח. אחרי שהקשר נוצר, הוא נהיה דו-סטרי. הלקוח משתמש בכתובת ובמִפתח של השרת כדי לשלוח אליו הודעות בקשה, והשרת משתמש בכתובת ובמִפתח של הלקוח כדי לשלוח אליו הודעות תגובה.

אחת הבעיות החמורות באינטרנט קשורה לאבטחת נתונים. לא רק אנחנו יכולים להתקשר באמצעות המחשב שלנו למחשבים אחרים כדי לבקש מידע וכדי לקבל מידע, אלא גם מחשבים אחרים יכולים להתקשר למחשב שלנו. כך המחשב שלנו נחשף לחדירת וירוסים, תולעים, גניבת מידע וכו׳. מערכת הפעלה, Windows XP לדוגמה, מכילה **חומת אש** (**Firewall**) המגנה על המחשב מתוכנות מזיקות חיצוניות המנסות לחדור לרשת. חומת האש מונעת הוצאה של מידע מהמחשב ללא אישור המשתמש, כדי למנוע מרוגלות (תוכנות ריגול מסוג spyware), למשל, לשלוח מידע מתוך המחשב אל האינטרנט. חומת האש עושה זאת על-ידי ניטור של כל המְפתחים במחשב. אם היא מגלה שהתקבלה בקשה להוצאת מידע מהמחשב, היא מבקשת את אישור המשתמש לביצוע פעולה זו. בלי אישור המשתמש, לא יצא מידע מהמחשב.

1.5 שימוש במְאַתר משאבים אחיד לזיהוי משאב

למדנו כי מזהים מחשבים ברשת באמצעות כתובות IP, אולם עדיין לא למדנו כיצד מזהים דף web מסוים בשרת, שהרי בכל שרת יכולים להישמר דפים רבים. כדי לאחזר משאבים כמו דפי ווב (שאותם ניתן לכנות גם מסמכים), קובצי קול או קובצי תמונה, יש לזהות את מיקום המשאב במחשב שבו הוא מאוחסן. למשל, כאשר משתמש מפעיל את הדפדפן, עליו לציין מהו הדף ההתחלתי שאותו הוא רוצה לראות ומהו הנתיב אל הדף.

מתכנני האינטרנט הגדירו דרך לזיהוי ייחודי של משאבים: מבנה שנקרא **מְאַתר משאבים אחיד** – **URL** (**Unified Resource Locator**). בהינתן URL, הדפדפן יכול להגיע מיד לדף המבוקש בלי שיהיה צורך לעבור דרך מסמכים אחרים.

URL הוא מחרוזת תווים המורכבת מחמשת החלקים האלה:

- הפרוטוקול שבו משתמש השרת
- שם המחשב או כתובת IP
- מספר המְפתח
- שם הקובץ (כולל נתיב התיקיות שבו נמצא הקובץ)
- נקודת הייחוס (reference) בתוך הקובץ

דוגמה ל- URL שכולל את כל חמשת החלקים:

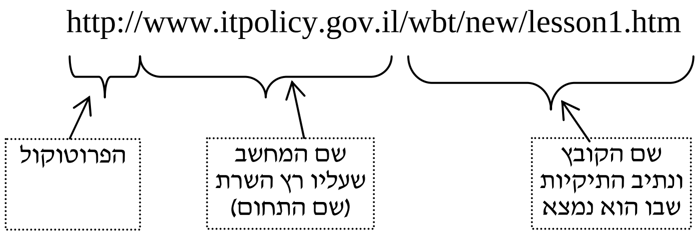
http://www.openu.ac.il:80/main/subjects/education.html#java

נסביר את מרכיבי כתובת ה- URL :

- http הוא הפרוטוקול שבו משתמש השרת .
- www.openu.ac.il הוא שם המחשב .
- 80 הוא מספר המִפתח.
- main/subjects/education.html הוא שם הקובץ (כולל נתיב התיקיות שבו הוא נמצא).
- #java היא נקודת הייחוס (reference) בתוך הקובץ.

הערה: אחרי החלק שמציין את שם השרת מופיע המספר 80. המספר הזה מציין את מספר המפתח. כאשר מדובר בפרוטוקול HTTP, מספר זה הוא מיותר שכן הדפדפן יודע שזהו המִפתח של שרתי HTTP.

לא כל החלקים הללו חייבים להופיע; המִפתח והייחוס מופיעים רק לעיתים רחוקות. הנה דוגמה ל-URL :



במקרה זה ה-URL מורכב משלושה חלקים. אלה החלקים העיקריים שיש בכל URL :

1. הקידומת //:http, מציינת את הפרוטוקול המזהה את השרת. מחשב יכול להריץ כמה שרתים ולכן יש לציין לאיזה שרת יש לפנות. במקרה זה מדובר בשרת Web שהוא שרת http. כאשר רוצים לפנות לשרת אחר, למשל לשרת ftp, יש לרשום את הקידומת //:ftp.
2. החלק השני, www.itpolicy.gov.il, הוא כתובת של מחשב ברשת. במקרה זה מדובר במחשב המשמש כשרת ה-Web של אתר טכנולוגיות המידע הממשלתי[4] (שבאנגלית שמו itpolicy).

---

[4] אתר זה כולל הדרכה על נושאים הקשורים לאינטרנט.

3. החלק השלישי של ה-URL, מזהה דף על-ידי ציון הנתיב (path) ושם הקובץ. במקרה זה, הנתיב (ביחס לתיקיית השורש) הוא: /wbt/new, ושם הקובץ: lesson1.htm.

מאחר שפרוטוקול HTTP הוא הפרוטוקול של ה-Web, הרי שבמרבית המקרים הדפדפן יפנה לשרת HTTP. לפיכך, הקידומת //:http היא נפוצה ביותר ומהווה ברירת מחדל; כאשר לא מופיעה קידומת ב-URL, הדפדפן מוסיף בעצמו את הקידומת //:http.

במחרוזות רבות של URL אין מציינים את החלק המזהה את דף הבית של השרת. לקובץ של דף הבית יש שם קבוע, לרוב index.html או index.htm. לפיכך, כדי לאחזר את דף הבית של אתר טכנולוגיות המידע הממשלתי, מספיק להקליד את ה-URL הזה:

www.itpolicy.gov.il/

1.6 תהליך התקשרות שרת-לקוח בפרוטוקול HTTP

התקשורת בין השרת ללקוח ב-HTTP נעשית באמצעות בקשות ששולח הלקוח ותגובות שמחזיר השרת. ראשית, הלקוח יוצר חיבור קשר באמצעות כתובת ה-IP וציון המִפתח שבו השרת נמצא, בדרך-כלל מִפתח 80. לאחר מכן נשלחת הבקשה, הכוללת את הכתובת של האובייקט המבוקש (למשל, דף HTML) ופרטים נוספים על הבקשה ועל הלקוח. השרת קורא את הבקשה, מפענח אותה, שולח ללקוח תשובה ולרוב מנתק את הקשר עם הלקוח לאחר שליחת התשובה.

פרוטוקול HTTP מגדיר את המבנה של הבקשות ושל התגובות, ומבנה זה מחייב את יישומי הלקוח והשרת. המשתמשים בדפדפן לא צריכים להיות מוטרדים מכך, משום שהדפדפן מעביר בעבורם את הנתונים בצורה תקינה. מי שצריכים לתת דעתם על כך הם הכותבים של יישומי הרשת. הכותבים צריכים להבין את הפרוטוקול, להכיר את המבנה של ההודעות ואת הפעולות שהפרוטוקול מאפשר לבצע.

תהליך ההתקשרות כולל מספר פעולות:

1. הלקוח (תוכנת הדפדפן) יוצר קשר עם השרת. לשם כך הוא מפרק את ה-URL למרכיביו ומחלץ את שם התחום של מחשב היעד; הדפדפן מתקשר עם שרת השמות כדי לתרגם את שם המחשב המרוחק לכתובת IP ולמספר מִפתח.
2. הדפדפן שולח בקשת HTTP (HTTP Request), ומכניס את נתיב המסמך להודעת הבקשה.
3. השרת מקבל את הודעת הבקשה, מאתר את הקובץ המבוקש במערכת הקבצים של המחשב שבו הוא רץ, מכניס את הקובץ לתוך הודעת תגובת HTTP (HTTP Response) ושולח את הודעת התגובה ללקוח.
4. השרת מנתק את הקשר עם הלקוח.
5. הדפדפן קורא את הקובץ, מעצב את הטקסט לפי ההנחיות הכלולות בקובץ ה-HTML, ומציג אותו בחלון שלו.

שיחה בין לקוח לשרת מתחילה בשליחת הודעת בקשה מאת הלקוח (הנקראת HTTP Request) ומסתיימת בשליחת הודעת תגובה מהשרת (הנקראת HTTP Response). שיחה זה נקראת מעבר **הלוך-ושוב** (**Round Trip**), והיא חייבת לכלול לפחות מעבר הלוך-ושוב אחד. בקשת דף המכיל תמונות או קובצי קול תגרום לקיום מעבר הלוך-ושוב גם בעבור הדף עצמו וגם בעבור כל קובץ (קול או תמונה) בנפרד, וזאת לפני ניתוק הקשר בין השרת ללקוח.

שרת אשר קיבל בקשה מלקוח יספק ללקוח תגובה הכוללת את הדף המבוקש. אם לא צוין שם המשאב (הקובץ), יישלח דף ברירת המחדל של האתר. אם הדף המבוקש לא נמצא, תישלח הודעת שגיאה.

תכנות מצב-חסר (Stateless)

שרת ה- Web אינו שומר כל מידע על התקשורת בינו ובין הלקוח (אלא אם מגדירים זאת בדרכים מסוימות שחלקן נציג בהמשך). בכל בקשה המגיעה מלקוח (דפדפן) פותח השרת קשר אל הלקוח ששלח את הבקשה ובונה בעבור אותו לקוח דף HTML מבוקש. בסיום הטיפול בבקשה, סוגר השרת את הקשר עם הלקוח ומתפנה לטפל בבקשות של לקוחות אחרים. אם הדפדפן רוצה לבקש בקשה נוספת מהשרת, עליו להפעיל תהליך זה מחדש, שכן השרת לא שמר כל מידע על השיחה האחרונה, ולפיכך טיפול בבקשה חדשה של אותו לקוח

מתבצע ללא כל תלות בבקשה הקודמת. שיטה זו, שבה השרת אינו שומר מידע על ההתקשרות, מאפשרת לו לטפל באלפי בקשות ביעילות רבה יותר.

צורת ההתקשרות הזאת בין שרת ללקוח נקראת **מצב-חסר** (**Stateless**) ולתכנות בסביבה זו קוראים **תכנות-מצב-חסר** (**Stateless Programming**). אולם, צורת ההתקשרות הזאת היא לעיתים בעייתית. מאחר שמידע לא נשמר בין התקשרות להתקשרות, המשתמש שגלש לאותו אתר צריך להזדהות בכל פעם מחדש. עם זאת, ישנן כמה שיטות שמאפשרות לפתור את הבעיה. בהמשך נכיר שיטה שמאפשרת לשמור נתונים שנאספו במהלך ההתקשרות לטווח קצר (פרק 3) ונלמד להשתמש במסד נתונים שמאפשר לשמור נתונים שנאספו במהלך ההתקשרות לטווח ארוך (פרק 4).

1.7 יישום שרת-לקוח בטכנולוגיית ASP

יישום שרת-לקוח, כמו כל יישום אחר, מבצע את הפעולות האלה: קלט, עיבוד ופלט. כאשר מפתחים יישום מסוג זה יש לפתח הן את יישום הלקוח והן את יישום השרת.

יישום הלקוח

יישום הלקוח מטפל כאמור בהצגת מידע למשתמש. לכן הוא כולל ממשק ידידותי המאפשר למשתמש להעביר נתונים ליישום השרת. כפי שציינו, רוב העיבוד מתבצע בשרת אך לעיתים אנו נדרשים לבצע פעולות מסוימות גם ביישום הלקוח. הסיבה לכך היא שהרצה של יישום במודל שרת-לקוח כרוכה בתקשורת ולכן אורכת זמן רב יותר מאשר הרצה של תכנית רגילה במחשב אחד. לכן רצוי להעביר לשרת רק נתונים תקינים וכך לחסוך בזמן. נוכל למנוע העברת נתונים שגויים לשרת אם יישום הלקוח יכלול פונקציות לבדיקת תקינות הנתונים המוזנים על-ידי הלקוח. קיימות שפות מיוחדות שפותחו למטרה זו והן נקראות שפות **תסריט** (**script**). בדרך-כלל משתמשים בשפות מסוג זה למימוש של בדיקות תקינות אצל הלקוח.

יישום השרת

תפקידו של יישום השרת הוא לעבד את הנתונים המגיעים מהלקוח ולשלוח בחזרה מידע לתצוגה באמצעות הדפדפן המריץ את יישום הלקוח. כדי לפתח יישום שרת נשתמש

בטכנולוגיית **דפי שרת פעילים**, ובקיצור **ASP** ‏(**A**ctive **S**erver **P**ages). ASP היא טכנולוגיית צד-שרת, שפותחה על-ידי חברת מיקרוסופט, ומאפשרת להציג תוכן דינמי בדפי אינטרנט. טכנולוגיית ה-ASP מיועדת לטפל בתהליך ההתקשרות שבין השרת ללקוח באמצעות אוסף של עצמים המבצעים פעולות נפוצות כמו קריאת בקשת HTTP והכנת תגובה ללקוח. הפלט של העיבוד שביצע השרת נשלח כדף HTML.

אתר סטטי

כאמור, אתרים בנויים מדפים שונים. חלק מן הדפים הם סטטיים וחלקם דינמיים. דף סטטי הוא דף web השמור בשרת שכל לקוח יכול לבקשו ולקבלו. כדי לשנות את הדף עלינו לערוך אותו, לעדכן את תוכנו ולשמור אותו שוב.

לשיטה זו יש חסרונות. לדוגמה, נניח כי אדם מעוניין לרכוש מחשב דרך אתר מכירות. בהיכנסו לאתר הוא מתבקש לבחור את סוג המעבד, את גודל הזיכרון, את סוג כרטיס המסך ועוד. כעת על השרת באתר המכירות לשלוח לו דף הכולל נתונים העונים לדרישות שצוינו. שיטת עבודה הבנויה על טכנולוגיית HTML בלבד מאפשרת ליצור **אתר סטטי** המכיל דפים קבועים שהוכנו מראש. כלומר, כדי לשלוח מידע המתאים לפרטי המחשב שהמשתמש ביקש, על מפתח האתר לצפות את כל סוגי הבקשות האפשריות ולהכין דפים רבים שכל אחד מהם יכול בעתיד להתאים לבקשה מסוימת של לקוח.

שאלה למחשבה

האם אפשר להעריך כמה דפים סטטיים ישנם באתר מכירות מחשבים?

לסיכום, נציג את החסרונות העיקריים של השימוש בדפים סטטיים:

- יש לתכנן מראש את הדפים שהלקוחות עשויים לבקש; דבר זה דורש לנהל מספר רב של דפים (דף אחד לכל בקשה).
- אם הנתונים המופיעים בדפים הסטטיים משתנים, יש לעדכן את הדפים.
- אפשרות לערוך חיפוש רק לפי קטגוריות שנקבעו מראש.

אתר דינמי

ברור אפוא שיש יתרון לטכנולוגיה שבה אפשר לבנות דפים באופן דינמי, כלומר לבנות דפים על-פי בקשת הלקוח, כאשר הנתונים המוצגים בהם נלקחים מהשרת, אך הדפים עצמם אינם נשמרים בשרת. לדוגמה, באתר מכירות מחשבים נרצה לבנות דף המתאים לדרישות הלקוח, ולשלוח אותו ללקוח, בלי שיהיה דף כזה בשרת.

טכנולוגית ASP מאפשרת ליצור דפי אינטרנט דינמיים, על-פי בקשת הלקוח. כאשר הלקוח מבקש דף כלשהו, לדוגמה דף הכולל מידע על מחשבים ניידים בעלי דיסק בגודל 80G ומעלה, הוא פונה לדף אשר רץ על השרת (דף עם סיומת asp / aspx). בתגובה, השרת יוצר באופן דינמי דף HTML המתאר את כל המחשבים העונים לדרישות הלקוח, ושולח אותו ללקוח. הדף שהלקוח מקבל אינו קיים פיזית בשרת, ואף לא נוצר כאשר התקבלה הבקשה. המידע המבוקש נשלח בצורה דינמית ללקוח בפורמט HTML, ותוכנת הלקוח בונה את הדף ומציגה אותו. את הנתונים שבהם השרת משתמש כאשר הוא בונה דף דינמי ניתן לאחסן בשיטות שונות. מרבית הנתונים נשמרים במסד נתונים, למשל פרטי כל המחשבים שהחנות הווירטואלית מוכרת; נתונים אחרים נשמרים בעצמים מיוחדים שעליהם נלמד בהמשך. בפרקים 3 ו-4 נציג כלים ושיטות לבניית דף דינמי בשרת ולשליחתו ללקוח.

באיור 1-5 המופיע בעמוד הבא מתואר תהליך התקשרות שרת-לקוח ויצירת דף ASP דינמי. להלן שלבי התהליך המתוארים באיור:

שלב 1 – מְפַתח משתמש בעורך כדי ליצור דף ASP.

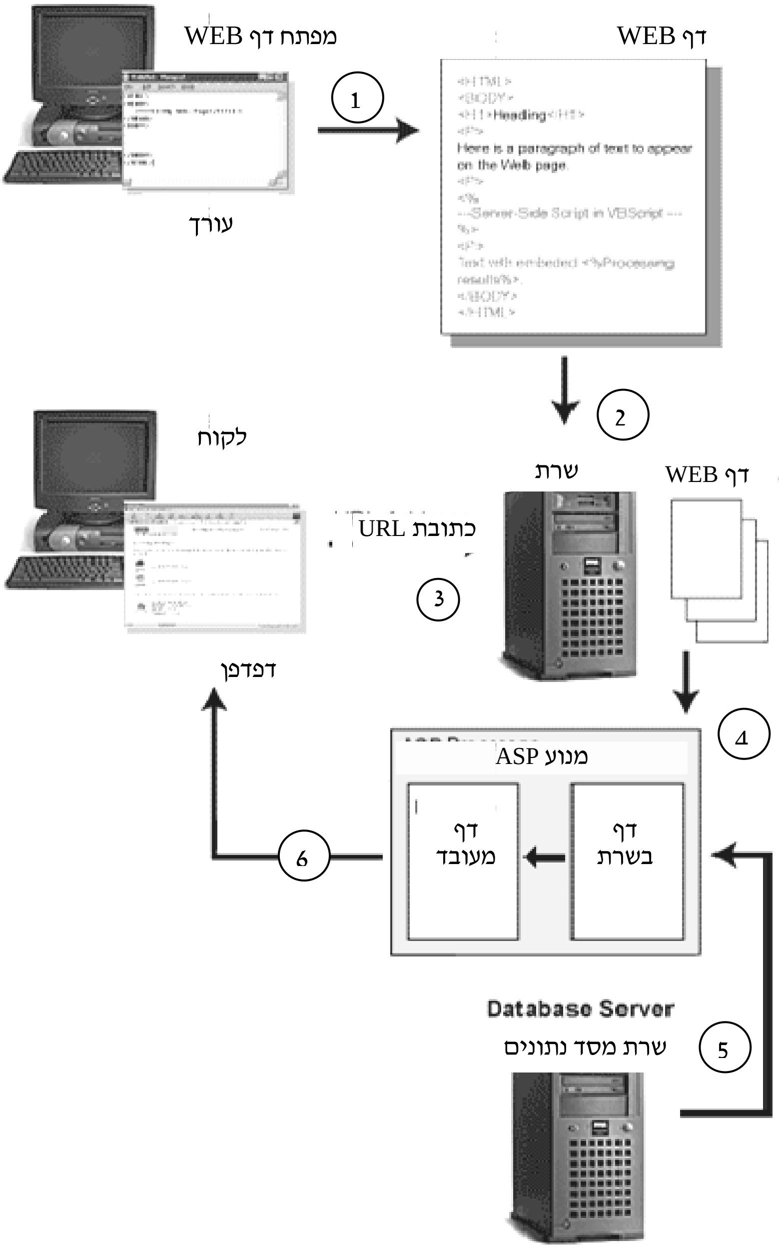
שלב 2 – דף ה-ASP מאוחסן בשרת. השרת ממתין לבקשה של לקוח.

שלב 3 – לקוח שולח בקשה לשרת לקבלת דף.

שלב 4 – השרת משתמש בכתובת URL כדי לאחזר את דף ה-ASP המתאים.

שלב 5 – דף ה-ASP מעבד את הנתונים; במידת הצורך השרת מאחזר נתונים ממסד נתונים. השרת יוצר תגובת HTTP המורכבת מתגיות HTML.

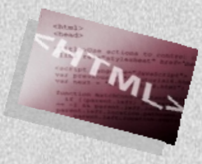
שלב 6 – השרת שולח ללקוח את תגובת ה-HTTP, והלקוח מציג את הדף בעזרת הדפדפן.



איור 5-1
תהליך העברת תוכן דינמי בין לקוח ובין שרת

לסיכום, נציג את יתרונות הטכנולוגיה הדינמית:

- התאמה לצורכי הלקוחות – אפשר ליצור מספר לא מוגבל של דפים המותאמים לצורכי הלקוחות.
- תחזוקה נוחה יותר של נתונים בשרת – קל יותר לתחזק מסדי נתונים שבהם נשמר המידע הנדרש. למשל, קל לעדכן נתונים קיימים ולהוסיף נתונים חדשים. דף HTML שנוצר באופן דינמי ישקף את מצב הנתונים העדכני.
- קיצור זמן הכתיבה של האתר – כותב האתר צריך ליצור אך ורק דפי תצוגה כלליים. המידע עצמו נבנה בצורה דינמית מתוך מסד הנתונים. עדכון המידע נעשה במסד הנתונים.



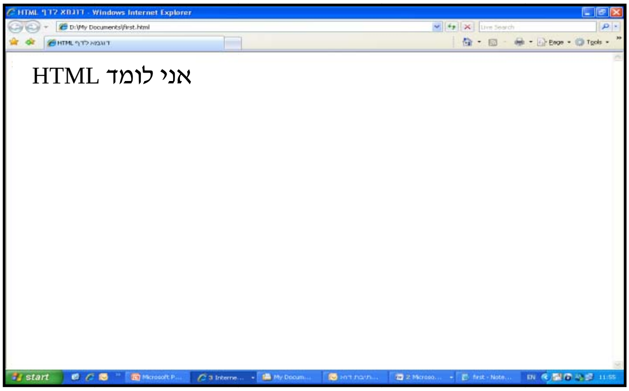
פרק 2

יצירת דפי HTML ו–CSS

2.1 מבוא לשפת HTML

שפת סימון של היפר-טקסט או שפת HTML (hypertext markup language) היא שפת הסימון של המסמכים ב-Web. כיום, יש כלים רבים המשמשים להפקה ולעריכה של מסמכי HTML. גם מעבד התמלילים הנפוץ Word מציע אפשרות לשמירת קובץ כמסמך HTML. לכן, אפשר להפיק מסמכי HTML גם בלי להכיר שפה זו. עם זה, כדי לפתח תוכנה היוצרת מסמכי HTML, יש להבין את המבנה הבסיסי של השפה. לכן נסקור בפרק זה את שפת HTML בקצרה; אין בכוונתנו לתאר את כל האפשרויות של שפה זו, אלא להבהיר את המבנה הכללי שלה ולתאר כמה מהאפשרויות הגלומות בה.

איור 2-1 מציג דף Web פשוט ביותר, כפי שהוא מוצג בתוך חלון של הדפדפן Microsoft Internet Explorer.



איור 2-1
הצגת הודעה בדף Web

דף זה נוצר על-ידי כך שהדפדפן מעבד את קובץ ה-HTML הזה :

```
<html>
<head>
  <title>
            דוגמא לדף HTML
 <title/>
<head/>
<body>
            אני לומד HTML
<body/>
</html>
```

אפשר לראות שמסמך HTML הוא קובץ טקסט המכיל **תגים** (**tags**). התגים הם הנחיות אשר מורות לדפדפן כיצד לעצב את הטקסט. תג מורכב משֵׁם, שנתון בתוך סוגרים מהצורה < > ; למשל, התג <head> מציין את תחילת הכותרת של המסמך. חלק גדול מהתגים תוחמים אזור של טקסט. במקרה כזה משתמשים בתג פותח – שמציין את תחילת אזור הטקסט, ותג סוגר – שמציין את סופו. התג הסוגר דומה לתג הפותח אך מתחיל בלוכסן ; למשל, התג </head> הוא התג הסוגר המתאים לתג הפותח <head>. כל מסמך צריך להתחיל בתג הפותח <html> ולהסתיים בתג הסוגר </html>.

מסמך HTML כולל שני חלקים עיקריים : **כותרת עליונה** (**header**) ולאחריה **גוף** (**body**). כותרת המסמך מכילה מידע על המסמך וגוף המסמך כולל את פריטי המידע שבמסמך. למשל, הכותרת העליונה מכילה בדרך-כלל **כותרת** (**title**) שמתארת את תוכן המסמך ושמוצגת על-ידי הדפדפן בשורת הכותרת של החלון. מטרת הכותרת היא לאפשר למשתמש לדעת מהו הדף הנוכחי שבו הוא צופה. התגים <title> ו-</title> תוחמים את כותרת המסמך. התגים <body> ו-</body> תוחמים את גוף המסמך.

המבנה הכללי של מסמך HTML הוא כדלקמן :

```
<html>
<head>
 <title>טקסט שמרכיב את כותרת המסמך</title>
</head>
```

```
<body>
גוף המסמך
</body>
</html>
```

אפשר לראות שהדוגמה שהבאנו לעיל דומה ביותר למבנה זה.

אפשר לרשום את הטקסט בקובץ HTML בפריסה כלשהי השומרת על סדר התגים; אין חובה לרשום כל תג בשורה נפרדת. עם זה, כדי להקל את קריאת המסמך רצוי לרשום את הקובץ באופן שישקף את הצורה שבה הוא מוצג למשתמש. כך נעשה בדוגמאות בספר זה.

מרכיבי שפת HTML: אלמנטים, תגים ומאפיינים

מסמך HTML הוא קובץ טקסט הבנוי מאלמנטים של HTML. אלמנטים אלו מגדירים את הצורה שבה הדפדפן יציג את הטקסט שבמסמך. כל אלמנט מוגדר על-ידי תגי HTML. לכל תג HTML יש שם והוא מסומן על-ידי התווים < >. לדוגמה:

```
<title>
```

מרבית התגים מופיעים בזוגות: תג פותח ותג סוגר, וביניהם אנו רושמים את תוכן האלמנט. לדוגמה:

```
<title>טקסט שמרכיב את כותרת המסמך</title>
```

התפקיד של התג <title> הוא להגדיר אלמנט שיוצג בשורת הכותרת העליונה בדפדפן.

שימו לב, בשפת HTML אין הבדל בין כתיבה באותיות קטנות לכתיבה באותיות גדולות. בספר זה נשתמש באותיות קטנות.

לחלק מהתגים אנו יכולים להוסיף **מאפיינים** (**properties**) **וערכים** (**values**) המספקים לדפדפן מידע נוסף על אופן הצגת האלמנט. לדוגמה,

```
<div dir="rtl"> אני לומד HTML </div>
```

התג <div>, שנדון בו בהרחבה בהמשך, מגדיר את האלמנט 'מקטע במסמך'; המאפיין **dir** (direction) שקיבל את הערך "rtl" מציין את הערך 'כיוון הטקסט במקטע מימין לשמאל' (**rtl** – **R**ight **T**o **L**eft). בצורה דומה ניתן להגדיר כיוון הטקסט במקטע משמאל לימין (**ltr** – **L**eft **T**o **R**ight). המבנה הכללי של תג בעל מאפיינים הוא:

```
<tag-name  property1 = "value1"  property2 = "value2" ... >
```

כלומר, לאחר שם התג יכולים להופיע, בתוך הסוגריים המשולשים, מאפיינים אחדים. כל מאפיין (property) מורכב משֵם, לאחריו הסימן =, ולאחריו ערך המאפיין. לכל תג יש מאפיינים משלו ולכל מאפיין יש קבוצת ערכים אפשריים. את ערכי המאפיינים יש לרשום בין גרשיים (אף-על-פי שלא כל הדפדפנים מקפידים על כך).

התגים <--   --! > מייצגים הערה . לדוגמה,

```
<!-- זו הערה -->
```

הדפדפן אינו מתחשב במה שכתוב בהערה, ולכן ניתן להשתמש בתגים <--   --! > כדי לתת הערות על הכתוב בקוד המקור. ההערות ניתנות לצפייה על-ידי פתיחת קוד המקור .

2.2 שימוש בתגים לכתיבת קובצי HTML

2.2.1 כותרות

שפת HTML מכילה שישה זוגות של תגים המשמשים לקביעת כותרות במסמך. תג במבנה <h*i*> מציין התחלה של כותרת ברמה *i* ; תג במבנה </h*i*> מסמן את סוף הכותרת. למשל כותרת ברמה הגבוהה ביותר תופיע בין התג <h1> לתג </h1>. כל דפדפן קובע כיצד יוצגו הכותרות, אך ככל שהכותרת היא ברמה גבוהה יותר, כך האותיות שלה יהיו גדולות יותר ומודגשות יותר. שימו לב, ככל שערכו של *i* קטן יותר, רמת הכותרת גדולה יותר. כלומר, התג <h1> מגדיר את הכותרת ברמה הגבוהה ביותר ו- <h6> את הכותרת ברמה הנמוכה ביותר.

דוגמאות :

```
<h1> כותרת עיקרית </h1>
<h2> תת כותרת </h2>
```

שאלה 2.1

בנו דף HTML בשם sol2-1.htm, המורכב מכותרות. בכותרת ברמה הראשונה רשמו "דף הבית של <שמכם>"; בכותרת ברמה השנייה רשמו את שם בית-הספר ואת הכיתה שבה אתם לומדים; בכותרת ברמה השלישית רשמו את התחביבים שלכם.

2.2.2 המאפיין style

style (עיצוב) הוא אחד המאפיינים שמסייעים לנו להפריד בין העיצוב של הדף ובין התוכן שלו. כפי שנלמד בפרק זה, השימוש במאפיין style עוזר לבנות דף HTML קריא ומאפשר לבצע בקלות שינויים סגנוניים.

נניח שברצוננו להציג דף המעוצב באופן הזה: הכותרות מיושרות לימין, רקע הדף הוא בצבע ירוק, הכותרות המסומנות על-ידי התגים <h1> ו-<h4> הן בצבע כחול, הכותרת המסומנת על-ידי התג <h2> היא בצבע אדום. כיצד נעשה זאת? לפני שנתאר את דף ה-HTML נציג את כל אחת מתכונות העיצוב המאפשרות לקבוע את צבע הרקע, את צבע הגופן ואת אופן היישור של הטקסט. להלן התכונות:

א. לקביעת אופן יישור הטקסט נשתמש בתכונה text-align, שיכולה לקבל ערכים כגון right, left, center.

ב. לקביעת צבע הרקע נשתמש בתכונה background-color. שיכולה לקבל ערכים כגון red, blue, black, lime.

ג. לקביעת צבע הגופן נשתמש בתכונה color, שיכולה לקבל ערכים כגון black, blue, red, lime. שיטה אחרת המאפשרת להגדיר את ערכי הצבע היא שיטת **RGB (red green blue)**. בשיטה זו הצבע נקבע על-ידי צירוף של שלושת צבעי היסוד: אדום, ירוק וכחול. לדוגמה, נרשום כחול כצירוף הזה:

```
style= "color:#0000FF"
```

ד. לקביעת סוג הגופן נשתמש בתכונה font-family, שיכולה לקבל ערכים כגון Arial, Sans-serif, Courier.

נשתמש בתכונות אלו כדי לבנות את דף ה-HTML המתאים. להלן גוף הדף:

```
<!-- דוגמה לשימוש במאפיין עיצוב -->
<body style="background-color:lime; text-align:right;
             color:blue; font-family: sans-serif ">
   <h1>דף הבית שלי</h1>
   <h2 style="color:red"> ברוכים הבאים </h2>
   <h4> HTML אני לומד </h4>
</body>
```

שימו לב כי את המאפיינים המגדירים את העיצוב של הדף כולו אנו רושמים בתוך התג הפותח <body>. המאפיין style מכיל תכונות רבות; כל תכונה מוגדרת בצורה הזאת: "ערך: שם". כדי להפריד בין התכונות אנו משתמשים בתו ';' . לדוגמה:

```
<body style="background-color:lime; text-align:right;  color:blue; font-family: sans-serif ">
```

לתגים <h1> ו- <h4> לא הגדרנו מאפייני עיצוב נוספים, ולכן העיצוב שלהם נקבע על-ידי העיצוב שהגדרנו בתג <body>.

לתג <h2> הוספנו מאפיין עיצוב הקובע שצבע הטקסט יהיה אדום.

```
<h2 style="color:red"> ברוכים הבאים </h2>
```

הגדרה זו מתקיימת רק בעבור הצגת האלמנט כותרת <h2>.

הגדרת סגנון זו, שבה הסגנון מוגדר בתוך התג, נקראת inline. בהמשך נכיר הגדרות סגנון נוספות.

שאלה 2.2

תארו מה יוצג בדפדפן אם נרשום את התגים שלפניכם בגוף דף HTML:

```
<body>
   <h1 style="background-color:lime; text-align:right;
               color:blue"> דף הבית שלי </h1>
    <h2 style="color:red"> ברוכים הבאים </h2>
    <h4 style="background-color:lime; text-align:right;
               color:blue">  HTML אני לומד  </h4>
</body>
```

שאלה 2.3

כתבו דף HTML בשם sol2-3.htm, שרקעו צהוב, ובו יוצג שמכם בחמש כותרות בגדלים שונים ובצבעים שונים.

2.2.3 מעברי שורות וחלוקת דף למקטעים

תגים אחדים מציינים פעולות שיש לבצע שצריכות להשפיע באופן מיידי. במקרה כזה התג ממוקם בדף ה-HTML בדיוק במקום שבו יש לבצע את הפעולה. דוגמה לכך הוא התג המורה על מעבר משורה נוכחית לשורה חדשה. אם נרשום בדף HTML את הטקסט הבא ונציג את הדף, נראה כי כל הטקסט יופיע בשורה אחת, למרות שרשמנו את הטקסט בשלוש שורות:

```
<body>
        ברוכים הבאים
        לדף הבית שלי
        אני לומד HTML
</body>
```

הרווחים ומעברי השורות שביצענו לא יוצגו בדפדפן, כי לא השתמשנו בתג המורה על מעבר בין שורות. כדי לעבור לשורת פלט חדשה יש להשתמש בתג המיידי **
** (קיצור של break או של 'שבירת שורה'). שימו לב, כי תג מיידי נכתב כך שבסופו יש רווח ולאחריו התו /.

פסקה מופיעה בין התג הפותח **<p>** (קיצור של paragraph – פסקה) ובין התג הסוגר </p>.

להלן גוף מסמך HTML הכולל תגים של מעברי שורות ושל תגי פסקה.

```
<body>
  <p>
        טקסט מופיע
        ללא שימוש
        בתג להפרדת שורות
  </p>
        <br />יצירת שורה ריקה
  <p>
        <br /> טקסט זה מופיע
        <br />  בפסקה נפרדת
  </p>
</body>
```

לצורך חלוקה לוגית של הדף למקטעים נשתמש בתג הפותח **<div>** (קיצור של division) ובתג הסוגר </div>. החלוקה הלוגית של דף למקטעים מאפשרת להגדיר עיצוב מסוים לכל מקטע. לדוגמה, נכתוב דף HTML שהרקע שלו ורוד והוא מכיל שני מקטעים. למקטע הראשון יש רקע צהוב וצבע גופו כחול ולמקטע השני יש רקע כחול וצבע גופן אדום.

להלן גוף דף HTML הכולל תגי מעבר שורות ותגי חלוקה למקטעים :

```
<body style="text-align:right; background-color: Fuchsia">
<h1 > ברוכים הבאים </h1>
<!-- מקטע ראשון -->
<div style="color:blue; background-color:yellow;
              font-family: sans-serif">
                דף הבית שלי   <br />
<h3>  HTML אני לומד  <br /> </h3>
<br /><br />
</div>
<!-- מקטע שני -->
<div style="color:red; background-color:Aqua">
<p>
   פסקה שנייה  <br />
   תוכן פסקה שנייה  <br />
</p>
</div>
</body>
```

2.2.4 שילוב תמונות במסמך

קובץ HTML הוא קובץ טקסט, ולכן נשאלת השאלה כיצד מסמך Web יכול להכיל תמונות? התשובה היא שתמונות אינן מופיעות ישירות בתוך קובץ HTML, אלא מאוחסנות בקובץ אחר ומשובצות במסמך באמצעות התג המיידי <img /> המגדיר אלמנט מסוג תמונה. לכן, מסמך או דף Web מורכב בדרך-כלל מכמה קבצים. התג img גורם לדפדפן לאחזר את קובץ התמונה ולהציג אותו. התג <img /> גם הוא תג מיידי ועל כן מוסיפים בסיום התג רווח ולאחריו את התו /, כמו בתג <br />.

המאפיין src של התג <img /> מציין היכן מצוי קובץ התמונה. התג שלהלן משבץ במסמך את התמונה המאוחסנת בקובץ face.gif :

```
<img src="face.gif" alt="תמונת פנים" />
```

התמונה תופיע במסמך בדיוק במקום שבו ממוקם התג. המאפיין src מציין את מיקום התמונה והמאפיין alt מגדיר את הטקסט שיופיע במקום התמונה אם לא יהיה אפשר להציגה, או כאשר המשתמש יצביע עם העכבר על התמונה.

המבנה הכללי של התג המיידי <img > הוא:

```
<img src= "url"  alt="טקסט שיופיע במקום התמונה"  />
```

נציין כמה מאפיינים נוספים של התג <img>, שתוכלו להשתמש בהם כדי להשפיע על אופן הצגת התמונה:

- המאפיין alt – מציין את הערך 'טקסט קצר המתאר את התמונה'. הטקסט מוצג כאשר אנו מצביעים עם העכבר על התמונה, או כאשר אי אפשר להציג את התמונה.
- המאפיין align – מגדיר את צורת היישור של התמונה.
- המאפיין height – מגדיר את הגובה של התמונה בפיקסלים[1].
- המאפיין width – מגדיר את הרוחב של התמונה בפיקסלים.

נציג במסמך HTML תמונה של חורף, בגובה וברוחב של 100 פיקסלים, המיושרת לשמאל לעומת הכותרות שמיושרות לימין.

```
<!-- usingImage.htm -->
<!-- דוגמה לשימוש בתג תמונה -->
<body style="text-align:right">
<h1> ברוכים הבאים </h1>
<!-- הוספת תמונה -->
<h1> דף הבית שלי   <br /> <br /> </h1>
<div>
        <img src="images/Winter.jpg" align="left"
         height="100px"  width="100px"  alt="חורף" />
</div>
</body>
```

[1] פיקסל (pixel) יחידת מידע המתארת נקודה בתמונה ממוחשבת

אחד המאפיינים החשובים של התג img הוא המאפיין src, המגדיר את ערך כתובת התמונה באמצעות מבנה של URL (מְאַתר משאבים אחיד). ערך זה מאפשר לדפדפן להגיע מיד לתמונה המבוקשת. בדוגמה שלפנינו, ה-URL כולל את שם התמונה (Winter.jpg) ואת המדריך שבו נמצאת התמונה (שם המדריך הוא images). בדוגמה זו, הכתובת של המדריך היא כתובת יחסית למדריך שממנו מוצג דף HTML, משום שהיא מגדירה רק חלק מהנתיב של התמונה. לדוגמה, אם דף ה-HTML מאוחסן במדריך:

D:\asp

הרי שכתובת המדריך המלאה של התמונה היא:

D:\asp\images

השימוש בכתובת יחסית מאפשר גמישות, בין היתר הוא מאפשר להעביר את דף ה-HTML למדריך אחר (שבו נמקם את תת-המדריך images) ולהציג את דף ה-HTML בלי לשנות את תוכנו. יש כמובן להקפיד להוסיף את התמונה לפרויקט ולתיקייה המתאימה.

שאלה 2.4

צרו את דף הבית של האתר שלכם. דף הבית צריך להכיל את שמכם ופרטים אישיים נוספים שאתם רוצים להציג. נוסף על כך, הציגו בדף שלוש תמונות. הציגו כל תמונה במיקום שונה בדף (לדוגמה: למעלה, באמצע ולמטה).

2.2.5 קישורי היפר-טקסט, התג <a>

התכונה שמייחדת את שפת HTML משפות עיצוב אחרות (ושבזכותה היא נקראת שפת סימון להיפר-טקסט) היא היכולת לכלול במסמך קישור למסמך אחר, או קישור לטקסט המופיע במיקום אחר באותו מסמך. **קישור היפר-טקסט (hypertext reference)** הוא מצביע פסיבי; שלא כמו קישור לתמונה באמצעות התג <img />, המוֹרה לדפדפן לבצע פעולה מיידית, קישור היפר-טקסט הוא פריט שניתן לבחירה. פריט כזה מוצג כחלק מהמסמך ומודגש באופן מיוחד (באמצעות קו או צבע). כאשר מצביע העכבר נמצא מעל פריט זה, הוא משנה את צורתו (באינטרנט אקספלורר הוא משנה את צורתו לכף יד),

ובתחתית הדף, משמאל, נקבל את כתובת ה-URL שהקישור מפנה אליה. אם המשתמש בוחר את הפריט (על-ידי הקלקה), הדפדפן מאחזר את המסמך המקושר, ומציג אותו במקום המסמך הנוכחי (או עובר למקום אחר בתוך המסמך הנוכחי).

כל פריט יכול לשמש כקישור היפר-טקסט: מילה אחת, כמה מילים, פסקה או תמונה. הקלקה בתוך הפריט תגרום לאחזור המסמך המקושר. כדי לבנות את הקישור יש לציין, אם כן, מהו הפריט הניתן לבחירה והיכן מאוחסן המסמך המקושר.

התג שמציין קישור היפר-טקסט הוא התג **<a>** (קיצור של **anchor** – עוגן). כל מה שמופיע בין התגים <a> ו-, יוצג כפריט שניתן לבחירה (מגיב להקלקה). לתג הפותח <a> יש מאפיין בשם href, שערכו הוא ה-URL שמזהה את המסמך המקושר.

המבנה הכללי של התג הוא:

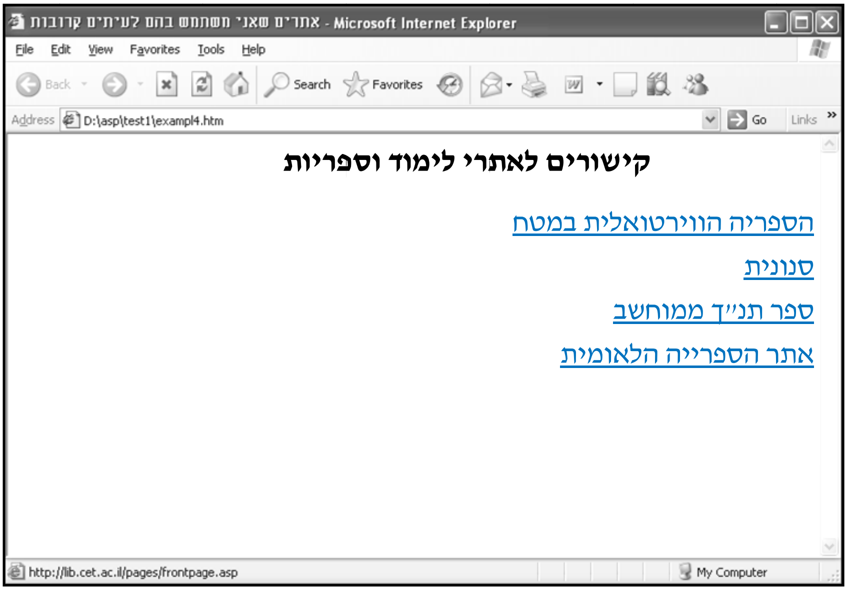
```
<a  href=”URL”>  item </a>
```

ה-item הוא הפריט הניתן לבחירה, והוא יכול להיות טקסט או תמונה. הקלקה על פריט זה תגרום לדפדפן לאחזר את הדף המזוהה על-ידי ה-URL.

לדוגמה, מוצג להלן מסמך HTML הכולל כמה קישורי היפר-טקסט. איור 2-2 מראה כיצד קובץ זה מוצג בדפדפן.

```
<!-- hyperText.htm -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head>
        <title> אתרים שאני משתמש בהם לעיתים קרובות </title>
  </head>

 <!-- דוגמה לשימוש בתג קישור -->
<body style="text-align:right">
  <h1 style="text-align:center">
        קישורים לאתרי לימוד וספריות </h1>
  <h2>
    <a href="http://lib.cet.ac.il/pages/frontpage.asp">
        ספריה וירטואלית במטח</a>
    <br /><br />
```

```
    <a href="http://www.snunit.k12.il"> סנונית </a>
    <br /><br />
    <a href="http://www.mikranet.org.il/">
             ספר תנך ממוחשב </a>
    <br /><br />
    <a href="http://jnul.huji.ac.il/heb/">
             אתר הספריה הלאומית </a>
    <br /><br />
  </h2>
</body>
</html>
```



איור 2-2
הצגת קובץ HTML בשם hypertext.htm, הכולל קישורים

שאלה 2.5

א. צרו שלושה מסמכי HTML כמפורט להלן: מסמך ראשי בשם home.htm ובו שני קישורים: קישור אחד למסמך myPictures.htm שבו נמצאות תמונות שלכם; קישור שני למסמך myFriends.htm שבו מצויות תמונות של חבריכם. לכל תמונה הוסיפו טקסט קצר המסביר את התמונה. נוסף על כך, הוסיפו בדפים myPictures.htm ו-myFriends.htm קישורים לדף הבית.

ב. למסמך הראשי שיצרתם בסעיף א', הוסיפו קישור שיאפשר שליחת דוא"ל אליכם. לשם כך השתמשו בערך "mailto" שאפשר לתת למאפיין href. לדוגמה, כדי להוסיף קישור לשליחת דוא"ל לכתובת myname@gmail.com, נרשום את התג הזה:

 לשליחת אימייל

2.2.6 טבלאות

טבלאות מסדרות את המידע במסמך באמצעות שימוש בתאים ובשורות. טבלאות ב-HTML בנויות משלושה חלקים עיקריים: הגדרת הטבלה, כותרות ושורות של תאים. לצורך הגדרת טבלה משתמשים בתגים <table> </table>. בין התגים כותבים את תוכן המידע, שורה אחר שורה תא אחר תא. כל שורה מוגדרת על-ידי התגים <tr> </tr> (**t**able **r**ow); כל תא מוגדר על-ידי התגים <td> </td> (**t**able **d**ata); תאי הכותרת מוגדרים על-ידי התגים <th> </th> (**t**able **h**eader). תאי כותרת הם תאי מידע שתוכנם ממורכז וכתוב בכתב מודגש.

להלן קוד ליצירת טבלה שיש בה שורת כותרת אחת, שתי שורות טבלה ושני תאים בכל שורת טבלה.

```
<table border="1">
 <tr>
   <th>כותרת עמודה ראשונה</th>
   <th>כותרת עמודה שנייה</th>
 </tr>
 <tr>
   <td>שורה ראשונה, תא ראשון</td>
   <td>שורה ראשונה, תא שני</td>
 </tr>
 <tr>
   <td>שורה שנייה, תא ראשון</td>
   <td>שורה שנייה, תא שני</td>
 </tr>
</table>
```

כפלט נקבל את הטבלה הזאת:

כותרת העמודה הראשונה	כותרת העמודה השנייה
שורה ראשונה, תא ראשון	שורה ראשונה, תא שני
שורה שנייה, תא ראשון	שורה שנייה, תא שני

איור 3-2
מסמך HTML שכולל טבלה בת שלוש שורות (שאחת מהן היא כותרת) ושתי עמודות

שאלה 2.6

צרו מסמך HTML בשם sol2-6.htm, שמציג טבלה ובה שלוש שורות, ובכל שורה יש שני תאים. בתא הראשון יופיע שם של אתר אינטרנט ובתא השני יופיע הקישור לאותו אתר. לדוגמה:

שם של אתר אינטרנט ראשון	קישור לאתר ראשון
שם של אתר אינטרנט שני	קישור לאתר שני
שם של אתר אינטרנט שלישי	קישור לאתר שלישי

שאלה 2.7

צרו מסמך HTML בשם sol2-7.htm, המכיל טבלה שבה שתי שורות ושלוש עמודות. כל תא בטבלה יכיל תמונה המשמשת כקישור היפר-טקסט. כאשר משתמש ילחץ על תמונה כלשהי, הוא יועבר לאתר המקושר.

כדי לכלול תמונה בתא בטבלה, נרשום את התגים המפורטים להלן. שימו לב לאופן קינון התגים הפותחים והסוגרים:

```
<td>
   <a href=...>
     <img ="..." width="..." src="..." alt="..."            />
   </a>
</td>
```

טבלה ניתן לעצב בעזרת מאפיינים של כל אחת מהתגים המשמשים בהגדרת הטבלה. לדוגמה, בעזרת התכונה background-color של המאפיין style, אנו קובעים את צבע הרקע של הטבלה, של השורה או של התא, בהתאם לתג שבו השתמשנו בתכונה. עוביו של הקו

המפריד בין התאים נקבע על-ידי המאפיין border, המציין ערך מספרי חיובי ושלם. סגנון הקו נקבע על-ידי התכונה border-style. ברירת המחדל של התכונה היא ללא סימון גבול (none). ניתן לשנות את ברירת המחדל ולהשתמש באחד מסגנונות הקו נפוצים שהם: קו (solid), קו מקווקו (dashed) ונקודות (dotted). התכונה border-color קובעת את צבעו של הקו. המרווח בין הטקסט שבתא לגבול התא נקבע על-ידי המאפיין cellpadding, המציין אף הוא ערך מספרי חיובי ושלם. את הפירוט של מאפייני התגיות ניתן למצוא בנספח ב׳ שבסוף פרק זה.

להלן דוגמה לטבלה שבה מוגדר צבע הגופן, צבע הרקע למקצת התאים, סגנון הקו, עובי גבול הטבלה והמרווח בין הטקסט שבתא לבין גבול התא.

```
<table style="border:2; border-style:dashed; border-color:Black" cellpadding="4">
  <tr>
    <td style="background-color:Black; color:White">אמסטף</td>
    <td style="border-color:Black; border-style:dotted">בלקי</td>
  </tr>
  <tr>
    <td style="border-color:Black; border-style:dotted">פודל</td>
    <td style="background-color:Gray">מוקי</td>
  </tr>
  <tr>
    <td style="background-color:Black; color:White">דוברמן</td>
    <td style="border-color:Black; border-style:dotted">שוקי</td>
  </tr>
</table>
```

כפלט נקבל את הטבלה הזאת:

אמסטף	בלקי
פודל	מוקי
דוברמן	שוקי

איור 4-2
טבלה בת שלוש שורות ושתי עמודות המשמשת כמאפייני הטבלה

לעיתים נרצה ליצור בראש הטבלה שורת כותרת, כך שכל התאים בשורה יתמזגו לתא אחד. בצורה דומה אפשר שנרצה למזג שורות לתא אחד. מיזוגים אלו אפשריים בעזרת המאפיינים colspan, rowspan של התג <td>. המאפיין colspan קובע על כמה עמודות יתפרס התא. המאפיין rowspan קובע על כמה שורות יתפרס התא. בצורה זו ניתן להגדיר

טבלה מורכבת, המכילה תאים ארוכים ותאים גבוהים, הנפרסים על פני כמה שורות או עמודות. לדוגמה, לפניכם טבלה בעלת שש שורות ושלוש עמודות, שחלק מתאיהן מוזגו ועוצבו בעזרת גופנים שונים:

```
<table>
  <tr >
    <td colspan="3" style="font-family:Guttman Stam; text-align:center;
                        background-color:Black; color:White">חיות לאימוץ</td>
  </tr>
  <tr style="font-family:Guttman Yad-Brush">
    <td style="border-style:solid; border-width:thin">סוג</td>
    <td>שם</td>
    <td>גזע</td>
  </tr>
  <tr style=" font-family:Guttman Arial (Hebrew)">
    <td rowspan="2" style=" font-family:Guttman Yad-Brush">חתולים</td>
    <td>לאקי</td>
    <td>פרסי</td>
  </tr>
  <tr style=" font-family:Guttman Arial (Hebrew)">
    <td>פרינס</td>
    <td>אנגורה</td>
  </tr>
  <tr style=" font-family:Guttman Arial (Hebrew)">
    <td rowspan="2" style=" font-family:Guttman Yad-Brush">כלבים</td>
    <td>מוקי</td>
    <td>פודל</td>
  </tr>
  <tr style=" font-family:Guttman Arial (Hebrew)">
    <td>בלקי</td>
    <td>דוברמן</td>
  </tr>
 </table>
```

כפלט נקבל את הטבלה הזאת:

חיות לאימוץ		
סוג	שם	גזע
חתולים	לאקי	פרסי
	פרינס	אנגורה
כלבים	מוקי	פודל
	בלקי	דוברמן

איור 2-5
טבלה שבה מוזגו תאים

שאלה 2.8

צרו מסמך HTML בשם sol2-8.htm, המכיל את רשימת הדיסקים שברשותכם. כל שורה תכיל את שם הזמר ו/או הלהקה ואת שם הדיסק שהוציא. דיסקים שונים של אותו מבצע יוצגו בשורות עוקבות כך ששם המבצע יצוין פעם אחת בלבד בתא, ולידו יצוינו שמות הדיסקים, בזה אחר זה בשורות עוקבות. כותרת הטבלה תוצג בשורה הראשונה שבנויה משני תאים אשר מוזגו לתא אחד.

לפניכם דוגמה לטבלה העונה על הדרישות:

<table dir="rtl">
<tr><th colspan="2">הדיסקים שלי</th></tr>
<tr><th rowspan="3">יהודה פוליקר</th><td>אפר ואבק</td></tr>
<tr><td>עיניים שלי</td></tr>
<tr><td>פחות אבל כואב</td></tr>
<tr><th rowspan="2">יהודית רביץ</th><td>באה מאהבה</td></tr>
<tr><td>עיר קטנה</td></tr>
</table>

איור 2-6
טבלה המכילה נתוני דיסקים

עצבו את הטבלה והשתמשו בסוגי גבולות שונים, בצבעים וברקעים שונים.

שאלה 2.9

צרו מסמך HTML בשם sol2-9.htm, המציג את מערכת השעות שלכם בצורת טבלה. השורה הראשונה תכיל כותרת, השורה השנייה - את ימות השבוע, התא הראשון בכל שורה - את מספר השעה במערכת ותאי הטבלה יכילו את מערכת השעות; שיעורים כפולים יוצגו בתאים שמוזגו, כמתואר בטבלה שלפניכם.

<table dir="rtl">
<tr><th colspan="7">מערכת השעות שלי</th></tr>
<tr><th></th><th>ראשון</th><th>שני</th><th>שלישי</th><th>רביעי</th><th>חמישי</th><th>שישי</th></tr>
<tr><th>1</th><td>אנגלית</td><td>מתמטיקה</td><td>חנ״ג</td><td>ספרות</td><td rowspan="2">לשון</td><td>תנ״ך</td></tr>
<tr><th>2</th><td rowspan="2">מתמטיקה</td><td>אנגלית</td><td>תנ״ך</td><td rowspan="2">כימיה</td><td rowspan="3">מתמטיקה</td></tr>
<tr><th>3</th><td>הסטוריה</td><td>אנגלית</td><td>ספרות</td></tr>
<tr><th>4</th><td rowspan="2">הסטוריה</td><td>מתמטיקה</td><td>ספרות</td><td rowspan="2">תנ״ך</td><td>חנ״ג</td></tr>
<tr><th>5</th><td>לשון</td><td>ספרות</td><td>מתמטיקה</td><td>ספרות</td></tr>
</table>

איור 2-7
טבלת מערכת שעות

2.2.7 רשימות

שפת HTML מאפשרת ליצור רשימות ממוספרות ורשימות לא-ממוספרות. הסוג הפשוט ביותר הוא רשימה 'לא מסודרת', כלומר, רשימה של פריטים ללא מספור. התגים <ul> ו- תוחמים את הרשימה; כל פריט ברשימה נרשם בין התגים <li> (קיצור של list item) ו- .

להלן דוגמה לרשימה שמעוצבת על-ידי המאפיינים והתכונות האלה: text-align – הרשימה תיושר לימין; dir (קיצור של direction) – כיוון הכתיבה יהיה right to left (rtl), מימין לשמאל.

```
<ul style="text-align:right" dir="rtl" >
   <li> פריט אחד </li>
   <li> פריט שני </li>
   <li> פריט שלישי </li>
</ul>
```

ובדפדפן תוצג הרשימה בצורה הזאת:

- פריט ראשון
- פריט שני
- פריט שלישי

כדי ליצור רשימות ממוספרות, יש להשתמש בתגים <ol> ו- , במקום ב-<ul>.

2.2.8 טפסים

אתרים רבים מבקשים מהגולשים מידע, כגון שם, סיסמה וכתובת דוא"ל. את המידע המבוקש יש למלא בטופס להקלדת נתונים. טופס מורכב מכמה חלקים: טקסט המכיל הסברים והוא מופנה לגולש; תיבות טקסט (שדות הטופס) שבהן הגולש מקליד את המידע וכפתור לשליחת הנתונים אל שרת האתר.

תיבות הטקסט וכפתור השליחה הם אלמנטים של שפת HTML, ומטרתם לקבל קלט מהמשתמש. קיימים אלמנטים נוספים, כגון כפתורי רדיו (radio button) המשמשים לבחירת פריט מתוך אוסף פריטים, תיבות סימון (check box) המאפשרות סימון של פריטים מתוך אוסף פריטים, כפתור reset המוחק את הטקסט שהוקלד בכל שדות הטופס,

תיבת סיסמה המשמשת לכתיבת טקסט שאינו מוצג על המסך (במקומו מוצגים עיגולים או כוכביות) ועוד.

כל האלמנטים האלה מוגדרים על-ידי התג המיידי <input /> ועל-ידי המאפיין type. ערך המאפיין (type) של תיבת טקסט הוא text ; ערך המאפיין של כפתור שליחת הנתונים הוא submit וערך המאפיין value, הקיים בעבור כפתור שליחת הנתונים (submit), מגדיר את הטקסט שיהיה רשום על הכפתור ; ערך המאפיין של תיבת סיסמה הוא password ; ערך המאפיין של כפתורי רדיו הוא radio וערך המאפיין של תיבות הסימון הוא checkbox. המאפיין name מגדיר את שמו של האלמנט, כך שבהמשך יהיה ניתן להתייחס אליו בקטע אחר של הדף ; המאפיין id מזהה ייחודי של אלמנט HTML.

המאפיינים name ו-id משמשים למטרות דומות ומטרתם לייחד את האלמנט. בעבר השתמשו לצורך כך במאפיין name בלבד. בעתיד הכוונה להשתמש במאפיין id בלבד, אולם כיוון שקיימים עדיין דפדפנים שאינם מזהים את המאפיין id, יש להגדיר גם את המאפיין id וגם את המאפיין name. ערכם של שני מאפיינים אלה הוא זהה ביחס למרבית האלמנטים. ערכם אינו זהה רק ביחס לכפתורי הרדיו, כפי שנציג בהמשך.

קליטת הנתונים נעשית גם על –ידי התגים <textarea> </textarea>, שמשמשים לקליטת טקסט חופשי המורכב ממספר שורות, והתגים <select> </select> , המשמשים לבחירה של אפשרות אחת מתוך רשימה נגללת של אפשרויות, כגון בחירת עיר מגורים מתוך רשימת ערים, בחירת תחביב מתוך רשימת תחביבים מוגדרת וכו׳.

מאפייני תג הטקסט החופשי <textarea> מגדירים את שמו (name), זיהויו (id), רוחבו, כלומר מספר התווים בשורה (cols) וגובהו – מספר השורות (rows). מאפייני תג הרשימה הנגללת מגדירים את שמו (name), זיהויו (id), גודל הרשימה (size) ואם ניתן לבחור יותר מאפשרות אחת (multiple). את האפשרויות עצמן מגדירים בין התגים <option> </option> הקיימים רק בין תגי <select> </select>. מאפייני התג <option> הינם value, המציין את ערך הבחירה, ו-selected המציין את הבחירה שתוצג ראשונה.

טופס מוגדר על-ידי התגים <form> </form>, וכולל את המאפיין id המזהה את הטופס, את המאפיין action המגדיר לאן יישלח המידע שבטופס ואת המאפיין method אשר מגדיר את דרך שליחת הנתונים בעת לחיצה על הכפתור submit. נתוני טופס יכולים להישלח לשרת לפי אחת משתי שיטות האלה : get או post (שבהן נעסוק בהרחבה בפרק 3).

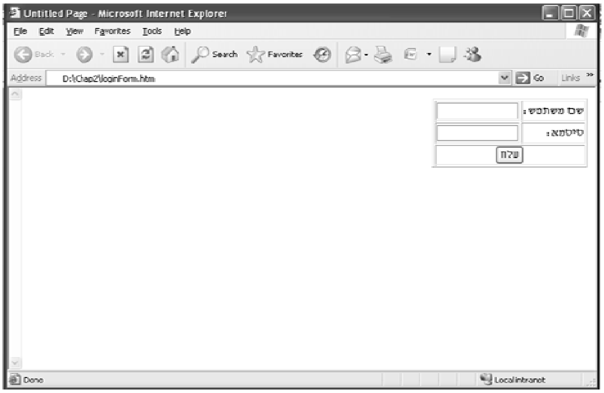
לפניכם דוגמה לטופס בסיסי המכיל את שדות הטופס ואת כפתור שליחת הנתונים – submit.

```
<form id   = [מזהה הטופס]
        name= [שם]
        action= [כתובת קובץ בשרת שאליו יישלחו נתוני הטופס (כולל המסלול)]
        method= [get או post :השיטה שלפיה יישלחו נתוני הטופס]   >
        <input type="text" name="field1" id="field1" size="20" />
         הקלד נתוני שדה 1
         <input type="text" name="field2" id="field2" size="20" />
         הקלד נתוני שדה 2
        …
         <input type="submit" value="שלח"/>
         <input type="reset" value="נקה" />
</form>
```

אחד הטפסים הנפוצים ביותר הוא טופס הכניסה לאתר, המבקש מהמשתמש להקליד שם וסיסמה. לפניכם הגדרה של טופס המבצע משימה זו :

```
<form  id="Login" method="post">
 <table border="0">
  <tr>
    <td>שם משתמש: </td>
    <td><input type="text" name="txtAccount"
               id="txtAccount" size="10" /> </td>
  </tr>
  <tr>
    <td>סיסמא: </td>
    <td><input type="password" name="Password"
               id="Password" size="10" /> </td>
  </tr>
  <tr>
    <td colspan="2" align="center">
              <input type="submit"  size="10" name="btnSubmit"
               id="btnSubmit" value="שלח" />
    </td>
  </tr>
 </table>
</form>
```

דף המכיל הגדרות אלו יוצג בדפדפן כך :



איור 2-8
טופס לקליטת שם משתמש וסיסמה

טבלה 2.1 מציגה את סוגי האלמנטים העיקריים השייכים לתג < input….. /> :

טבלה 2.1

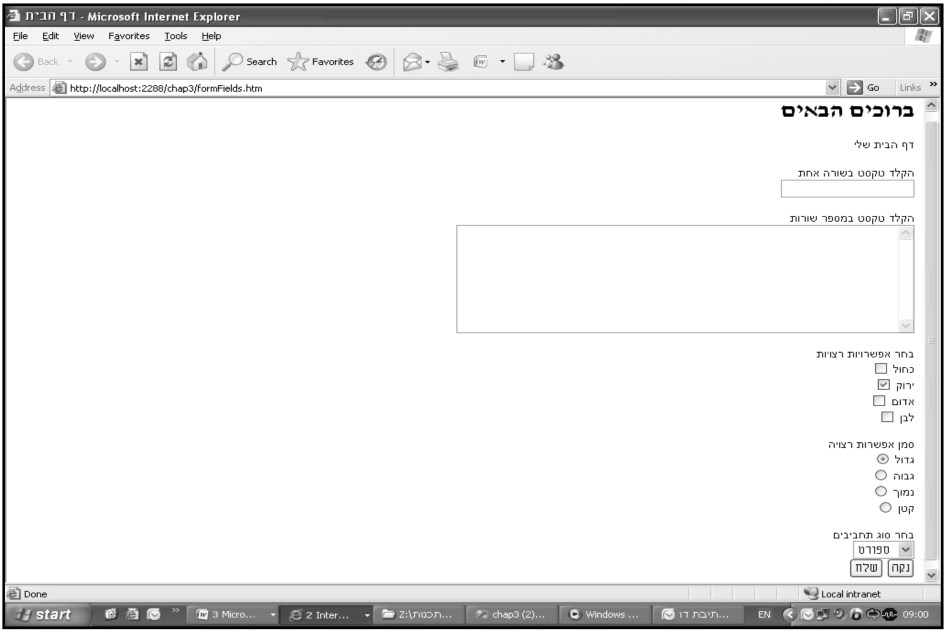
סוג האלמנט	המאפיינים וערכיהם
טקסט	<input type = text size = …. name = …… id= …… maxlength = …. value = …. >
סיסמה	<input type = password size = …. name = …… id= …… maxlength = …. value = …. >
כפתורי רדיו	<input type = radio name = …… id= …… value = …. checked = … >
תיבות סימון	<input type = checkbox name = …… id= …… value = …. checked = … >
כפתור (לחצן)	<input type = button name = …… id= …… value = …. >
כפתור לניקוי נתונים שהוכנסו לטופס	<input type = reset value = …. >
כפתור לשליחת טופס	<input type = submit value = …. >

להלן דוגמה של דף HTML המשתמש באלמנטים שהוצגו להלן. כמו-כן, באיור 9-2 מוצג הטופס שהדפדפן מציג למשתמש.

שימו לב, שמם של כל כפתורי הרדיו הוא rb (radio button). לכפתורים האלה ניתן שם זהה כדי שיהיה אפשר לזהותם כאוסף אחד שממנו מתאפשרת בחירה של פריט אחד. עם זאת, לכל כפתור יש id משלו, המורכב משם של כפתור הרדיו וממספר סידורי. מאותה סיבה (אם כי הדבר אינו הכרחי), שמן של כל תיבות הסימון הוא cb (checkbox), ולכל תיבה יש id משלה, המורכב משם הפקד וממספר סידורי.

```
<!-- formFields.htm -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
<title> דף הבית </title>

</head>
<!-- דוגמה לשימוש בתג פסקה -->
<body style="text-align:right;">
   <h1 > ברוכים הבאים </h1>
   <!-- מקטע ראשון -->
   <div style="color:black; font-family: sans-serif">
   דף הבית שלי <br />

   <form id="form1" name= "form1" action=" formFields.htm">
   <div >
      הקלד טקסט בשורה אחת <br />
      <input type="text" size="20" maxlength="10" />   <br /><br />
      הקלד טקסט במספר שורות <br />
      <textarea rows="10" cols="60" name="test" id="test"></textarea><br /><br />
      <!-- תיבת סימון -->
      בחר אפשרויות רצויות   <br />
      <input type="checkbox" name="cb" id="cb1" value="blue" /> כחול <br />
      <input type="checkbox" name="cb" id="cb2" value="green"
               checked="checked" /> ירוק <br />
      <input type="checkbox" name="cb" id="cb3" value="red" /> אדום <br />
      <input type="checkbox" name="cb" id="cb4" value="white" /> לבן <br /><br />
      <!-- תיבת רדיו -->
      סמן אפשרות רצויה <br />
      <input type="radio" name="rb" id="rb1" value="big"
               checked="checked" /> גדול <br />
```

```
<input type="radio" name="rb" id="rb2" value="high" />
         גבוה <br />
<input type="radio" name="rb" id="rb3" value="low" />
         נמוך <br />
<input type="radio" name="rb" id="rb4" value="little" />
         קטן <br /><br />
<!-- רשימה נגללת -->
בחר סוג תחביבים<br />
         <select name="hobbies">
            <option value="sport">ספורט</option>
            <option value="art">אומנות</option>
            <option value="science">מדע</option>
         </select><br />
<!-- לחצן לשליחת הנתונים -->
<input type="submit" value="שלח" />
<!-- לחצן איפוס הנתונים -->
<input type="reset" value="נקה" />
</div>
</form>

</div>
</body>
</html>
```



איור 2-9

טופס לקליטת נתונים

שדות נסתרים (hidden fields)

שדות נסתרים הם שדות הדומים לשדות טקסט, אלא שבניגוד לשדות הטקסט, המשתמש אינו יכול לראות אותם ולכן לא יכול להקליד בהם ערכים. השדות הנסתרים נועדו כדי להעביר נתונים מלקוח לשרת מבלי שהלקוח יצטרך להכניס נתונים אלה. לדוגמה, אם נרשום את התגים הבאים:

```
<input type=”hidden” name=”id” value=”123456”>
<input type="text" name "name" size="20" maxlength="10" />
```

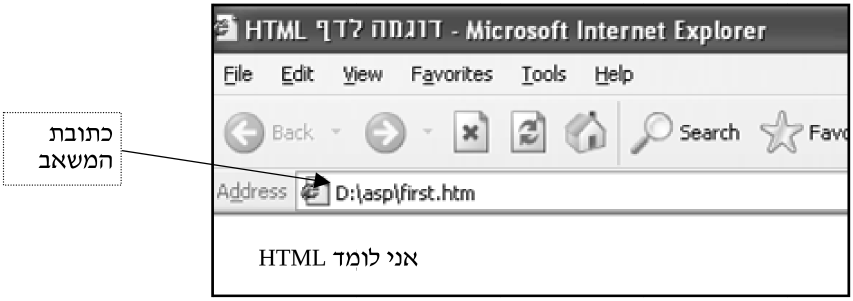
למשתמש יוצג רק שדה טקסט אחד שבו הוא יקליד את שמו, אך לשרת יישלחו שני נתונים: השם שהקליד המשתמש והערך "123456" של השדה הנסתר id.

2.3 פנייה לדף סטטי בשרת

דף HTML מוצג בדפדפן. כדי להציג את הדף בדפדפן, יש לרשום את שם הקובץ ואת המיקום שלו במחשב. לדוגמה, נציג את הקובץ first.htm שנמצא בספרייה D:\asp בדפדפן. כתובת המשאב היא אפוא:

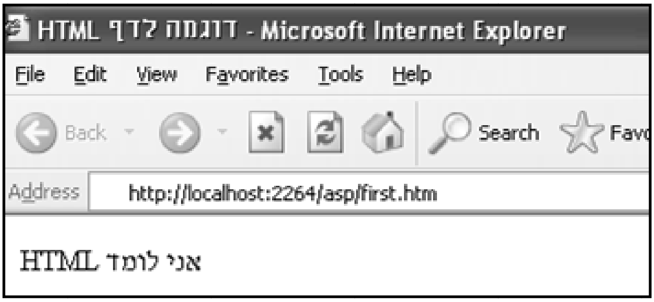
D:\asp\first.htm

לשם כך עלינו להקליק פעמיים על שם הקובץ או, לחילופין, לפתוח את הדפדפן ולרשום בתיבת הטקסט לכתיבת הכתובת שבדפדפן, כפי שמוצג באיור 2-10.



איור 2-10
הצגת דף HTML כדף לקוח שכתובתו היא D:\asp\first.htm

כדי להריץ את אותו הדף מתוך השרת – בסביבת Visual Studio – נשתמש באפשרות 'View in Browser' (ראו נספח א'). גם הפעם יוצג הדף בדפדפן, כפי שמוצג באיור 2-11, אך שימו לב כי הכתובת של המשאב השתנתה.



איור 2-11
הרצת דף HTML כדף סטטי הרץ בשרת

הכתובת כתובה במבנה URL

http://localhost:2264/asp/first.htm

כאשר הפרוטוקול הוא HTTP, שם התחום של המחשב הוא localhost ומספר המִפתח הוא 2264. המיקום של המשאב הוא במדריך asp (שהוא תת-מדריך במדריך שממנו מופעל השרת) ושם הדף הוא first.htm.

שם התחום, localhost, הוא שם שמציין את המחשב המקומי (המחשב שממנו מריצים את השרת). כתובת ה-IP שמוקצית לשם התחום localhost היא תמיד 127.0.0.1. מספר המִפתח נבחר על-ידי סביבת העבודה Visual Studio הבוחרת מִפתח שאינו תפוס על-ידי יישום כלשהו.

ניתן כמובן לרשום כתובת IP בתיבת הכתובת בדפדפן, במקום לרשום את שם התחום, לדוגמה:

http://127.0.0.1:2264/asp/first.htm

2.4 גיליונות סגנון מדורגים (CSS)

גיליונות סגנון מדורגים (**Cascading Style Sheets**), ובקיצור **CSS**, מאפשרים להפריד בין העיצוב ובין התוכן של דפי HTML, וכך להקל על העיצוב ועל הפיתוח של הדפים. CSS מאפשר למעצב לקבוע כיצד יוצג כל אלמנט בדף HTML, באיזו תבנית נשמר הקול בקובץ המוזיקה או הדיבור, כיצד ייראה הדף בדפדפן, וכדומה. עד כה השתמשנו במאפיין style

לעיצוב כל הדף, לעיצוב מקטע בדף או לעיצוב אלמנט מסוים. CSS מאפשר להגדיר סגנון פנימי לדף, כך שהגדרת הסגנון תחול על כל התגים בדף. לדוגמה, כדי להגדיר את תגי הכותרת h1 בגופן מסוג sans serif ובגודל גופן 24, צריך לכתוב את השורות האלה בראשית הדף:

```
h1 {
   font-family: san-serif;
   font-size: 24pt;
}
```

כל שימוש בתג h1 בהמשך הדף יגרום לעיצוב הכותרות בצורה זהה. אם נחליט לשנות את העיצוב של כותרות h1, יהיה עלינו לשנות רק את הגדרת התג, ולא לשנות כל כותרת h1 בנפרד.

תקן CSS נקבע על-ידי איגוד האינטרנט העולמי (**World Wide Web Consortium**). ה-W3C מפרסם גם תקנים נוספים בתחום הצגת תוכן ב-Web. הקפדה על יישום תקני האינטרנט (Web Standards) בכל אתר אינטרנט ברשת נועדה לגרום לכך שכל דפדפן תקני יציג את האתרים באופן דומה. כך, כותבי האתרים לא יצטרכו לכתוב את דפי האתרים בגרסאות שונות, המתאימות לכל דפדפן.

כפי שכבר ראינו, ניתן לקבוע סגנון במספר דרכים. נציגם להלן.

שיטה ראשונה – המאפיין style (Inline)

שיטה זו כבר הצגנו קודם לכן. לפי שיטה זו קובעים את הסגנון של כל אלמנט בנפרד. למשל, נגדיר מקטע שהרקע שלו צהוב:

```
<div style="background-color: yellow"> xxx  </div>
```

סגנון זה נקרא inline משום שהוא מגדיר רק מקטע מסוים בדף. הוא אינו מתייחס לאלמנטים אחרים בדף או באתר.

שיטה שנייה – התג style (Internal)

לפי שיטה זו מגדירים סגנונות כלליים בעבור כל הדף, בדרך-כלל, בתוך התג head. כותבים את התג <style>, לאחריו מגדירים סגנונות (כפי שיוסבר בהמשך), ואז סוגרים את התג באמצעות </style>.

לדוגמה, במסמך HTML שלהלן אנו מגדירים סגנון כללי לכותרות מרמה ראשונה (h1) וסגנון כללי לכותרות מרמה שנייה (h2). כל שימוש בתג כותרת בגוף הדף יוצג בהתאם לסגנון שקבענו בתג <style> :

```
<!-- internalCss.htm -->
<html>
<head>
   <title>Internal Style</title>
   <style type="text/css">
     h1 {
          font-family: san-serif ;
          font-size: 40pt ;
          color: #00ff00 ;
        }
     h2 {color: #dda0dd}
   </style>
</head>
<body>
  <h1> שלום עולם</h1>
  <h2> ברוכים הבאים</h2>
  <h1> שימוש חוזר בכותרת </h1>

</body>
</html>
```

האלמנטים שהוגדרו באמצעות התג style מאפשרים לקבוע את כל הסגנונות בדף ה-HTML הנוכחי.

שיטה שלישית – קישור לגיליון עיצוב חיצוני (External)

נהוג ליצור גיליון עיצוב CSS חיצוני שבו מוגדרים הסגנונות המשמשים את כל דפי האתר. כל דף HTML המקושר לגיליון העיצוב מעוצב על-פי הסגנונות המוגדרים בגיליון.

גיליון העיצוב הוא קובץ טקסט שהסיומת שלו היא css. בדומה לכל קובץ אחר, ניתן למקם אותו בשרת האינטרנט או בכונן הקשיח של מחשב לקוח. אז יוצרים קישור ממסמך

ה-HTML (קובץ בעל סיומת htm) לגיליון העיצוב (קובץ בעל סיומת css). קישור כזה נוצר באמצעות התג link. להלן דוגמה:

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

הנתיב לגיליון העיצוב מוגדר על-ידי המאפיין href. הדפדפן משתמש במאפיין קישור זה כדי להציג את קובץ ה-HTML, כשהוא משתמש בתבנית העיצוב שמוגדרת בקובץ ה-css שמופיע בערך של המאפיין href. התכונה rel מגדירה את הקשר בין המסמך הנוכחי למסמך המוגדר בעוגן במאפיין href. תג הקישור חייב להיכלל בקטע ה-header של קוד ה-HTML, כלומר בין התגים <head> ל-</head>. למשל כך:

```
<!-- externalCss.htm -->
<html>
  <head>
    <title>My document</title>
    <link rel="stylesheet" type="text/css"  href="style.css" />
  </head>
  <body>
</html>
```

במקרים שבהם מוגדרים סגנונות שונים לאותו תג, נבחר הסגנון 'הקרוב' ביותר להגדרת התג. כלומר, אם הוגדר סגנון inline, ייעשה בו שימוש; אם לא הוגדר סגנון inline, ייעשה שימוש בסגנון internal; אם לא הוגדרו סגנון inline או סגנון internal, ייעשה שימוש בסגנון החיצוני external.

בעת כתיבת אתר, המכיל מספר דפים בעלי סגנונות אחידים, נהוג לכתוב קובץ הגדרת סגנונות אשר כל דפי האתר מקושרים אליו. היתרון של השיטה שהיא מפשטת את התחזוקה של האתר, והחיסרון שלה הוא שטעינת דף מהאתר נמשכת זמן ממושך יותר. הסיבה לכך היא שיש לטעון לא רק את הדף עצמו, אלא גם את דף הסגנונות. לפיכך יש להקפיד על כתיבת קובצי css קטנים.

שאלה 2.10

א. העתיקו את שורות הקוד שלעיל (בעמוד הקודם) לתוך קובץ, כתבו את שמכם בכותרת מרמה 1. שמרו את הקובץ בשם sol2-10.htm .

ב. כתבו קובץ בשם style.css, שבו יוגדר העיצוב של דף HTML בעל רקע אדום. עצבו את הכותרות מרמה 1 בגופן sans serif. לשם כך כתבו בקובץ את ההוראות שלהלן, ושמרו אותו בספרייה שבה שמרתם את הקובץ sol2-10.htm. אם הנכם עובדים בסביבת Visual Studio , פתחו את שני הקבצים באותו פרויקט.

```
<!-- style.css -->
body {
        background-color: blue;  }
h1    {
        font-family: sans-serif ; }
```

ג. השתמשו בדפדפן כדי להציג את הקובץ sol2-10.htm.

לגיליונות סגנון חיצוניים ופנימיים (לא inline) יש תחביר פשוט :

- selector – אלמנט ה-HTML שאליו מתייחסים או שם כללי של הסגנון.
- property – המאפיין שאותו רוצים להגדיר.
- value – הערך שנקבע במאפיין.

להלן המבנה של התחבירי של סגנון :

value;}	{property :	selector
הערך שהמאפיין יכול לקבל (לדוגמה : "blue")	המאפיין (לדוגמה : צבע רקע "background-color")	לאילו תגי html המאפיין מתייחס (לדוגמה : "table").

לדוגמה, כאשר רוצים שהרקע של כל הטבלאות יהיה כחול, רושמים :

```
table{backgound-color:blue}
```

דף HTML ניתן לסימון בדרכים שונות ודפדפנים יכולים להציג דף HTML בצורות רבות. במילים אחרות, ניתן להגיד שבשפת ה-HTML יש "ניבים" רבים. כיום מנסים ליצור תקן משותף ל-HTML באמצעות הארגון **(W3C)** World Wide Web Consortium . כך, כאשר מקודדים דף HTML בהתאם לתקני ה-W3C, יהיה אפשר לצפות בו באמצעות כל הדפדפנים – כיום ובעתיד. כיום משתמשים רוב מפתחי האתרים בתקני HTML 4.01

ו-XHTML (XHTML – eXtensible HyperText Markup Language). למעשה XHTML היא שילוב של HTML ו-XML.[2] בשלב זה איננו צריכים להכיר את שפת XML כדי להשתמש ב-XHTML.

מאחר שקיימות גרסאות שונות של HTML, עלינו להגיד לדפדפן מהו ה"ניב" של HTML שבו אנו משתמשים. כדי לעשות זאת, יש לרשום הצהרת סוג מסמך. הצהרת סוג המסמך נכתבת תמיד בראש המסמך: כאשר משתמשים בסביבת Visual Studio, שורות אלו מתווספות אוטומטית לדף HTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="he" xml:lang="he" dir="rtl">
```

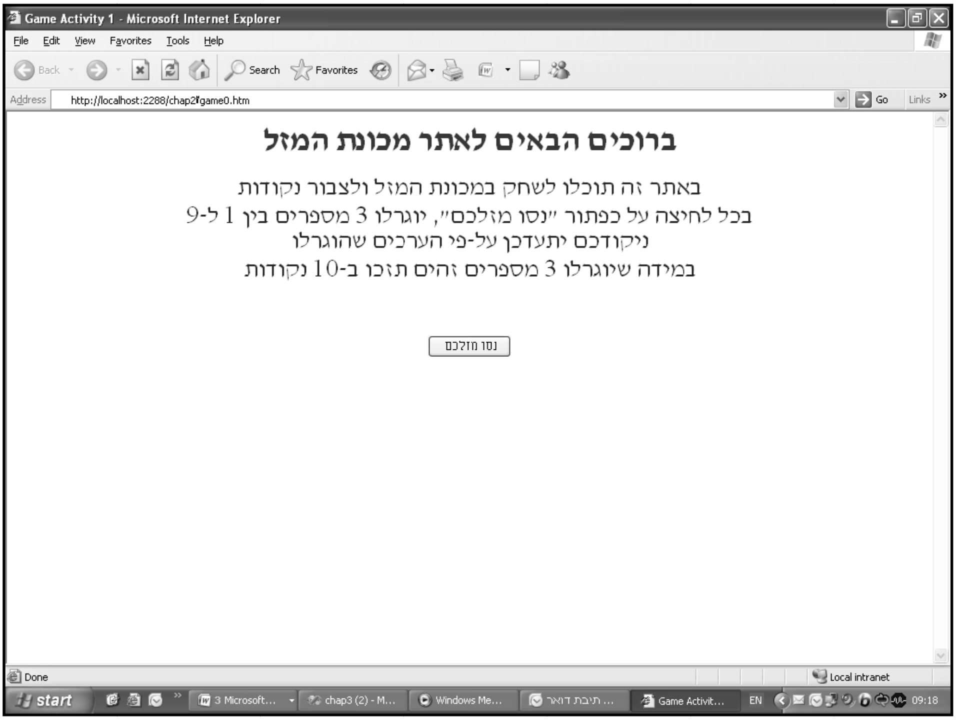
2.5 פעילות מסכמת – בניית דף HTML

משימה א'

במסגרת פרויקט אשר ילווה את הספר הזה, ניצור אתר שבו יוצג משחק מכונת מזל. מכונת המזל מגרילה בכל פעם 3 מספרים בין 1 ל-9. בתחילת המשחק יעמדו לרשות המשתמש 0 נקודות, והניקוד שיינתן לו יתעדכן על-פי הערכים שיוגרלו ועל-פי הכלל הזה: אם המשתמש יקבל בו-זמנית 3 מספרים זהים, הוא יקבל 10 נקודות. המסך שיוצג למשתמש מתואר באיור 2-12.

---

[2] XML (**EX**tensible **M**arkup **L**anguage) היא שפת תגים שמטפלת במידע; לעומת זאת, HTML מאפשרת להציג מידע.



איור 2-12
מסך פתיחה למשחק מכונת המזל

להלן דף HTML הדרוש. בשלב זה האתר כולל דף אחד, ועל כן הגדרת הסגנונות תהיה פנימית ותתבצע על-ידי הגדרת התג style.

```
<!—game0.htm -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>Game Activity 1</title>
    <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1" name="form1" action="game0.htm">
   <div>
     <h1>ברוכים הבאים לאתר מכונת המזל</h1>
     <p>
        <br /> באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות.
        <br /> בכל לחיצה על כפתור "נסו מזלכם", יוגרלו 3 מספרים בין 1 ל–9
```

```
        <br /> ניקודכם יתעדכן על–פי הערכים שהוגרלו.
        <br /> במידה שיוגרלו 3 מספרים זהים תזכו ב–10 נקודות.
      </p>
      <br /><br />
      <input type="submit" value=" נסו מזלכם" name="Send" id="Send"/>
      <br /><br />
    </div>
    </form>
</body>
</html>
```

להלן קובץ לעיצוב הפלט Chap3.css

```
body {   text-align:right;
        direction: rtl }
   h1 {color:red;
       direction:rtl}
   h3 {color:Blue;
       direction:rtl}
   p  {color:Purple;
          font-size:larger;
           text-align:right}
   div {direction: rtl}
```

שמרו את הקובץ עם סיומת Game0.htm להצגת הדף הקליקו (הקלקה כפולה) על שם הקובץ. הדפדפן יפתח אותו ויריץ את הדף כמתואר באיור 2-12.

משימה ב'

תכננו דף משחק למכונת המזל, שכללי הניקוד בו יוגדרו כך :

היחס בין שלושת המספרים	הניקוד שיצבור השחקן
שלושת המספרים שונים זה מזה	-1
שניים מבין המספרים שווים זה לזה	+1
שלושת המספרים זהים זה לזה	+10
שלושת המספרים זהים ל-7	+50

באתר יוצגו חוקי המשחק, והמשתמש יוזמן לשחק במכונת המזל. שמרו דף זה בשם MyGame.htm. לפניכם דוגמה לדף המשחק.

לפני בניית דף ה-HTML עלינו לתכנן אותו. לשם כך נשתמש בדף משובץ שבו נרשום ונמקם את הכותרות השונות שאמורות להופיע בדף המשחק. איור 2-13 מציג דוגמה לעריכה של דף הבית. נוסף על כך, רצוי לצרף לטקסט המוצג הוראות עיצוב המתייחסות לגודל התווים, לסוג הגופנים, לצבע וכדומה.

ברוכים הבאים לאתר מכונת המזל			
באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות. **בכל לחיצה על כפתור "נסו מזלכם", יוגרלו 3 מספרים בין 1 ל-9.** **הניקוד יתעדכן על-פי הערכים שיוגרלו.** **כללי הניקוד הם:**			
	התוצאה	**הניקוד**	
	שלושת המספרים שונים זה מזה	**-1**	
	שניים מבין המספרים שווים זה לזה	**+1**	
	שלושת המספרים זהים זה לזה	**+10**	
	שלושת המספרים זהים ל-7	**+50**	
	לרשותכם 0 נקודות		
	נסו מזלכם		

איור 2-13
הצעה לעיצוב דף משחק חדש

השתמשו בסגנונות ובתגים כדי להגדיר טבלה. הטבלה תאפשר להציג את דף המשחק באופן אסתטי וברור. אפשר לעצב את הטקסט שמוצג בדף כולו בתוך טבלה, ואפשר לעצב רק את כללי הניקוד בתוך טבלה.

שימו לב, הטבלה מכילה תאים המיושרים לימין ותאים המיושרים למרכז. צריך אם כן להשתמש במספר צורות להגדרת סגנון.

נספח א':

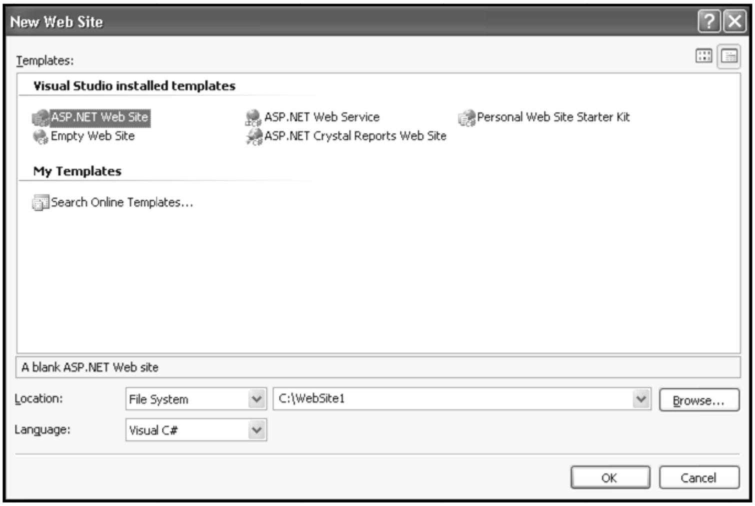
יצירת מסמך HTML והרצתו בסביבת Microsoft Visual Studio

נספח זה מתאר כיצד יוצרים קובץ HTML ומריצים אותו כדף סטטי מתוך שרת. סביבת העבודה שבה אנו משתמשים היא Microsoft Visual Studio developer 2008, שאותה ניתן להוריד מהרשת ולהתקינה במחשב.

א. פתיחת יישום שרת

בחרו בתפריט ראשי File ← New ← Web Site. בחלון שנפתח (איור 2-10) :

- בחרו ביישום רשת מסוג ASP.Net Web site
- קבעו את שם היישום ואת המקום שבו תשמרו אותו. לדוגמה, D:\asp\htmltest.
  אם הספרייה לא קיימת, תתקבל הודעה מתאימה אשר תשאל אם ליצור ספרייה. יש לבחור במקש OK.
- הגדירו את שפת התכנות #Visual C.

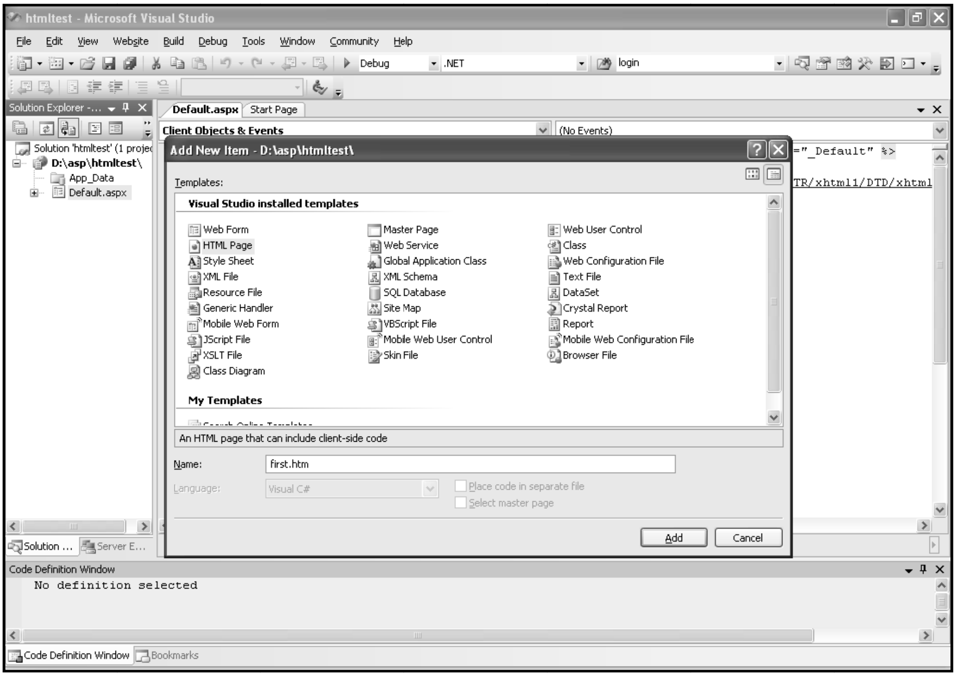


איור 2-14
יצירת יישום רשת חדש

ב. יצירת קובץ HTML

לאחר שתגדירו יישום רשת חדש, תיפתח סביבת העבודה. סביבת העבודה כוללת כמה חלונות. חלון Solution Explorer (הנמצא לרוב בצד ימין של המסך) מציג את הקבצים שהיישום מכיל ובאילו מדריכים יישמרו הקבצים. אם אינכם רואים את החלון הזה, בחרו בתפריט הראשי View ← Solution Explorer.

החלון הראשי הוא חלון העריכה. בתפריט הראשי בחרו ב- File ← New ← File.

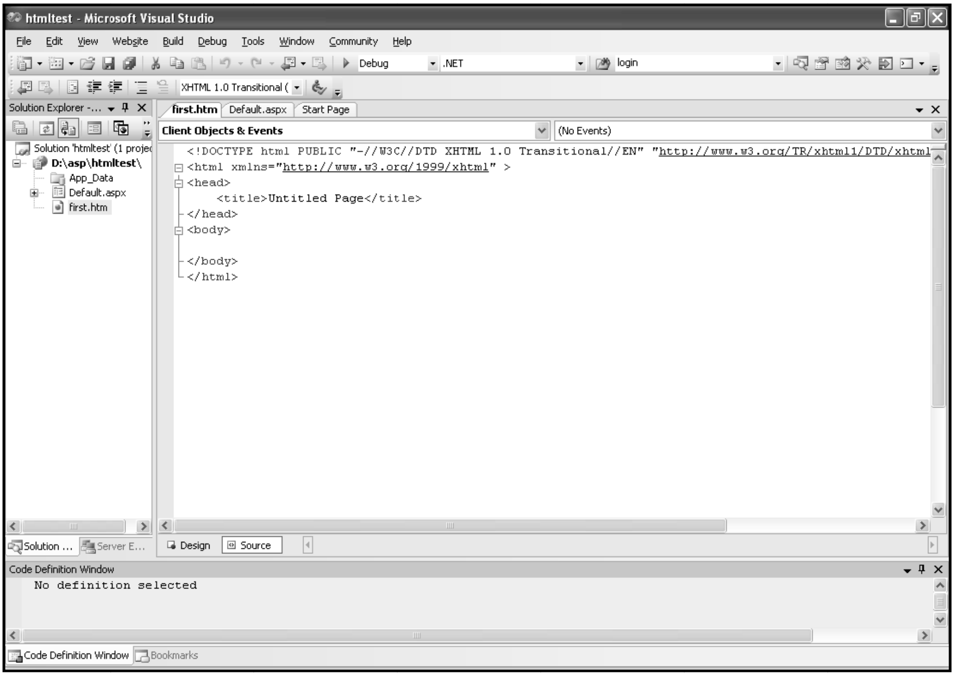


איור 2-15
פתיחת דף HTML

בחלון שייפתח:

- בחרו בקובץ מסוג HTML Page.
- שנו את שם הקובץ (בררת המחדל), ורשמו first.
- לסיום לחצו Add.

סביבת Visual Studio יוצרת דף המכיל הגדרות בסיסיות ומוכן לעריכה כדף HTML.



איור 2-16
תבנית בסיסית של מסמך HTML בסביבת Visual Studio

ג. עריכת מסמך ה-HTML

הקלידו את שמכם בגוף המסמך.

```
<body>
            רשמו את שמכם
</body>
```

לאחר השינוי שמרו את הקובץ.

סביבת Visual Studio כוללת כלים שמסייעים למתכנת בכתיבת דפי HTML. אחד מהכלים האלה הוא הצגה של הדף בזמן העריכה של התגים, המאפיינים והערכים.

ד. הצגת דף HTML

סביבת Visual Studio מספקת כמה אפשרויות להצגת מסמך מעוצב:

- אפשר ללחוץ על כפתור design, הנמצא בתחתית הדף משמאל. פעולה זו מציגה את המסמך המעוצב, אך אינה מאפשרת להשתמש בשדות קלט ובכפתורים גם כאשר הדף כולל תגים של טופס. שיטה זו מאפשרת לבחון את אופן עיצוב הדף.
- אפשר ללחוץ עם הלחצן הימני של העכבר על שם הקובץ המופיע בחלון ה- Solution Explorer. בחלון שייפתח יש לבחור באפשרות 'View in Browser'. שיטה זו מריצה את הדף מתוך השרת.

ניתן להריץ את הדף גם לא מתוך סביבת העבודה. הקלקה פעמיים על שם הקובץ במדריך שבו הקובץ מאוחסן, תריץ את הדף מתוך יישום הלקוח (הדפדפן).

נספח ב':

סיכום תגי HTML

תגים לעיצוב טקסט

		התג
כותרות		** ** **…** ** **
מעבר לשורה חדשה	break	** **
כתיבת הערה בתוך המסמך		**<!-- -->**
הוספת רווחים בין תווים	non breakable space	** **

תכונות המאפיין Style

את המאפיין Style ותכונותיו ניתן להוסיף לכל תג. נציג כאן תכונות נפוצות, אם כי קיימות תכונות נוספות למאפיין זה.

צבע הרקע	**background-color**
תמונת הרקע	**background-image**
צבע הגבול	**border-color**
סגנון הגבול	**border-style**
צבע הגופן	**color**
משפחת הגופן	**font-family**
יישור הטקסט	**text-align**
רוחב	**width**
גובה	**height**

תגים ומאפייניהם

	התג	המאפיינים	
גוף הדף	<body> </body>	כיוון הכתיבה של הדף	dir
פסקה		כיוון הכתיבה של הפסקה	dir
תיחום הטקסט		כיוון הכתיבה של המקטע	dir
תמונה	<img />	מקור התמונה הרוחב באחוזים או בפיקסלים	src="Picture Name" width

		הגובה באחוזים או בפיקסלים height הטקסט להצגה כאשר התמונה לא זמינה alt
טבלה	<table> </table>	רוחב גבול הטבלה border המרווח בין תוכן התא לגבולותיו cellpadding המרווח בין התאים cellspacing
שורה בטבלה	<tr> </tr>	יישור אופקי של תוכן התאים בשורה align יישור אנכי של תוכן התאים בשורה valign
תא כותרת תא מידע	<th> </th> <td> </td>	יישור אופקי של תוכן התא align יישור אנכי של תוכן התא valign מספר התאים בעמודה / השורה שמוזגו לתא זה colspan / rowspan
קישור	<a> 	קישור לאתר href="Page Name" קישור לדוא״ל href="mailto: Email address" קישור לעוגן בדף href="#anchorName" בנקודת העוגן נגדיר: קישור באמצעות תמונה: <img src="pic.jpg" alt="לחץ">
רשימה סדורה		כיוון הכתיבה של הרשימה dir יש להשתמש בתג להגדרת הפריטים
רשימת תבליטים		כיוון הכתיבה של הרשימה dir סוג התבליט (עיגול ריק) circle, type (ריבוע מלא) square, (עיגול מלא) disc) יש להשתמש בתג להגדרת הפריטים
טופס	<form> </form>	מספר הזיהוי של הטופס id הדף המטפל בבקשה בשרת action שיטת שליחת הנתונים method
אלמנטים בקלט	<input />	שם name זיהוי האלמנט (בדרך-כלל זהה לשם הפקד, פרט לפקדי רדיו) id ערך האלמנט value סוג האלמנט type טקסט <input type="text" /> סיסמה <input type="password" /> כפתורי רדיו <input type="radio" /> לכל כפתורי הרדיו **חייב** להיות אותו שם ומספר זיהוי ייחודי. לכפתור המסומן יתווסף המאפיין checked תיבות הסימון <input type="checkbox" /> רצוי לתת לכל תיבות הסימון את אותו השם. כפתור שליחה <input type="submit" /> כפתור ניקוי <input type="reset" /> כפתור לחצן <input type="button" />

פרק 3

תכנות שרת – יצירת דף דינמי ותכנות "חסר מצב"

3.1 יצירת דף דינמי ושליחת נתונים מהשרת ללקוח

בפרק הזה נלמד כיצד יוצר השרת דף דינמי בעקבות בקשת HTTP. נדגים זאת על-ידי בנייה של אתר של משחק מכונת מזל. משתמש (גולש) שרוצה לשחק במכונת המזל משתמש בדפדפן (שמשמש כתוכנת צד הלקוח ונכנה אותו בקיצור 'הלקוח') כדי לשלוח לשרת בקשה להגרלת מספרים. כאשר השרת מקבל פנייה מלקוח (דפדפן), הוא מגריל שלושה מספרים בתחום נתון (למשל, בין 1 ל-9), ובהתאם לתוצאות ההגרלה הוא מחשב ניקוד ושולח ללקוח תגובה מתאימה. המשחק יוצג בכמה גרסאות; בגרסה הראשונה חישוב הניקוד הוא פשוט – אם שלושת המספרים זהים, יקבל המשתמש 10 נקודות, ולא – הוא יקבל שתי נקודות. השרת מכין דף HTML המכיל פלט ששלח השרת ללקוח כתגובת HTTP. הקובץ הזה מוצג לפני המשתמש באמצעות הדפדפן שבמחשבו.

הדף המורץ בשרת צריך לבצע שתי משימות עיקריות:

א. להגריל שלושה מספרים ולחשב את הניקוד;

ב. לבנות דף דינמי שכולל את הניקוד ולשלחו כתגובת HTTP ללקוח.

כתיבת תסריט המגדיר את הפונקציונליות של אתר

לשם לימוד הנושא בחרנו בטכנולוגיית ASP של חברת מיקרוסופט. טכנולוגיה זו כוללת דפי שרת הנקראים **ASP**[1] (דפי שרת דינמיים – **Active Server Pages**).

---

[1] מיקרוסופט שיפרה את טכנולוגיית ASP , וכיום משתמשים בטכנולוגיה ASP.NET שמריצה גם דפי ASP שנכתבו בעבור הדור הקודם של הטכנולוגיה. דפי ASP שנכתבים בטכנולוגיה המתקדמת יותר נשמרים בקבצים בעלי הסיומת aspx (להבדיל מקבצים בעלי הסיומת asp שניתנה לדפים שנכתבו בגרסה הקודמת). בספר הזה הסיומות של הקבצים שבהם נשמרים דפי ASP הם .aspx

השרת יריץ דף ASP כדי לבצע את העיבודים הדרושים (משימה א׳) וכדי ליצור דף HTML שבו יוצגו תוצאות העיבוד (משימה ב׳). לאחר יצירת דף ה-HTML, ישלח אותו השרת ללקוח.

דוגמה לדף ASP – אחזור התאריך בשרת

לפני שנִפתח אתר למשחק ׳מכונת המזל׳, נציג דף ASP פשוט, שמציג לפני המשתמש את התאריך והזמן בשרת. עיינו בקוד הבא שבו רשמנו את תגי ה-HTML באותיות קטנות (זכרו כי שפת HTML אינה רגישה לגודל האותיות):

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Date ASP </title>
</head>
<body>
        <h1>
              Today's date is <%= System.DateTime.Now %>
        </h1>
   </body>
</html>
```

כתבו את הקוד ושמרו את הקובץ בשם Simple.aspx. כפי שתוכלו לראות, קובץ Simple.aspx מכיל תגי ה-HTML וקוד ASP הרשום בין התגים <% %>. הדגשנו ברקע אפור את קטע הקוד הזה. כפי שנראה בהמשך, בין התגים <% ו- %> ניתן לרשום משפטים בשפת #C ולהשתמש בעצמים מוכנים מראש המוגדרים בטכנולוגית ASP. קטע קוד זה נקרא תסריט (script) והם מורצים בצד שרת.

בדוגמה שלנו, השתמשנו **בביטוי ASP (asp expression)**. ביטוי ASP נרשם בין התג הפותח: <% לתג הסוגר: %>. סימן השוויון (=) המוצמד לתג הפותח מסמל השמה. לאחר התג <%= ניתן לרשום שם של משתנה, ביטוי לחישוב או פונקציה המחזירה ערך כלשהו. רישום זה הוא קיצור של הפעולה Response.Write שנציג בהרחבה בסעיף הבא.

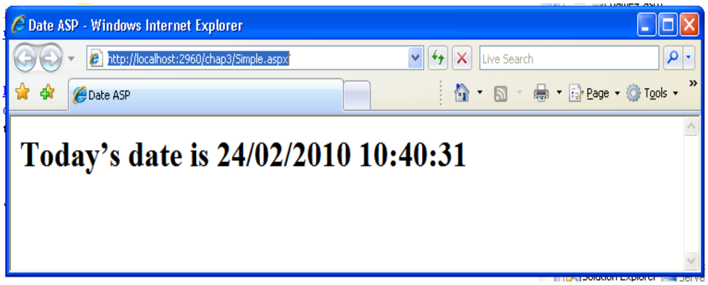
פעולה זו ממירה למחרוזת את הערכים שהתקבלו מתוצאות החישוב (למשל התאריך והיום) ומכינה תגובת HTTP הנשלחת כדף HTML מהשרת אל הלקוח.

כתיבת משפטים בשפת C# בתסריטים המתבצעים בשרת אינה שונה מכתיבת תכנית רגילה בשפה זו. אפשר להשתמש במרכיבים של השפה ולהגדיר משתנים ופונקציות.

```
<%= System.DateTime.Now %>
```

הביטוי שלעיל הוא ביטוי בשפת C# שיוצר עצם חדש מהטיפוס System.DateTime (עצם DateTime בספריית המחלקות System של C#). המחלקה DateTime נועדה כדי לייצג זמן במחשב (שכולל את התאריך והשעה). התכונה Now של מחלקה זו מייצגת זמן מקומי. ערכים אלו מוצבים בתכונה Now כאשר נוצר עצם מטיפוס המחלקה

כאשר הרצנו את הקובץ הזה בשרת (כשאנו משתמשים בסביבת Visual Studio), התקבלה התצוגה שלפניכם :



איור 3-1
מסך המציג תאריך בשרת

בשדה הכתובת של הדפדפן שבאיור 3-1 מופיע ה-URL הזה :

http://localhost:2960/chap3/Simple.aspx

הרכיבים של ה-URL הם כדלקמן :

- http:// הוא שם הפרוטוקול של יישום ה-Web.
- localhost מייצג את המחשב המקומי ; כלומר, במקרה הזה השרת רץ במחשב שבו רץ הלקוח. באופן כללי, אפשר לרשום כאן שם של מחשב כלשהו ברשת שעליו רץ השרת.

- במקרה הזה, 2960 הוא המַפתח (port) של תוכנת השרת.
- Simple.aspx הוא שמו של קובץ ה-aspx שנמצא במדריך chap3.

כאשר חוזרים על ההרצה של קובץ הזה, יופק דף HTML חדש שמכיל זמן אחר. אם השרת מורץ באזור זמן שונה מאזור הזמן[2] של הלקוח, השעה שתוצג תהיה שונה מהשעה המקומית.

כדי לראות את תוכן דף ה-HTML שנוצר על-ידי השרת, אפשר לבצע בדפדפן פקודת 'הצג מקור' (view ← source). להלן הקוד שהתקבל מהרצת הדף בשרת שבו הדגשנו באפור את המרת ערך התכונה Now לעצם DateTime ולמחרוזת:

```
<!--Simple -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
	<head>
		<title> Date ASPX </title>
	<body>
		<h1>
		Today's date is 24/02/2010 10:40:31
		</h1>
	</body>
</html>
```

שימו לב, שהביטוי בשפת C# נעלם ובמקומו מופיעים התאריך והשעה. זהו הדף שהכין השרת כתגובה לבקשה של המשתמש לדף Simple.aspx.

חזרה למשחק מכונת המזל

הגרסה הראשונה של משחק מכונת המזל תורכב משני דפים: הדף הראשון הוא דף סטטי (דף HTML) שמכיל הפניה למשחק. דף זה יציג תוכן זהה בכל בקשה. הדף השני הוא דף

---

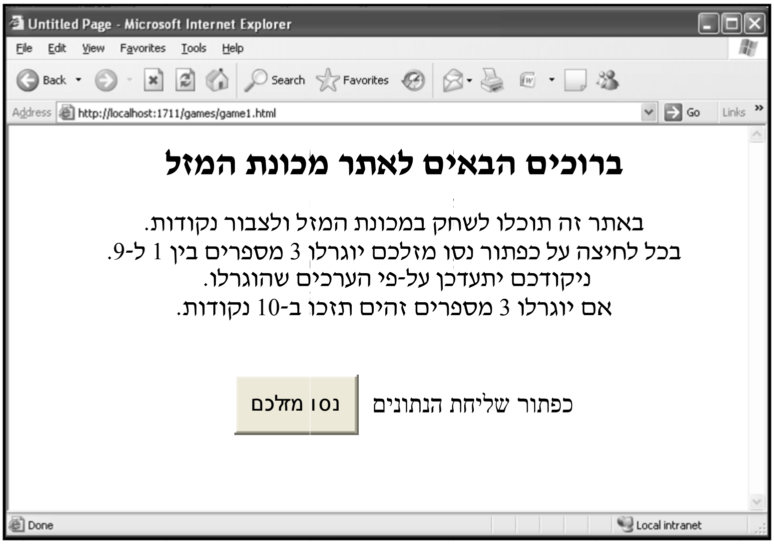
[2] אזור זמן הוא אזור על פני כדור הארץ שבין גבולותיו השעה הרשמית והתאריך הרשמי זהים.

דינמי (דף ASP המורץ בשרת) שבו נעשה חישוב של המספרים שיוגרלו, ולכן בכל בקשה של דף זה נקבל מספרים שונים (בהתאם להגרלה). היישום מורכב משני קבצים כדלקמן:

- קובץ בשם Game1.htm שמציג דף סטטי המכיל הודעת פתיחה, אשר מתאר את כללי המשחק ושמציג כפתור שליחה (submit).
- קובץ בשם Game1.aspx שמגריל מספרים, שמחשב את הניקוד ומכין דף דינמי הנשלח אל הדפדפן.

הדף הסטטי (הקובץ Game1.htm)

המשאב הראשון שהמשתמש מבקש מהשרת הוא דף ה-HTML. כתוצאה מכך מוצג החלון שבאיור 3-2:



איור 3-2
הצגה של דף המשחק מכונת המזל שנשלח מהשרת

שימו לב ל-URL שמופיע בשדה הכתובת של הדפדפן. הפעם שֵׁם המשאב הוא games/Game1.htm. הוספנו את התיקייה games לנתיב (path) שבו נמצאים דפי השרת.

שאלה למחשבה

האם על פי איור 3-2 ניתן לדעת אם הדף Game1.htm הורץ בדפדפן או בשרת?

כמפתחים של דפי שרת, נוח להריץ אותם מתוך סביבת הפיתוח (Visual Studio). לעומת זאת, משתמשים שמבקשים דף שרת כלשהו משתמשים בדפדפן ורושמים בחלון הכתובות את כתובת ה- URL המתאימה. כתרגיל, פתחו את הדפדפן ורשמו את כתובת ה- URL שהתקבלה מהרצת דף Simple.aspx בשרת.

משתמש שרוצה לנסות את מזלו במשחק לוחץ על הכפתור 'נסו מזלכם'. כאשר הוא עושה זאת, נשלחת לשרת בקשה לאחזר משאב אחר; משאב זה הוא דף השמור בשרת. אם תעיינו בקוד של הדף הסטטי (ששמו Game1.htm). שלהלן, תוכלו לראות שהערך של המאפיין action של התג form הוא "Game1.aspx" – זהו קובץ ה-aspx שבו שמור דף השרת שאותו מבקש הלקוח לאחזר והוא אמור לממש את ההגרלה וליצור דף HTML שיהיה תגובת השרת:

```
<!--- Game1.htm --------------->
<html>
<head >
   <title>Game Activity 1</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>

<body dir="rtl">
   <form id="form1" action="Game1.aspx" >
   <div>
        <h1> ברוכים הבאים לאתר מכונת המזל </h1>
      <p>

           באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות <br />
           בכל לחיצה של הכפתור 'נסו מזלכם', יוגרלו 3 מספרים בין 1 ל- 9 <br />
            ניקודכם יתעדכן על-פי הערכים שהוגרלו <br />
           אם יוגרלו 3 מספרים זהים תזכו ב-10 נקודות<br />
```

```
  </p>
            <input type="submit" value="נסו מזלכם" name="Send"/>  כפתור שליחת הנתונים
            <br /> <br />
   </div>
  </form>
</body>
</html>
```

להלן קובץ Chap3.css לעיצוב הפלט

```
body {   text-align:right;
           direction: rtl }
      h1 {color:red;
            direction:rtl}
      h3 {color:Blue;
           direction:rtl}
      p  {color:Purple;
          font-size:larger;
          text-align:right}
     div {direction: rtl}
```

הדף הדינמי (הקובץ Game1.aspx)

כפי שציינו, בשרת שמור דף ASP שמעבד את הבקשה שנשלחה אליו מהלקוח ויוצר דף דינמי המכיל את תוצאות ההגרלה של שלושה מספרים בתחום 1 עד 9 וחישוב הניקוד. כדי לבצע פעולות אלו מכיל דף ASP מספר מרכיבים (שאותם נפרט בהמשך) וביניהם קטע קוד בשפת #C.

להלן קטע הקוד בשפת #C המבצע את הפעולות הדרושות:

```
 int n1, n2, n3 , points;     // הגדרת משתנים מטיפוס שלם לאחסון המספרים שיוגרלו והנקוד
Random rnd=new Random();             // הגדרת עצם ליצירת מספר אקראי
n1 = rnd.Next(1,10);                 // הגרלת מספר בין 1 ל- 9 //
 n2 = rnd.Next(1,10);
```

```
n3 = rnd.Next(1,10);
if (n1==n2 && n2==n3)              // בדיקה אם שלושת המספרים שווים
      points = 10;                 // אם כן, הניקוד 10
else
      points = 2;                  // אחרת – הניקוד 2
```

כדי להגריל מספרים אקראיים השתמשנו בעצם מהטיפוס Random ובפעולה ()Next המקבלת שני פרמטרים שמגדירים את תחום המספרים האקראיים. המספרים האקראיים שנוצרים הם גדולים מן הפרמטר הראשון או שווים לו וקטנים מהפרמטר השני. כך לדוגמה, כדי לייצר מספר אקראי בין 1 ל-9 נרשום: (Next(1,10. הפעולה מחזירה מספר מטיפוס שלם.

לכן, כדי להגריל מספר בין 1 ל-9 ולאחסנו במשתנה n1 רשמנו:

```
int n1 = rnd.Next(1,10);           // מגריל מספר בין 1 ל- 9
```

היכן נרשום את הקטע של קוד התכנות שהשרת זקוק לו כדי לבצע את ההגרלה וכדי לחשב את הניקוד?

דף ASP מכיל כמה מרכיבים ובהם: טקסט (literal text), תגי HTML, הנחיות וקוד ASP (שחלקן נציג בספר הזה). לכל אחד מהמרכיבים ישנם סימנים שבעזרתם ניתן לזהות את סוג המרכיב. ראינו כבר שניתן לרשום קוד לשליחת תגובה ללקוח (במקרה כזה השתמשנו בתגים <%= %>), אבל כעת ברצוננו להגדיר משתנים, ובהמשך גם פונקציות, שאותם נזמן כביטויי ASP.

הגדרה של משתנים או של פונקציות בדף ASP נעשית בקטע שתָחום על-ידי התגים שלהלן:

```
<script runat="server">
            רשמו כאן הגדרות של משתנים ופונקציות
</script>
```

אחת הדרכים להגדיר תסריט היא להשתמש בתגים <script> ו-</script> של שפת הסימון HTML. לתג זה ניתן להגדיר מאפיין runat מציין שהקוד הזה מורץ בשרת. בתסריטים מסוג זה נשתמש כדי להגדיר משתנים ופונקציות שתחום ההכרה שלהם הוא בכל הדף. תסריטים אלו יכתבו לפני התג הפותח <html>.

לדוגמה, את קטע הקוד בשפת C# לחישוב הניקוד נרשום כך:

```
<script runat="server">
  int n1, n2, n3 , points;   //  הגדרת משתנים מטיפוס שלם לאחסון המספרים שיוגרלו והניקוד
 public void Page_Load()
 {
        Random rnd=new Random();            // הגדרת עצם ליצירת מספר אקראי
         n1 = rnd.Next(1,10);               // הגרלת מספרים אקראיים
         n2 = rnd.Next(1,10);
         n3 = rnd.Next(1,10);
                                            // חישוב הניקוד
        if ((n1 == n2) && (n2==n3))
          points = 10;                      //10 אם שלושת המספרים זהים – הניקוד הוא
        else
          points=2;                         // 2 אחרת – הניקוד הוא
  }
</script>
```

האירוע Page_Load

שימו לב, המשפטים להגרלת שלושה מספרים ולחישוב הניקוד נרשמו בתוך פעולה בשם Page_Load. למעשה, זהו **אירוע (event)** המתרחש בכל פעם שמתקבלת בקשה לטעינת הדף – בעקבות לחיצה על הכפתור 'נסו מזלכם', או בעת בקשה ראשונית לטעינת הדף. כאשר מתרחש האירוע הזה, המערכת מבצעת את הקוד הרשום בתוך האירוע Page_Load.

מהו אירוע? תכנית מורצת צריכה להגיב לסדרה של אירועים – את קצתם יוזמת התכנית (למשל, היא מבקשת מן המשתמש להכניס נתונים) ואת האחרים יוזם המשתמש (למשל,

המשתמש מקליק בעכבר על צלמית של הדפדפן). אירוע שיוזם המשתמש נקרא 'התרחשות חיצונית'. לדוגמה, אם המשתמש הקליק בעכבר על צלמית של דפדפן, כתגובה ייפתח חלון של הדפדפן. תוכניות שונות, כגון מערכת הפעלה, מכילות קטעי קוד שמותאמים לכל אחד מהאירועים וכן נעזרות במנגנון שמזהה אירוע מסוים שיזם המשתמש ומפעיל את קטע הקוד המתאים לטיפול באירוע זה. כתיבת תכנית המבוססת על העיקרון שלפיו בעקבות התרחשות חיצונית במערכת מתבצע קטע של תכנית שהוגדר מראש, נקראת **תכנות מונחה אירועים** (**Event-driven programming**).

תכנות מונחה אירועים מבוסס על הגישה, שבה בונים תכנית המכילה רכיבים ה"ממתינים" לקבל אות. האות נקרא מאורע (event) והוא מתקבל כאשר מתרחש אירוע מסוים במערכת אליה קשוב היישום. לדוגמה, יישום המכיל ממשק משתמש גרפי (Graphical unit interface) המציג חלון ועליו רכיבים שונים, כגון: לחצן, תיבות טקסט, תיבות בחירה, רשימות ועוד. בנוסף לעיצוב ובנית החלון שמכיל את הרכיבים, מפתחי היישום כותבים קטעי תכנות לביצוע פעולות מסוימות שיתרחשו רק כאשר המשתמש ישתמש ברכיב כלשהו. לדוגמה: אם המשתמש לחץ על כפתור "משחק", יופעל קטע קוד המאפשר למשתמש לשחק במשחק. דוגמה נוספת, כאשר המשתמש מעביר את העכבר באזור מסוים בחלון, יופעל קטע קוד המציג חץ המתאר את תנועת העכבר. אירועים אלו גורמים להפעלת מנגנון הנקרא מתפעל אירועים, שתפקידו לזמן את הפעולה המתאימה לרכיב הנבחר. גישה זו מאפשרת בניית יישום אינטראקטיבי בצורה פשוטה ומודולארית הקלה לתחזוקה ולהרחבה.

כפי שציינו, השרת ממתין לבקשת HTTP (התרחשות חיצונית), וברגע שהתקבלה בקשה כזאת, הוא מריץ את דף ASP המתאים ומבצע את המשפטים הרשומים באירוע Page_Load.

כעת, נבנה את דף ASP המורכב משני חלקים עיקריים: החלק המכיל את ההגדרות והפונקציות והחלק המכיל את תגי ה-HTML שבו מכין השרת את התגובה ללקוח.

החלק הראשון כולל הנחיות והגדרה של משתנים ופונקציות בתוך תסריט התחום בין התגים <script> ו - </script>. תוכלו לראות שיש בתחילת הקובץ שתי הנחיות. הנחיות

בדף ASP כוללות מידע לשרת ומתייחסות לאופן פעולתו. הנחיה ראשונה מגדירה את שפת התכנות שבה אנו משתמשים לכתיבת קטעי קוד התכנות.

```
<%@ Page Language="C#" %>
```

הנחיה שנייה מגדירה לדפדפן את התקן של שפת HTML שבה אנו משתמשים:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

לאחר מכן כתבנו שורה שמגדירה את המשתנים n1, n2, n3 ו-points. משתנים אלה מוגדרים כאן כך שיוכרו בדף כולו. לאחר שורה זו כתבנו את התסריט המכיל הגדרה של האירוע Page_Load שיתבצע מיד עם טעינת הדף. שימו לב, את התסריט התחום על ידי התגים <script> ו- </script> יש לכתוב לפני התג html.

החלק השני המופיע כרגיל בין התגים: <html> ו- </html> מכיל תגי HTML שמתארים את האופן שבו הדפדפן יציג את הפלט למשתמש. בדוגמה שלנו הפלט מורכב מהמספרים שהוגרלו והניקוד שחושב. נתונים אלו יתקבלו מחישוב ביטויי ASP, שנכתבו בין התגים <% =%>. בין תגים אלו אנו יכולים לרשום שמות משתנים או לזמן פונקציה שהגדרנו בתסריט התחום בין התג <script> והתג </script>.

בדוגמה שלנו כדי להציג את הניקוד המתקבל, נרשום בין תגי ה-HTML את הביטוי הזה:

```
הניקוד הוא <%= points %> <br />
```

וכדי לרשום את המספרים שהוגרלו נשבץ את התסריט שלהלן. שימו לב כי לכל מספר רשמנו ביטוי משלו:

```
המספרים שהגרלת הם <%= n1 %>, <%= n2 %>, <%= n3 %>
```

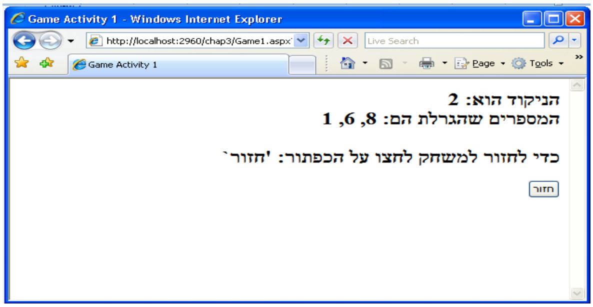
להלן הקוד המלא של הדף Game1.aspx. כדי להדגיש את חלקי הדף, הוספנו רקע אפור לחלק הראשון של הדף.

```
<!-- Game1.aspx -->
<!-- הנחיות המגדירות את שפת התכנות והסטנדרטים של HTML -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<!-- תסריט לבחירת שלושה מספרים וחישוב הניקוד -->
<script runat="server">
 int n1, n2, n3 , points;               // הגדרת משתנים מטיפוס שלם לאחסון המספרים שיוגרלו
והניקוד  public void Page_Load()
  {
        Random rnd=new Random();           // הגדרת עצם ליצירת מספר אקראי
        n1 = rnd.Next(1,10);               // הגרלת מספרים אקראיים בין 1 ל-9
        n2 = rnd.Next(1,10);
        n3 = rnd.Next(1,10);
                                           // חישוב הניקוד
      if ((n1 == n2) && (n2 == n3))        // בדיקה אם שלושת המספרים זהים
        points=10;
      else
        points=2;
  }
</script>

<!-- הוראות ותגי עיצוב להצגת הפלט ללקוח -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
  <title>Game Activity 1</title>
  <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
  </head>
<body>
  <form method="get" action="Game1.htm">
  <div style="text-align:right">
    <h2> הניקוד הוא: <%= points %><br />
      המספרים שהגרלת הם <%=n1 %>, <%=n2 %>, <%=n3 %>
      <br /><br />
      כדי לחזור למשחק לחצו על הכפתור: 'חזור'
       </h2>
         <input type="submit" value="חזור" />
     </div>
  </form>
</body>
```

</html>

להלן החלון שמתקבל כתוצאה מהפעלת ASP :



איור 3-3
דף המשחק של אתר מכונת המזל שנשלח מהשרת

במקרה זה, למשתמש לא היה מזל, ולכן הוא זכה בשתי נקודות בלבד. ואולם הוא יכול לחזור למשחק ולנסות את מזלו שוב על-ידי לחיצה על הכפתור ' חזור'.

הוספת הערות לדף ASP

שיבצנו בדף Game1.aspx גם הערות. הערה כזו נרשמת בין התגים האלה :

<!-- הערה של aspx -->

הערות בדף ASP אינן חלק ממשפטי הביצוע. הערות הן חלק מהתיעוד של הדף ומיועדות בעיקר למפתחים של התכנית. ניתן גם לרשום הערות בהמשך למשפט #C על-ידי שימוש בתווים //, לדוגמה:

```
n1 = rnd.Next(1,10);                    // הגרלת מספר אקראי בין 1 ל-9
```

שאלה למחשבה

עיינו בקוד של Game1.aspx ומצאו את הסיבה מדוע כאשר המשתמש לוחץ על הכפתור 'חזור' הוא מוחזר למשחק 'מכונת המזל'.

שאלה למחשבה

הסבירו מה היה קורה לדעתכם אילו היינו מגדירים את המשתנים n1, n2, n3 בתוך קטע הקוד של האירוע Page_Load.

אם נגדיר את המשתנים הללו כחלק מקטע הקוד של האירוע Page_Load, הם יהיו משתנים שלא יהיו מוכרים מחוץ לקטע הקוד הזה. כאשר רוצים להשתמש במשתנים אלה גם

בקטעי קוד נוספים (למשל כדי לשלוח פלט ללקוח), יש להגדיר אותם מחוץ לאירוע Page_Load. תחום ההכרה של המשתנים קובע אילו הוראות יכולות להשתמש במשתנים. תחום ההכרה של משתנה שהוגדר בתוך פונקציה או אירוע, הוא מקומי וניתן לגשת אליו רק מתוך הפונקציה. תחום ההכרה של משתנה שהוגדר מחוץ לפונקציה או האירוע Page_Load הוא כללי (גלובלי) והוא נגיש לכל משפט בתסריט ולכל ביטוי ASP מכל מקום בדף.

להלן קובץ ה-HTML שנוצר באופן דינמי על-ידי השרת כתוצאה מעיבוד קובץ ה- aspx (זכרו כי כדי לראות את הקובץ הזה עליכם להשתמש בפקודה 'הצג מקור' של הדפדפן).

```
<!--Game1.aspx     -->
<!--HTMLהנחיות המגדירות את שפת התכנות והסטנדרטים של -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
                                    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!--  תסריט לבחירת שלושה מספרים וחישוב הניקוד  -->

<!--  הוראות ותגי עיצוב להצגת הפלט ללקוח  -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
   <title>Game Activity 1</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
   </head>
<body>
   <form method="get" action="Game1.htm">
   <div style="text-align:right">
     <h2>
        הניקוד הוא: 2 <br />
        המספרים שהגרלת הם: 8, 6, 1
        <br /><br />
        כדי לחזור למשחק לחצו על הכפתור: 'חזור`
     </h2>
        <input type="submit" value="חזור" />
```

```
    </div>
  </form>
</body>
</html>
```

תוכלו לראות כי אין זכר להגדרות או לביטויי ASP; בקטע המודגש באפור ניתן לראות כי הביטויים הוחלפו על-ידי הערכים שחושבו בהרצה מסוימת שבה הוגרלו המספרים 1, 6 ו-8 וכי כל הקוד שהיה בין התגים <script> ... </script> נעלם.

מה קורה מאחורי הקלעים?

כפי שהסברנו, כאשר השרת מקבל את הבקשה הראשונה לדף ASP, הוא מהדר את התסריטים בשפת C# לקוד שניתן להרצה. בבקשות הבאות של אותו דף אין כבר צורך לתרגם ולהדר את הדף. כאשר מגדירים משתנים ופונקציות באמצעות התגים <script> ו-</script>, מגדירים שדות ופעולות שבהן ניתן להשתמש בחלק העיצובי, כדי להכין תגובה לבקשת הלקוח. השדות והפעולות מוכרים לכל הקוד ולכן אפשר לגשת אליהם מכל מקום בדף.

שימוש בפונקציות

בפתרון שהצגנו כתבנו בתוך האירוע Page_Load את המשפטים לחישוב מספרים אקראיים ואת הניקוד. ברצוננו להפריד בין שתי הפעולות ולרשום בתוך האירוע Page_Load רק את הפעולות שרצוי לבצע בזמן שהדף נטען. בדוגמה שלנו נרשום את המשפטים לאתחול המשתנים n1, n2 ו-n3 במספרים אקראיים. ואילו את החישוב של הניקוד נבצע בפונקציה נפרדת בשם CalculatePoints, שאותה נוכל לזמן בהתאם לצורך. כדי להציג את הניקוד, נשתמש בהמשך בביטוי ASP שמזמן את הפונקציה CalculatePoints. לשימוש בפונקציות יש יתרונות רבים. בין השאר הוא יאפשר לנו לבנות את התכנית בצורה מובנת, ולהרחיב ולשנות את התכנית בקלות.

להלן דף ASP בשם Game1a.aspx שכולל שינויים אלה. השינויים מצוינים בקטעים האפורים. שינינו את האירוע Page_Load והוספנו הגדרה של הפונקציה CalculatePoints.

כמו כן, שנינו את הביצוע של התסריט שנמצא בחלק השני של הדף בו מעצבים את התגובה ללקוח. במקום לשלוח את ערכו של המשתנה points, כתבנו תסריט המזמן את הפונקציה:

```
<%= CalculatePoints()%>
```

להלן התכנית המלאה:

```
<!—Game1a.aspx     -->
<!—HTMLהנחיות המגדירות את שפת התכנות והסטנדרטים של  -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!--  תסריט לבחירת שלושה מספרים ולחישוב הניקוד  -->
<script runat="server">
  int n1, n2, n3;                          // הגדרת משתנים מטיפוס שלם לאחסון הניקוד
  public void Page_Load()
  {
         Random rnd=new Random();          // הגדרת עצם ליצירת מספר אקראי
         n1 = rnd.Next(1,10);              // הגרלת מספרים אקראיים
         n2 = rnd.Next(1,10);
         n3 = rnd.Next(1,10);
  }
<!-- חישוב הניקוד -->
  int CalculatePoints()                    // הגדרת פונקציה לחישוב הניקוד
  {
       if ((n1 == n2) && (n2 == n3))       // בדיקה אם שלושת המספרים זהים
         return 10;
     else
         return 2;                         // ניקוד בסיסי הוא 2 נקודות
  }
</script>

<!--  הוראות ותגי עיצוב להצגת הפלט ללקוח  -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
   <title>Game Activity 1</title>
```

```
    <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
  </head>
<body >
<form method="get" action="Game1a.aspx">
    <div style="text-align: right">
     <h3>
      הניקוד הוא      <%= CalculatePoints()%>
      <br /><br />
      המספרים שהגרלת הם <%=n1 %>,<%=n2 %>,<%=n3 %>
      <br /><br />
      כדי לחזור למשחק לחצו על הכפתור: חזור
     </h3>
     <input type="submit" value="חזור" />
    </div>
   </form>
</body>
</html>
```

זכרו כי עליכם לשנות כעת גם את קובץ HTML כך שיכלול הפניה לדף Game1a.aspx, כלומר נשנה את התג באופן הזה:

```
<form id="form1" action="Game1a.aspx" runat="server">
```

שאלה למחשבה

מה לדעתכם יכלול קובץ HTML שיתקבל בדפדפן כתגובה מבקשת דף Game1a.aspx?

לסיכום, עד כה הכרנו שלושה סוגים של תגים שבתוכם ניתן לכתוב קוד C# בתוך דף ASP:

- **הגדרות**: את ההגדרות של הפונקציות והמשתנים רושמים בין התגים <script> ו- </script>. נהוג לרשום את ההגדרות האלה בחלקו העליון של דף ה-ASP. פונקציות ומשתנים שמוגדרים באופן הזה הם שדות של העצם שהשרת יוצר כתוצאה מתרגום הדף והם מוכרים בדף כולו.

- **תסריט ASP** : בתסריטים שמכילים משפטי ביצוע וזימון של פונקציות שהוגדרו קודם לכן ירשמו בין התגים <% ...%> . בתסריטים מסוג זה נשתמש בחלק העיצובי שבו אנו מכינים את תגובת השרת. ניתן להגדיר משתנים ופונקציות בתוך תסריטים מסוג זה והם יהיו מוכרים רק באזור התחום על ידי התגים <% ו- %>.
- **ביטויי ASP** : אלה ביטויים בשפת #C שנכתבים בין התגים <%= %> . ביטויים כאלה יופיעו בחלק התחתון של דף ה-ASP, בתוך חלק העיצוב של הדף. הערך של ביטויים אלה מוכנס לתוך קובץ הפלט (HTML) שיוצר השרת. התגים <%= %> הם רישום מקוצר של הפעולה ()Response.Write שמבצעת חישוב ומחזירה תגובת HTTP.

שאלה 3.1

א. נסו להשתמש במשחק 'מכונת המזל', האם אתם מצליחים לקבל 10 נקודות?

ב. שנו את המשחק כך שאפשר יהיה לשנות בקלות את התחום של המספרים. הגדירו משתנה שמייצג את הגבול העליון של התחום והשתמשו בו בתכנית. קבעו את ערכו של המשתנה הזה ל-3 (כך המשחק יגריל שלושה מספרים בתחום 1..3). האם עכשיו אתם מצליחים לזכות ב-10 נקודות?

בדקו – האם שיניתם גם את ההודעה שמסבירה את כללי המשחק כך שתחום המספרים יוצג נכון?

3.2 העברת נתונים בין שרת לבין לקוח

גרסה 2 של משחק 'מכונת המזל'

בגרסה 2 נכניס כמה שינויים כדלקמן: המשתמש יתבקש להזין את שמו. אם הוא יעשה כך, הוא יקבל כבונוס תוספת של ארבע נקודות. נוסף על כך, כדי להקטין את מספר הקבצים ביישום, נהוג לאחד את דף ה-HTML עם דף ה- ASP. בגרסה זו נשתמש בקובץ aspx בלבד שיכלול את חלק התצוגה (המעוצב בעזרת תגי HTML) ואת החלק הלוגי שימומש על-ידי פונקציות.

כדי שנוכל לממש את השינויים האלה, עלינו לדון בנושאים שלהלן:

1. שימוש ב**עצמים מוגדרים מראש** (**predefined objects**) – בשלב הראשון נכיר שני עצמים אלה: העצם Response, שמייצג את תגובת השרת לבקשת הלקוח והעצם Request, שמייצג את בקשת הלקוח
2. כתיבת תסריטי ASP
3. שיטות לשליחת נתונים מהלקוח לשרת

להלן נדון בנושאים אלה.

שליחת פלט אל הלקוח – הפעולה Response.Write

בדפי ASP אפשר להשתמש בכמה עצמים שהמערכת מגדירה מראש. כלומר, המתכנת יכול להשתמש בעצמים אלה מבלי שיהיה עליו להגדירם או ליצור אותם. העצם הראשון מסוג זה שנכיר הוא העצם **Response**. עצם זה מטפל בהכנה של תגובת השרת, שנשלחת אל הלקוח במבנה שמגדיר פרוטוקול ה-HTTP (תגובת HTTP). העצם הזה כולל כמה תכונות ופעולות שאת קצתן נציג בפרק הזה. הפעולה העיקרית של עצם זה שבה נשתמש היא Write. פעולה זו מקבלת פרמטר מהטיפוס מחרוזת וכותבת אותו לדף שהשרת יוצר. למעשה, השתמשנו בפעולה דומה כאשר כתבנו ביטוי ASP בין התגים <%= %> . בתגים אלה יכולנו לרשום רק ביטויים המפיקים את הפלט הדרוש אך לא הודעות לעריכת הפלט. הפעולה Write מאפשרת לשלב פלט והודעות עריכה.

כדי להשתמש בפעולה Write של העצם Response, נרשום את שם העצם ואת שם הפעולה וביניהם - נקודה:

```
Response.Write( string );
```

string מייצג כאן פרמטר מטיפוס מחרוזת שיש להעביר לפעולה. המחרוזת יכולה להכיל טקסט, הוראות HTML וערכי משתנים.

לדוגמה:

```
Response.Write("נקודות " + CalculatePoints() + " בהגרלה האחרונה זכית ב-");
```

האופרטור '+' משמש בשפת C# גם לשרשור מחרוזות (נוסף על השימוש המוכר – חיבור מספרים). אפשר ליצור מחרוזת הכוללת נתונים מספריים, טקסט ותגי HTML. למשל, כאשר המשתמש הגריל שלושה מספרים שווים, הערך שיוחזר על-ידי הפונקציה

CalculatePoints הוא 10, ולכן הדפדפן יציג את ההודעה הזאת: "בהגרלה האחרונה זכית ב-10 נקודות".

כדי להציג את הניקוד שקיבל המשתמש, אפשר לרשום את הפעולה Response.Write בתסריט בצורה הזאת:

```
<%
Response.Write(name+" בהגרלה האחרונה זכית ב "+points+" נקודות");
%>
```

התסריט הזה ישולב בקטע שבו אנו מעצבים את הפלט כדף HTML בין התגים <body> ו- </body>.

לאחר שהפלט מוצג, הסמן נשאר באותה שורה שבה הוצג הפלט. כדי לעבור לשורה חדשה, נשרשר את התג
 נשרשר למחרוזת שמקבלת הפעולה Response.Write, לדוגמה:

```
Response.Write ("בהגרלה האחרונה זכית ב-" + CalculatePoints()+" נקודות" + "<br />");
```

שאלה 3.2

שנו את הדף שמורץ בשרת כך שהניקוד יחושב על-פי כללי הניקוד שהוגדרו בפרק 2 (סעיף 2.5) במשימה ב'. את עיצובו של דף זה כתבתם ושמרתם בקובץ Game0a.htm. הוסיפו לקובץ את התסריטים הנדרשים ושמרו את הדף בקובץ בשם Game1b.aspx.

שאלה 3.3

שנו את הדף של משחק 'מכונת המזל' שמורץ בשרת, והוסיפו לשיטת החישוב של הניקוד את התנאי הזה: אם שלושת המספרים שיוגרלו יהיו זהים וזוגיים, יינתנו 12 נקודות.

שליחה של הנתונים מהלקוח לשרת

בפעילות הקודמת למדנו כיצד השרת שולח מידע ללקוח, ואולם יישומי השרת- לקוח מאפשרים גם העברת נתונים מהלקוח לשרת. בגרסה הבאה של המשחק, המשתמש

מתבקש להקליד את שמו בטופס המוצג בדפדפן. לחיצה על הכפתור 'נסו מזלכם' תגרום, בין היתר, להעברת שם המשתמש לשרת. השרת יפעיל את דף ה- ASP המתאים ויבצע את אותן פעולות שביצע בפעילות הקודמת (כלומר: יגריל שלושה מספרים ויחשב את הניקוד בהתאם לכללי המשחק). נוסף על כך, השרת יבדוק אם הלקוח (הדפדפן) אכן שלח שם ולא מחרוזת ריקה. אם הלקוח שלח שם (ולא מחרוזת ריקה), המשתמש יזכה בבונוס של שתי נקודות. לסיום, השרת יכין תגובה המכילה את שם המשתמש ואת הניקוד שלו.

כדי לממש זאת, עלינו לשנות את דף ה- ASP – להוסיף לו קטעי קוד שבהם נשתמש בעצמים ופעולות כדי לקרוא את הנתונים שהגיעו מהלקוח ולעבד אותם. כמו-כן, עלינו לשנות את הדף שמוצג בצד של המשתמש – להוסיף תגי HTML להצגת טופס המכיל שדות קלט שבאמצעותו הנתון שהמשתמש הקליד בטופס יישלח לדף השרת.

כדי לשלוח נתונים מהלקוח לשרת, נשתמש בטופס המוגדר באמצעות התג form. להלן דוגמה לטופס המכיל הגדרות של שתי תיבות טקסט וכפתור לשליחת הנתונים לשרת. כמו-כן, מוגדרים להלן מאפיינים נוספים שמתייחסים לשיטת המשלוח של הנתונים ולדף שאליו נשלחים הנתונים (ראו הסבר בהמשך).

```
<form id="form1" action="Game1.aspx " method="get" runat="server">
    שם פרטי <input type="text" name="userFirstName" id="userFirstName" size="20" /> <br />
    שם משפחה <input type="text" name="userLastName" id=" userLastName" size="20" /> <br />
    <input type="submit" value="שלח" name="send"/> <br /> <br />
</form>
```

שיטות לשליחת נתונים מהלקוח לשרת

כדי להגדיר את דרך שליחת הנתונים שהמשתמש מילא בטופס, משתמשים במאפיין **method** הכלול בתג <form> ובאחד משני הערכים האלה: **get** או **post**.

```
<form id="form1" action="Game.aspx" method="get" runat="server">
```

או

```
<form id="form1" action="Game.aspx" method="post" runat="server">
```

כאמור, השיטה של שליחת הנתונים משפיעה על המבנה של בקשת ה-HTTP שנשלחת מהלקוח לשרת, ועל הפעולות שבאמצעותן השרת מפענח את הבקשה ומטפל בה. ההבדל בין שתי השיטות לשליחת הנתונים הוא שבשיטה get, הדפדפן שולח מחרוזת של נתונים המשורשרת לכתובת ה-URL, ואילו בשיטה post יופיעו הנתונים בגוף ההודעה שנשלחת אל השרת. נפרט להלן את כל אחת מהשיטות האלה.

א. שליחת נתונים באמצעות השיטה get

שיטה זו משמשת כבררת המחדל למשלוח של נתוני הטופס. בשיטה זו נתוני הטופס משורשרים לכתובת של דף השרת.

בדוגמה שלהלן, התג form בדף html (בקטע המודגש ברקע אפור) כולל את המאפיין method (שמגדיר את שיטת שליחת הנתונים) אשר מקבל את הערך get :

```
<form id="form1" action="Game1.aspx" method="get" runat="server">
  שם פרטי <input type="text" name="userFirstName" id=" userFirstName" size="20" /> <br />
  שם משפחה <input type="text" name="userLastName"  id=" userLastName" size="20" /><br />
  <input type="submit" value="try" name="send"/> <br /> <br />
</form>
```

לאחר לחיצה על הכפתור submit, נקבל מחרוזת המכילה את כתובת ה-URL שאליה משורשר התו '?' ואחריו משורשרים הנתונים. לדוגמה :

http://localhost:1231/WebSite1/Game1.aspx?
userFirstName=Lior&userLastName=Levy&send=try

מחרוזת הנתונים שנשלחת לשרת בנויה משני חלקים, כמפורט להלן :

- החלק הראשון של המחרוזת מכיל את כתובת ה-URL של הדף המבוקש במבנה הזה :

http://<server>:<port>/<path>/<page name>

בדוגמה זו רשום ה-URL שלהלן : http://localhost:1231/WebSite1/Game1.aspx

- החלק השני, לאחר התו '?', מורכב מזוגות של שמות השדות והערכים במבנה הזה :

שם השדה = ערך

בטופס הוגדרו שמות השדות כמאפיינים של < input> , והערכים הם הערכים שהקליד המשתמש. לדוגמה, אם המשתמש הקליד בטופס את השם הפרטי Lior ואת שם המשפחה Levy, ולסיום לחץ על הכפתור send (שאליו מוצמד הערך try), תישלח מחרוזת המורכבת משלושה שדות שמופרדים על-ידי התו & :

userFirstName=Lior // השם הפרטי
userLastName=Lev // שם המשפחה
send=try // כפתור השליחה

שימוש בשיטה זו במנועי חיפוש מאפשר לשמור יחד את כתובת הדף ואת נתוני שדות החיפוש ולאחזר אותם בהמשך. לדוגמה, נפעיל את מנוע החיפוש google.com באנגלית. נרשום באנגלית את המילים "asp tutorial", ולאחר מכן נלחץ על הכפתור Google Search. הכתובת שתתקבל בדפדפן תהיה :

http://www.google.com/search?hl=en&q=asp+tutorial

החלק הראשון של הכתובת הוא כתובת המחשב המריץ את מנוע החיפוש google :

http://www.google.com

החלק השני של הכתובת כולל את הפרטים האלה : המילה search – שמגדירה את הפעולה שבוצעה; התו '?'; הגדרת שפת החיפוש – השפה hl=en; מילות החיפוש שהוקלדו – q=asp+tutorial. הסימן '+' מחליף את תו הרווח בין מילים, והסימן '&' הוא סימן לשרשור של מחרוזות המכילות את מאפיין החיפוש ואת הערך שהוצמד לו. כלומר, המחרוזת שנשלחה מהלקוח לשרת כללה את הפעולה הדרושה, את השפה ואת מילות החיפוש.

כאשר נחפש את אותה המחרוזת במנוע החיפוש בעברית, נקבל את הכתובת הזאת :

http://www.google.co.il/search?hl=iw&q=asp+tutorial&btnG=%D7%97%D7%99%D7%A4
%D7%95%D7%A9+%D7%91-Google&meta

הכתובת מכילה תווים רבים שקצתם מטפלים בייצוג של מילות החיפוש בעברית, ולמרות זאת, ניתן לזהות את המרכיבים שציינו קודם לכן : את כתובת האתר, את הפעולה, את השפה ואת מילות החיפוש.

לשימוש בשיטה get יש כמה חסרונות :

א. השיטה אינה מאפשרת לאבטח את הנתונים שנשלחו ; המשתמש – או כל מי שיש לו גישה לנתונים – יכול לצפות בנתונים שהועברו, כלומר הנתונים חשופים ברשת וכל אחד יכול לשאוב מהם מידע. ברור כי בדרך זו לא נרצה להעביר מידע רגיש כמו סיסמאות ומספרי כרטיסי אשראי. כמו כן לא נרצה כי נתונים כמו שֵׁם וכתובת יהיו נגישים לכל משתמש.

ב. כיוון שהנתונים המועברים בטופס משורשרים לכתובת של דף השרת, האורך של מחרוזת הנתונים מוגבל לכ-1,000 תווים. שיטה זו אינה מתאימה למקרה שבו צריך להעביר כמות גדולה יותר של נתונים מהלקוח לשרת.

ב. שליחת הנתונים באמצעות השיטה post

כדי להשתמש בשיטה post נשנה את הערך שהצמדנו למאפיין method. להלן השינוי שיש להכניס בדף game1.htm שהצגנו בסעיף הקודם :

```
<form id="form1" action="Game1.aspx" method="post" runat="server">
  שם פרטי: <input type="text" name="userFirstName" id=" userFirstName" size="20" /> <br />
  שם משפחה: <input type="text" name="userLastName" id=" userFirstName" size="20" /> <br />
   <input type="submit" value="נסו מזלכם" name="send"/> <br /> <br />
</form>
```

גם בשיטה post, כמו בשיטה get, הנתונים שהוזנו לטופס נשלחים לשרת, לאחר שלחצנו על הכפתור send. גם כאן הנתונים נשלחים כסדרה של זוגות המורכבים משם השדה והערך שהזין עבורו המשתמש בטופס. זוגות אלה משורשרים עם התו & כמחרוזת שצורתה :

userFirstName=*value1*&userLastNAme=*value2*&send=*value3*

ואולם, בניגוד לשיטה get, בשיטה זו המחרוזת המשורשרת מוכנסת לגוף הודעת ה-HTTP שנשלחת מהלקוח לשרת, ואי-אפשר לראותה בכתובת ה-URL. מכאן יתרונותיה להלן :

א. היא בטוחה יותר להעברת נתונים חסויים.

ב. אין מגבלה על כמות התווים שניתן להעביר מהלקוח לשרת.

קריאת הנתונים ששלח הלקוח

לאחר שהמשתמש מילא את הטופס ולחץ על הכפתור send, הנתונים מועברים מהלקוח לשרת באחד מהמבנים שתיארנו לעיל, בהתאם לשיטת השליחה. כדי לאחזר את הנתונים ששלח הלקוח, משתמשת תוכנת השרת בעצם בשם **Request**. זהו עצם המוגדר מראש אשר

מייצג את הבקשה ששולח הלקוח לשרת. העצם מאחסן את הנתונים ששולח הלקוח לשרת באמצעות אוסף זוגות במבנה הזה:
<ערך, שם שדה>

כדי לאחזר את הנתונים ששלח הלקוח, ניתן להשתמש באחד משני סוגי האוספים של העצם Request:

א. אם הלקוח שלח את הנתונים בשיטה get, נשתמש באוסף Request.QueryString.

ב. אם הלקוח שלח את הנתונים בשיטה post, נשתמש באוסף Request.Form.

לדוגמה, כדי לקרוא את השם הפרטי ושם המשפחה ששלח הלקוח בשיטה get נרשום:

```
string  userFirstName = Request.QueryString["userFirstName"];
string  userLastName = Request.QueryString["userLastName"];
```

באופן דומה, כדי לקרוא את השם הפרטי ושם המשפחה ששלח הלקוח בשיטה post נרשום:

```
string  userFirstName = Request.Form["userFirstName"];
string  userLastName = Request.Form["userLastName"];
```

userFirstName ו- userLastName הם השמות של תיבות הטקסט שבהן הוקלדו השם הפרטי ושם המשפחה של המשתמש. הערכים שהקליד המשתמש בתיבות אלה יושמו למשתנים מטיפוס מחרוזת userFirstName ו- userLastName, בהתאמה.

כעת נוכל להשתמש במשתנים אלו כדי לשלוח ללקוח את הניקוד ואת השם של הלקוח, נשתמש בפעולה Response.Write שבה השתמשנו קודם לכן:

```
Response.Write(userFirstName+" זכית ב "+points+" נקודות בהגרלה האחרונה");
```

כדי לבדוק שהלקוח אכן שלח שם המכיל תווים כלשהם, נשתמש בפונקציה IsValidName. הפונקציה מקבלת כפרמטר מחרוזת ובודקת אם המחרוזת קיימת (איננה null) ואינה ריקה (מכילה לפחות תו אחד). במקרה כזה היא מחזירה את הערך הבוליאני 'אמת'. אחרת – הפונקציה מחזירה את הערך 'שקר':

```
bool  IsValidName (string value) {
      return (value != null   && value!="" );                    // אם מחרוזת קיימת ואינה ריקה
```

```
}
```

שימו לב, אנו מוודאים שהמשתמש אכן הקליד תו אחד לפחות בתיבת הטקסט אך אין אנו בודקים אם השם תקני, כלומר אם הוא מכיל תווים שהם אותיות ואינו מכיל רק רווחים. אנו נרחיב בנושא זה בפרק 5, שבו נתאר בין השאר כיצד בודקים את התקינות של הנתונים. את הפונקציה הזאת נזמן מתוך הפונקציה CalculatePoints שהצגנו בדף Game1a.aspx.

להלן הקוד של דף המשחק המיישם את השינוי בכללי הניקוד והמאחד את דף ה-HTML ודף ה-ASP לדף יחיד שרץ בשרת, כלומר לדף ASP. הדף יציג למשתמש את כללי המשחק, ולחיצה על הכפתור 'נסו מזלכם' תחזיר למשתמש את אותו הדף אשר יציג את כללי המשחק, את המספרים שהוגרלו ואת הניקוד. את השינויים בקוד שעשינו ביחס לדף Game1a.aspx הדגשנו ברקע אפור (הוספת הגדרת פונקציה IsValidName, שינוי הפונקציה CalculatePoints ושינוי התסריט המכין תגובה ללקוח).

עיינו בקובץ Game2.aspx שלהלן:

```
<!—Game2.aspx →
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
  int n1, n2, n3 = 0;                          // הגדרת משתנים לאחסון המספרים שהוגרלו
  public void Page_Load()
  {
        Random rnd=new Random();               // הגדרת עצם ליצירת מספר אקראי
        n1 = rnd.Next(1,10);                   // הגרלת מספרים אקראיים
        n2 = rnd.Next(1,10);
        n3 = rnd.Next(1,10);
  }

  bool IsValidName(string name)
  {
          return (name != null && name != "");
  }

int CalculatePoints(string name)    {
```

```
    // חישוב הניקוד
     int  tempPoints;
    if ((n1 == n2) && (n2 == n3))                  // בדיקה אם שלושת המספרים זהים
       tempPoints = 10;
    else
       tempPoints = 2;                             // ניקוד בסיסי הוא 2 נקודות
                                                   // תוספת ניקוד למי שהכניס את שמו
    if (IsValidName(name))
    {
       tempPoints += 2;
    }
    return tempPoints;
  }
</script>

<%--
החלק שלפניכם הוא חלק העיצוב של הדף, הנלקח כאמור משני קבצים. קוד ה–HTML נלקח מהקובץ
game1.htm וקוד ה–ASP נלקח מקובץ game1a.aspx. השינוי שהוכנס בקוד ה–ASP מודגש באפור.
--%>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
  <title>Game Activity 2</title>
  <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
  <form id="form1" name="form1" action="Game2.aspx" method="get" runat="server">
  <div>
    <h1>ברוכים הבאים לאתר מכונת המזל </h1>
    <p>
     באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות <br />
     בכל לחיצה על הכפתור 'נסו מזלכם' יוגרלו 3 מספרים בין 1 ל–9 <br />
     ניקודכם יתעדכן על–פי הערכים שהוגרלו <br />
     במידה שיוגרלו 3 מספרים זהים, תזכו ב–10 נקודות <br />
     אם תקלידו את שמכם, תזכו ב–2 נקודות נוספות <br />
    </p>
```

```
<input type="text" name="userName" id="userName"  size="20" /> : הקלידו את שמכם
 <br /><br />
<input type="submit" value="נסו מזלכם" name="send"/>
 כפתור שליחת נתונים   <br /><br />
```

```
 <%
      int points;
     string name;
     name = Request.QueryString["userName"];
     points = CalculatePoints(name);
     Response.Write(name + " הניקוד שקיבלת  " + points  + "<br />");
     Response.Write("המספרים שהגרלת הם " + n1 + " " + n2 + " " + n3);
  %>
```

```
   </div>
   </form>
</body>
</html>
```

לסיכום:

- גם אם לא נרשום את המאפיין method בתג form, יישלחו הנתונים בשיטה get משום שזו ברירת המחדל.
- בדף לעיל, המשתנה points והמשתנה name מוגדרים בתסריט כמשתנים מקומיים. נעדיף להגדיר משתנים שדרושים רק במקום אחד כמשתנים מקומיים.
- ביישומי Web המתוכננים כהלכה, נהוג להפריד בין חלק התצוגה לחלק הבקרה שמכיל את הלוגיקה של התכנית. הפרדה לשני מודולים מאפשרת לטפל בכל מודול בנפרד. הקשרים בין המודולים צריכים להיות מוגדרים היטב וברורים. לפיכך, חלק ה-HTML של הדף כולל זימון של הפעולות הדרושות להכנת תגובה למשתמש בתוך התגים %> ו-<% ואת משפטי הפלט (ששייכים לתצוגה). כל ההגדרות של הלוגיקה של המשחק רשומות בחלק העליון של הדף בין התגים <script> ו-</script>.

- הפונקציה CalculatePoints מקבלת כפרמטר מחרוזת שאוחזרה מתיבת הטקסט שמלא המשתמש בטופס וקוראת לפונקציה IsValidName כדי לבדוק אם המשתמש שלח את שמו ומחשבת ומחזירה את הניקוד הסופי.
- הפונקציה IsValidName מקבלת פרמטר מטיפוס מחרוזת ומחזירה את הערך אמת אם המחרוזת קיימת ואיננה ריקה.
- המשפטים להגרלת המספרים האקראיים מתבצעים אוטומטית עם טעינת הדף משום שמשפטים אלו נמצאים בתוך האירוע Page_Load.

שאלה למחשבה

הריצו את היישום Game2.aspx. האם הוא פועל כהלכה? האם אתם יכולים להבחין בדבר-מה מוזר שמתרחש בעת ההפעלה הראשונה של המשחק? אם כן, מהי הסיבה לכך?

אכן מתרחש דבר-מה מוזר: בפעם הראשונה שמפעילים את המשחק, בעת שהשרת מציג את הוראות המשחק, הוא כבר מגריל שלושה מספרים ומציג את הניקוד שקיבל המשתמש. ההתנהגות הזאת אינה תקינה משום שהמשתמש עדיין לא ביקש להגריל מספרים וכבר קיבל ניקוד.
מהי הסיבה לכך?

בגרסה 2 של משחק 'מכונת המזל' אנחנו משתמשים בדף ASP אחד, ללא קובץ HTML שמציג דף פתיחה. בפעם הראשונה שהמשתמש מבקש את הקובץ, השרת צריך לשלוח כתגובה דף פתיחה (קובץ HTML) שמציג את כללי המשחק. אך בשלב זה אין לערוך הגרלה. רק לאחר שהמשתמש מבקש לנסות את מזלו במשחק, יש צורך לערוך הגרלה ואז להציג את הניקוד שלו. אבל כאן מתעוררת בעיה: כיצד תדע תוכנת השרת שזו הפעם הראשונה שהמשתמש ניגש לדף ה-ASP?

כיצד יודעים שדף נטען בפעם הראשונה?

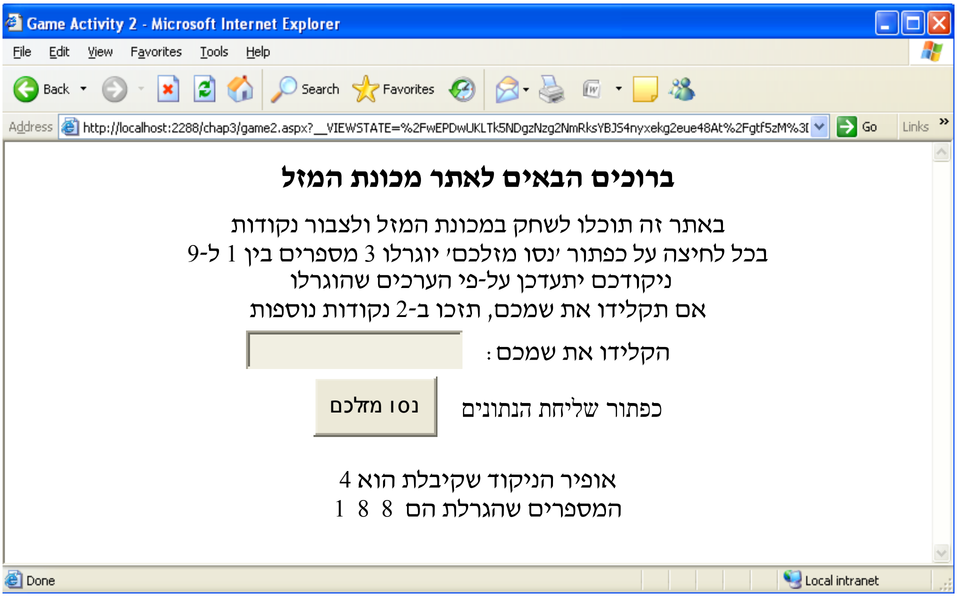
המאפיין IsPostBack

בבנייה של אתרים פעולה הבודקת אם הדף נטען בפעם הראשונה או לא היא פעולה שימושית מאוד. בדפים רבים נרצה להראות לגולש, בעת טעינה ראשונה של הדף, פרטים שונים שלא נרצה להראותם בטעינה חוזרת. לשם כך הוגדר מאפיין לעצם המייצג דף

העצם Page) הנקרא IsPostBack. אם הדף נטען בתגובה לבקשה חוזרת, מאפיין זה מחזיר את הערך הבוליאני 'אמת'. אם הדף נטען בפעם הראשונה – מוחזר הערך הבוליאני 'שקר'. העצם מהטיפוס Page הוא עצם נוסף המוגדר מראש בטכנולוגיית ASP.NET. נשתמש במאפיין זה ונשנה את התסריט שכתבנו בקובץ Game2.aspx כך שהפעולות לחישוב הניקוד והצגת הנתונים יזומנו רק אם המשתמש לחץ על הכפתור send :

```
<%
        int points;
        string name;
        if (Page.IsPostBack)
        {
            name = Request.QueryString["userName"];
            points = CalculatePoints(name);
            Response.Write(name + " שקיבלת הניקוד " + points + "<br />");
            Response.Write("המספרים שהגרלת הם " + n1 + " " + n2 + " " + n3);
        }
    %>
```

איור 3-4 מציג את הדף המוצג למשתמש אשר הקליד את השם 'אופיר' ; לכתובת ה-URL משורשרים הנתונים שנשלחו אל השרת.



איור 3-4

מסך המציג את הדף המתקבל לאחר שליחת השם לשרת

שאלה 3.4

א. הכניסו את השינויים הדרושים בדף Game2.aspx, וצרו את הדף Game2a.aspx הבודק שהמשתמש לחץ על הכפתור send והריצו אותו. השתמשו במאפיין IsPostBack – הריצו ובדקו את הדף.

ב. שנו את דף Game2a.apsx כך שהנתונים יישלחו בשיטה post וייקראו באמצעות Request.Form.

ג. צרו דף חדש שיאפשר למשתמש לשלוח לשרת את שמו הפרטי, את שם משפחתו ואת גילו. השרת יקרא את הנתונים ויכין הודעה בנוסח הזה:

Hello Ben Levy. You are 30 years old!

כאשר Ben, Levy, 30 הם הנתונים ששלח הלקוח.

קביעה של שיטת הניקוד במשחק

כעת, נוסיף לשחקן במשחק 'מכונת המזל' את האפשרות לשנות את שיטת החישוב של הניקוד. המשתמש יבחר באחת מבין שלוש שיטות הניקוד האלה:

- שיטת ניקוד ראשונה: אם הערך של כל המספרים שיתקבלו יהיה 7, יינתנו 50 נקודות.
- שיטת ניקוד שנייה: אם יוגרלו שלושה מספרים זהים שאינם 7 אך הם גדולים מ-5, יינתנו 20 נקודות.
- שיטת ניקוד שלישית: אם יוגרלו שלושה מספרים זהים שאינם גדולים מ- 5, יינתנו 10 נקודות (כמו קודם).

כדי לממש דרישה זו, נוסיף לטופס כפתורי רדיו (שאותם הצגנו בפרק 2). כפתורי הרדיו מאפשרים לבחור בערך אחד מבין כמה ערכים; במקרה שלנו, כל ערך מייצג שיטת ניקוד מסוימת. נשתמש בכפתורי הרדיו כדי לאפשר למשתמש לבחור את שיטת הניקוד.

```
ראשונה <input type="radio" id="rb1" name="group1" value="1" checked="checked" / >   <br />
שנייה <input type="radio" id="rb2" name="group1" value="2" /> <br />
שלישית <input type="radio" id="rb3" name="group1" value="3" /> <br />
```

נסמן את האפשרות הראשונה כברירת המחדל.

שאלה 3.5

האם אפשר להחליף את כפתורי הרדיו בתיבות סימון (checkbox) כדי לבחור את שיטת הניקוד של המשחק? נמקו את תשובתכם.

שימו לב, המחרוזות "1", "2" או "3" מייצגות את שיטת הניקוד של המשחק. שימוש במחרוזות הוא הדרך הנוחה ביותר לעיבוד המידע הזה משום שהנתונים שנשלחים מלקוח לשרת הם מהטיפוס מחרוזת. בדוגמה שלנו, הערכים "1", "2" ו-"3" מייצגים את שיטות הניקוד הראשונה, השנייה והשלישית, בהתאמה.

כעת, נשנה את התסריט שבו מוגרלים שלושת המספרים ומחושב הניקוד המתאים. בתחילה נאחזר את שיטת הניקוד שהמשתמש בחר ונאחסן אותה במשתנה level. הפעם נניח כי הטופס נשלח בשיטה post. על כן, נעבוד עם האוסף Request.Form. למימוש פעולות אלה נוסיף את הפונקציה GetPollsMethod. נוסף על כך, נשנה את הפונקציה CalculatePoints שתגריל שלושה מספרים ותזמן את הפונקציה GetPollsMethod כדי לחשב את הניקוד בהתאם לשיטת הניקוד שיבחר המשתמש.

```
string level;
level = Request.Form["group1"];                //קריאה של שיטת הניקוד שבחר המשתמש

//לאחר מכן נגריל שלושה מספרים ונחשב את הניקוד על-פי בחירת המשתמש
int points =2;                                 //ניקוד בסיסי הוא 2 נקודות
if ((n1 == n2) && (n2 == n3))                  //בדיקה אם שלושת המספרים זהים
{
   if ((level == "1") && (n1 == 7))            //שיטת חישוב ראשונה כל המספרים הם 7
      points = 50;
   else if ((level == "2") && (n1 > 5))        //שיטת החישוב השנייה כל המספרים גדולים מ-5
         points = 20;
      else if (level == "3")                   //שיטת החישוב השלישית: כל המספרים שווים
          points = 10;                         //וקטנים מ-5
}
```

לסיום, נרשום את הדף שיורץ בשרת במלואו (Game2b.aspx):

```
<!-- Game2b.aspx    -->
<!—HTMLהנחיות המגדירות את  שפת התכנות והסטנדרטים של  -->
```

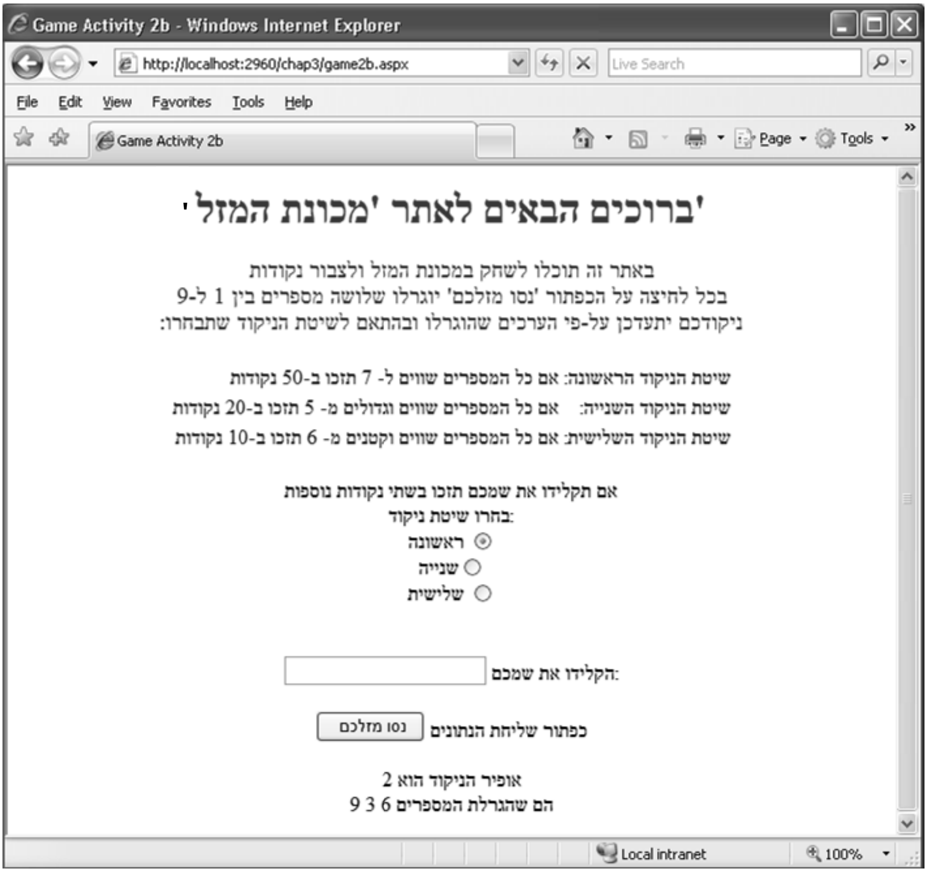
```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!--  תסריט לבחירת שלושה מספרים, שיטת הניקוד וחישוב הניקוד  -->
<script runat="server">
   int n1, n2, n3;
   string level;
   public void Page_Load()
   {
         Random rnd=new Random();            // הגדרת עצם ליצירת מספר אקראי
         n1 = rnd.Next(1,10);                // הגרלת מספרים אקראיים
         n2 = rnd.Next(1,10);
         n3 = rnd.Next(1,10);
  }
bool IsValidName  (string name)   {
      return (name != null && name != "");
   }

   string GetPollsMethod()
   {
      level = Request.Form["group1"];
      if (level != null)
         return level;
       else return "1";          //  הערך שיוחזר במידה והלקוח לא יבחר באף שיטת ניקוד
   }

   int CalculatePoints(string name)                    //  חישוב הניקוד
   {
       int tempPoints= 2;
        string level = GetPollsMethod();
      if (n1 == n2 && n2 == n3)
      {                          //  האם שלושת המספרים זהים
         if (n1 == 7 && level =="1")
         {              //  7 שיטת החישוב הראשונה – כל המספרים הם
            tempPoints = 50;
```

```
        }
        else if (n1 > 5 && level =="2")
        {               // 5-שיטת החישוב השנייה – המספרים גדולים מ
            tempPoints = 20;
        }
            else if (level =="3")
        {               // 5-שיטת החישוב השלישית – המספרים קטנים או שווים ל
      tempPoints = 10;
      }
     }
     if(IsValidName (name)){
        tempPoints +=2;
     }
   return tempPoints;
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
   <title>Game Activity 2b</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1" action="Game2b.aspx" method="post" runat="server">
   <div>
<h1>ברוכים הבאים לאתר 'מכונת המזל'</h1>
      <p>
       <br /> באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות
       <br /> בכל לחיצה על הכפתור 'נסו מזלכם' יוגרלו שלושה מספרים בין 1 ל-9
       <br /> ניקודכם יתעדכן על-פי הערכים שהוגרלו ובהתאם לשיטת הניקוד שתבחרו:
</p>
<table>
        <tr>
          <td> שיטת הניקוד הראשונה:</td>
          <td>אם כל המספרים שווים ל-7, תזכו ב-50 נקודות </td>
```

```
        </tr>
        <tr>
          <td>שיטת הניקוד השנייה:</td>
          <td>  אם כל המספרים שווים וגדולים מ-5, תזכו ב-20 נקודות </td>
         </tr>
         <tr>
          <td>שיטת הניקוד השלישית:</td>
         <td>אם כל המספרים שווים וקטנים מ-5, תזכו ב-10 נקודות</td>
       </tr>
       </table>
       <br />
    <br /> אם תקלידו את שמכם תזכו בשתי נקודות נוספות.
      <br /> :בחרו שיטת ניקוד
      ראשונה <input type="radio" id="rb1" name="group1" value="1"  checked="checked"  />
      <br />
      שנייה <input type="radio" id="rb2"  name="group1" value="2" /><br />
      שלישית <input type="radio" id="rb3" name="group1" value="3" /> <br />
      <br /></div>
      <div style="text-align:center"> <br  />
      הקלידו את שמכם
       <input type="text" name="userName" size="20" />  <br /><br />
       כפתור שליחת הנתונים
      <input type="submit" value=" נסו מזלכם" name="send"/>
       <br /><br />
<%
              // תסריט לקריאת שיטת החישוב ושם המשתמש וחישוב הניקוד
    int points;
    string name;
   // אם משתמש לחץ על כפתור השליחה
   if (Page.IsPostBack)
   {
            // זימון פונקציה לחישוב הניקוד
            name = Request.Form["userName"];
             points = CalculatePoints(name);
```

```
            Response.Write(name + " הניקוד הוא " + points + "<br />");
            Response.Write("המספרים שהגרלת הם " + n1 + " " + n2 + " " + n3 +"<br />");
  }
      %>
    </div>
  </form>
</body>
</html>
```

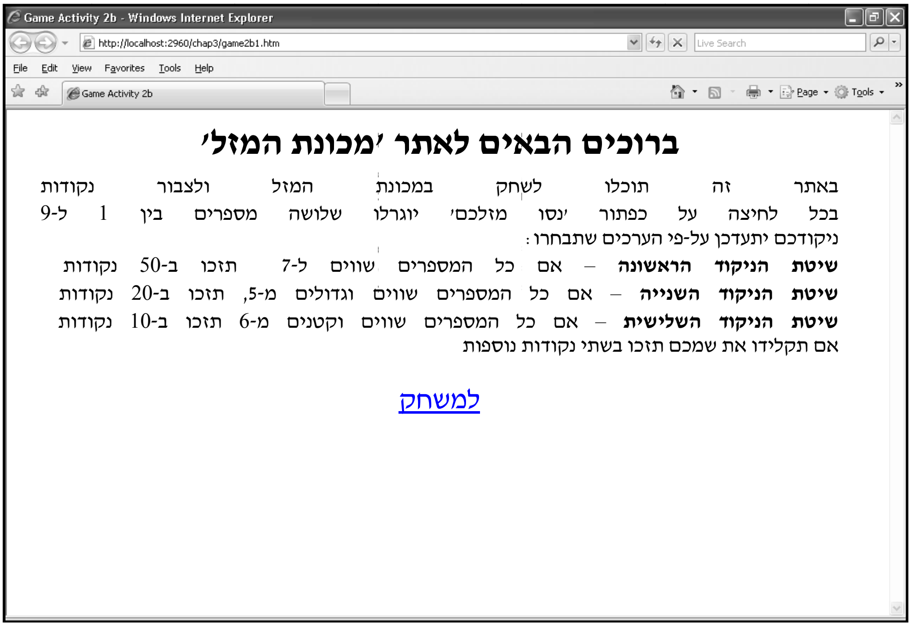
להלן המסך שיתקבל לאחר הבחירה בשיטת ניקוד ולחיצה על הכפתור 'נסו מזלכם'.



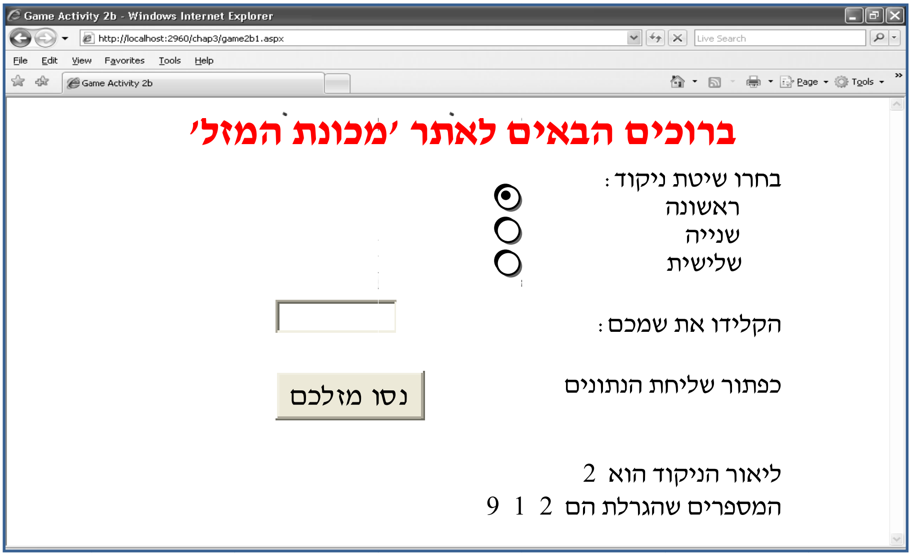
איור 3-5
דף משחק המכיל את הבחירה בשיטת הניקוד

שאלה 3.6

צרו דף HTML חדש אשר יציג את כללי המשחק ויכלול קישור לאתר המשחק, כמתואר באיור 3-6; צרו דף ASP אשר יכלול את טופס המשתמש, יחשב את הניקוד ויציג אותו, כמתואר באיור 3-7.



איור 3-6 דף HTML המציג את כללי המשחק וכולל קישור לאתר המשחק



איור 3-7 דף ASP לקליטה של נתוני הלקוח, שיטת הניקוד והצגתו

שאלה 3.7

שנו את משחק 'מכונת המזל' כדלקמן:

במסך הפתיחה של המשחק יוכל המשתמש לבחור אחת מבין שלוש רמות המשחק הבאות:

רמה 1 –

- o התחום שממנו יוגרלו שלושת המספרים הוא 1 – 3.
- o כאשר כל שלושת המספרים שונים זה מזה, הניקוד 1–.
- o כאשר שניים מהמספרים זהים, הניקוד הוא 3.
- o כאשר כל שלושת המספרים זהים, הניקוד הוא 10.

רמה 2 –

- o התחום שממנו יוגרלו שלושת המספרים הוא 1 – 6.
- o כאשר כל שלושת המספרים שונים זה מזה, הניקוד 2–.
- o כאשר שניים מהמספרים זהים, הניקוד הוא 5.
- o כאשר כל שלושת המספרים זהים, הניקוד הוא 20.

רמה 3 –

- o התחום שממנו יוגרלו שלושת המספרים הוא 1 – 9.
- o כאשר כל שלושת המספרים שונים זה מזה, הניקוד 3–.
- o כאשר שניים מהמספרים זהים, הניקוד הוא 8.
- o כאשר כל שלושת המספרים זהים, הניקוד הוא 50.

יש להציג לפני המשתמש את כללי הניקוד ברמות השונות, לקבל את בחירתו ואז להתחיל את המשחק. בדקו את שלוש הרמות: האם אתם מצליחים לזכות בניקוד חיובי? באיזו רמה אתם זוכים בניקוד גבוה יותר?

המרת נתונים

ראינו כי האוסף QueryString והאוסף Form מחזירים ערכים מהטיפוס מחרוזת. אם ברצוננו להשתמש בערכים המוחזרים כדי לערוך חישובים, עלינו להמיר בתחילה לנתונים מהטיפוס int או double, בהתאם לצורך.

נזכיר כי בשפת #C ניתן להגדיר משתנים מטיפוסי נתונים בסיסיים (primitives data type) שונים. טיפוס הנתונים מגדיר קבוצת ערכים ואת הפעולות שניתן לבצע עליהם. כאשר אנו מגדירים משתנה, עלינו להתאים לו טיפוס נתונים בהתאם לאופן השימוש בו. למשל ציונים של תלמידים הם בתחום בין – ל- 100 ולכן נגדיר משתנה שמיועד לאחסן ציון של תלמיד מהטיפוס int. לעומת זאת, משתנה שמאחסן מחיר של פריט, יהיה מספר לא שלם ולכן נגדיר אותו מהטיפוס double או float בהתאם לגודלו ומידת הדיוק הדרושה. הטבלה הבאה מסכמת את טיפוסי הנתונים בהם נשתמש בספר זה.

טבלה 3-1 טיפוסי נתונים בסיסיים

תיאור	הטיפוס	שימוש ל:
מספרים שלמים	short	ייצוג מספרים שלמים קטנים
	int	ייצוג מספרים שלמים
	long	ייצוג מספר שלמים גדולים
מספר ממשי	float	ייצוג מספרים ממשיים (לא שלמים) עם נקודה עשרונית קטנים
	double	ייצוג מספרים ממשיים (לא שלמים) גדולים ורמת דיוק גבוהה
בוליאני	boolean	ייצוג ערך בוליאני לשימוש במבני בקרה
תו	char	ייצוג של תווים /טקסט

נשנה את כללי המשחק כך שהמשתמש יוכל לקבוע (ולשלוח לשרת) את אחוז ההימור. בגרסה זו של המשחק, כללי המשחק יהיו כדלקמן:

1. השרת יגריל ניקוד בסיסי בין 1 ל-100, אשר יהיה מספר הנקודות הבסיסי שיקבל המשתמש.
2. השרת יגריל שלושה מספרים, n1, n2, n3 בתחום נתון.
3. השרת יקרא את אחוז ההימור שישלח המשתמש ויחשב את הניקוד בהתאם לכללים האלה:
   - אם כל המספרים n1, n2, n3 זהים, יתווסף לניקוד הבסיסי אחוז ההימור.
   - במקרה אחר (כלומר, לא כל המספרים n1, n2, n3 זהים), יופחת מהניקוד הבסיסי אחוז ההימור.

- **שימו לב**, בגרסה זו המשתמש אינו מתבקש לציין את שמו או לבחור ברמת קושי, אלא לקבוע את אחוז ההימור בלבד.

לדוגמה, נניח כי המשתמש ביקש שאחוז ההימור יהיה 10, והשרת הגריל ניקוד בסיסי של 50 נקודות ושלושה מספרים – 5, 6, 6. לפי החישוב הזה – 10% של 50 הם 5 נקודות ועל כן המשתמש ירוויח או יפסיד 5 נקודות. מאחר ששלושת המספרים אינם זהים, יפסיד המשתמש 5 נקודות מהניקוד הבסיסי . לפיכך, הניקוד יהיה:

$$50 - 5 = 45$$

אם הוגרלו שלושה מספרים זהים, לדוגמה: 6,6,6, המשתמש ירוויח 5 נקודות וניקודו יהיה:

$$50 + 5 = 55$$

כדי לממש את השינוי בכללי המשחק, על השרת לקרוא את המחרוזת שמכילה את אחוז ההימור ולהמירה למספר מטיפוס שלם או מטיפוס ממשי. כדי להמיר נתונים מטיפוס לטיפוס נשתמש בפעולה Parse של העצם double. לדוגמה, כדי להמיר מחרוזת "1234" למספר ממשי 12 נרשום:

```
string str = "1234";
double x= double.Parse(str);
```

באופן דומה ניתן להמיר מחרוזת לנתון מטיפוס שלם :

```
int f = int.Parse(str);
```

נכתוב פונקציה בשם GetBetPercentage שממירה את אחוז ההימור ששלח המשתמש כמחרוזת לערך מטיפוס שלם:

```
int GetBetPercentage()
  {
     int betPercentage = 0;                // הגדרת משתנה שבו יאוחסן אחוז ההימור
                                           // בדיקה לפני ההמרה: האם יש ערך להמרה?
     if (Request.Form["betPercentage"] != null) {
       // אם הוקלד אחוז ההימור, המר את המחרוזת למספר שלם
        betPercentage = int.Parse(Request.Form["betPercentage"]);
```

```
        }
        return betPercentage;
    }
```

כעת, נשנה את האירוע Page_Load כך שיכלול הגרלה של ערך הניקוד הבסיסי בין 1 ל-100, ונשמור אותו במשתנה baseNum :

```
public void Page_Load()
    {
        Random rnd=new Random();            // הגדרת עצם להגרלת מספרים אקראיים
        n1 = rnd.Next(1,10);                // הגרלת מספרים אקראיים
        n2 = rnd.Next(1,10);
        n3 = rnd.Next(1,10);
        baseNum =  rnd.Next(1,101);
    }
```

כמו-כן, נשנה את הפונקציה CalculatePoints שבה נזמן את הפונקציה GetBetPercentage כדי לאחזר את אחוז ההימור, ונחשב את הניקוד בהתאם לערך הניקוד הבסיסי שהוגרל ושאוחסן ב- baseNum (באירוע Page_Load) :

```
double CalculatePoints()
    {
        double points;
        int percentage = GetBetPercentage();        // אחזר את אחוז ההימור
                                // קבע את הניקוד לפי אחוז ההימור הבסיסי שהוגרל
        if ((n1 == n2) && (n2 == n3))       // האם המספרים זהים?
            points = baseNum * (1.0 + percentage / 100.0);
        else
            points = baseNum * (1.0 - percentage / 100.0);
        return points;
    }
```

בחלק שבו מעצבים את תגובת השרת, נוסיף שדה טקסט לקליטת אחוז ההימור ונכתוב את התסריט בצד השרת מחדש :

```
<%
    if (Page.IsPostBack)
      {
        Response.Write("המספרים שהוגרלו הם: " + n1 + " " + n2 + " " + n3 + "<br />"
```

```
+"אחוז ההימור הוא " + GetBetPercentage() + "<br />"
          + "הניקוד הבסיסי הוא " + baseNum + "<br />"
          + "בהגרלה האחרונה זכית " + CalculatePoints() + " נקודות" + "<br />");
} %>
```

שימו לב, זמנו את הפונקציות המחזירות את אחוז ההימור והניקוד כחלק מהכנה של תגובת HTTP.

בדיקת מצבי שגיאה בהמרה למספר

יש לשים לב שפעולה של המרת מחרוזת למספר עלולה להיות בעייתית. למשל, בדוגמה שלנו המשתמש נדרש להקליד מספר שהוא אחוז ההימור שלו. אם בטעות יקליד המשתמש ערך שאינו מספרי, אז הניסיון להמירו למספר ייכשל, והשרת ישלח ללקוח הודעת שגיאה במקום את הדף הרצוי.

כאשר לא ניתן להמיר מחרוזת למספר (למשל, כאשר היא מכילה אותיות או תווים לא מתאימים אחרים), ניתן להשתמש במנגנון לתפיסת חריגים[3], המאפשר למתכנת להגדיר מה צריך לבצע בעת שגיאה לוגית. המנגנון הזה מונע את קריסת התכנית. במנגנון הזה יש ל"עטוף" בתוך מבנה try – catch כל פעולה אשר עלולה לגרום לשגיאה בזמן הריצה. פעולות כגון int.Parse ו-char.Parse יזרקו חריגה במקרה שהפרמטר שנשלח אליהם אינו ניתן להמרה למספר (כאשר המחרוזת היא null או כאשר היא מכילה תווים שאינם יכולים להופיע כחלק מהמספר). ניסיון ההמרה יתבצע בתוך בלוק ה- try ואילו החריגה (שגיאת הריצה) תיתפס על-ידי בלוק ה-catch. במקרה של חריגה, כלומר הכנסת ערך שלא ניתן להמירו למספר, אחוז ההימור שיוחזר יהיה אפס, כך שהמשתמש לא יזכה כלל בנקודות (המילים try ו-catch הן מילים שמורות בשפת C#). נרשום אפוא את פעולת ההמרה במבנה הזה:

```
try {
        betPercentage = Integer.parseInt(paramString);
}
```

---

[3] חריג (exception) היא אירוע שנוצר כאשר יש שגיאה בביצוע הפעולות שרשומות בתוך הבלוק try. ברגע שנוצר אירוע כזה, הטיפול מועבר אל הפעולות שרשומות בתוך הבלוק catch.

```
catch (Exception e) {
        betPercentage =  0;
}
```

שאלה 3.8

שנו את הקובץ Game2b.aspx – הוסיפו לטופס שדה לשליחת אחוז ההימור; לאחר מכן בדקו אם חישוב הניקוד נעשה כהלכה. לשם כך, הדפיסו את הניקוד הבסיסי שיוגרל ואת הניקוד הסופי שיתקבל לאחר שימוש באחוז ההימור. את הקובץ הזה שמרו בשם game2c.aspx.

שאלה 3.9

תכננו דף שרת שמקבל מלקוח את פרטֵי ההזמנה של מנת פיצה. כאשר השרת מקבל את הבקשה, הוא מחשב מהו המחיר לתשלום ושולח ללקוח את המחיר ואת נתוני הפיצה שהוזמנה.

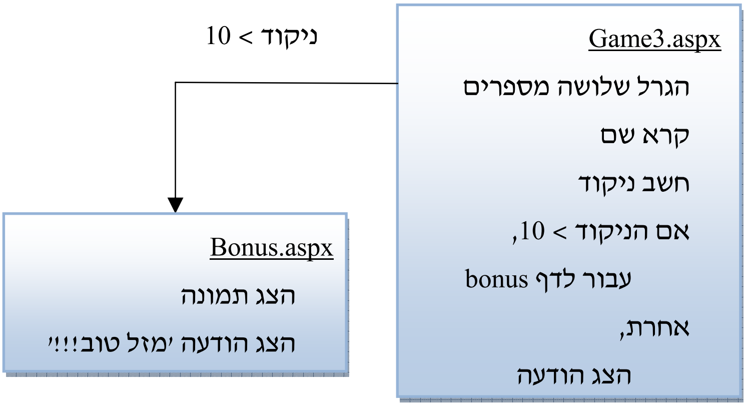
תכננו טופס שבאמצעותו יוכל הלקוח להזמין מנת פיצה. הטופס יכלול שדות לקליטת הנתונים האלה:

- שם המשתמש
- גודל מנת הפיצה: גדולה, בינונית או קטנה
- תוספת אחת או יותר מרשימת התוספות האפשריות: בצל, פטריות, ירקות, גבינה לבנה

בטופס שתבנו, השתמשו בכפתורי רדיו לבחירת גודל הפיצה ובתיבות סימון לבחירת התוספות. חישוב התשלום לפיצה ייעשה לפי גודל הפיצה ולפי מספר התוספות. מחירה של פיצה גדולה הוא 50 ₪, מחירה של פיצה בינונית הוא 40 ₪ ומחירה של פיצה קטנה הוא 30 ₪. הלקוח צריך לשלם 7 ₪ נוספים בעבור כל תוספת שהזמין. כמו-כן, יש להוסיף למחיר הפיצה מס ערך מוסף בשיעור של 15%.

3.3 הפניה לדף אחר

במקרים רבים, רוצים להפנות משתמשים לדף אחר. למשל, נוהגים להפנות משתמשים לדף אחר כאשר האתר מאובטח והמשתמשים צריכים תחילה להירשם אליו. משתמש אשר לא נרשם, יופנה לדף ההרשמה. בסעיף 3.5 נתאר כניסה מאובטחת לאתר המשחק, ונוסיף הפניה לדף רישום בעבור משתמש שאינו רשום לאתר. משתמש רשום יופנה לדף המשחק. בשלב הזה נחזור למשחק הבסיסי (קובץ Game2a.aspx), המכיל את האפשרות להוספת שם משתמש. נדגים כיצד משתמש שקיבל יותר מ-10 נקודות מופנה לדף אחר, שבו מוצגת ההודעה 'מזל טוב' ומופיע ציור של בלונים. האיור שלהלן מציג את השלד של הפתרון שמציג את הפעולות העיקריות שיש לממש.



איור 3-8
שלד הפתרון למשחק מכונת המזל ובו הפניה לדף אחר

כדי לעבור לדף אחר, נבצע את הפעולה Response.Redirect המקבלת שם של דף כפרמטר. זו פעולה של העצם Response. עם ביצוע הפעולה Redirect, השרת מחזיר תגובה המכילה כתובת URL חדשה שבה משתמש הדפדפן כדי לבקש בקשת HTTP חדשה.
למעשה, השינוי היחיד שיש לבצע בדף Game2a.aspx הוא לבדוק את הניקוד שקיבל המשתמש. אם ערכו גדול מ-10, יש להפנות את המשתמש לדף חדש בשם Bonus.aspx. את התנאי נוסיף בקוד התסריט הרשום בחלק התצוגה של הדף, מיד לאחר חישוב מספר הנקודות:

```
<%
```

```
        int points;
        string name;
        if (Page.IsPostBack)
        {
            name = Request.QueryString["userName"];
            points = CalculatePoints(name);
            if (points > 10)
                Response.Redirect("Bonus.aspx");
            Response.Write(name + " הניקוד שקיבלת " + points + "<br />");
            Response.Write("המספרים שהגרלת הם " + n1 + " " + n2 + " " + n3);
        }
%>
```

כתיבת התנאי בקוד הפעולה CalculatePoints המוגדרת בין תגי <script>…</script> הייתה נותנת תוצאה דומה במקרה של זכייה במספר נקודות גדול מ-10, אך הייתה שגויה מבחינת תכנון הקוד. הפעולה צריכה להחזיר ערך. אולם אם לפני ההחזרה הייתה מתבצעת פעולת שליחה לדף אחר, הפעולה לא הייתה מחזירה את הערך הדרוש. נוסף על כך, עקרונות התכנות המובנה מדגישים כי כל פעולה תבצע משימה אחת בלבד.

בהתאם נכתוב דף ה-Bonus.aspx המציג את ההודעה ואת התמונה:

```
<!--Bonus.aspx -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
   <title>Bonus</title>
    <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1"  runat="server">
   <div dir="rtl">
     מזל טוב <img alt="מזל טוב" src="Images/baloons.jpg"/>
   </div>
   </form>
</body>
</html>
```

שימו לב כי הקוד שלעיל מניח קיומה של תיקיית תמונות בשם Images המכילה קובץ בשם baloons.jpg.

שאלה 3.10

א. בצעו את השינויים הדרושים כדי ליצור הפניה לדף חדש :

1) שנו את הקובץ Game2a.aspx והוסיפו את התנאים להפניה לדף Bonus.aspx. שמרו את השינויים בקובץ בשם Game3.aspx.

2) צרו את הקובץ Bonus.aspx.

ב. האם ניתן להפנות את הלקוח לדף Bonus.htm, כלומר דף HTML? נמקו את תשובתכם.

שאלה 3.11

עצבו דף ברכה ליום הולדת. קלטו מהמשתמש את גילו ואת שמו של בעל יום ההולדת וכן את שם המברך, והפיקו כרטיס ברכה צבעוני המתאים לגילו של בעל יום ההולדת. השתמשו באובייקטים Response ו-Request ובאחת משיטות העברת הנתונים – post או get – כרצונכם. הכינו טופס לקבלת פרטים :

איור 3-9
דף המכיל טופס לקליטת נתוניו של הלקוח ושליחתם לשרת

עצבו את הברכה והוסיפו טקסט על-פי גילו של בעל יום ההולדת :

כמה גדלת! (לילדים עד גיל 6)

שיהיו לך חיים של צמיחה ושמחה! (לילדים עד גיל 12)

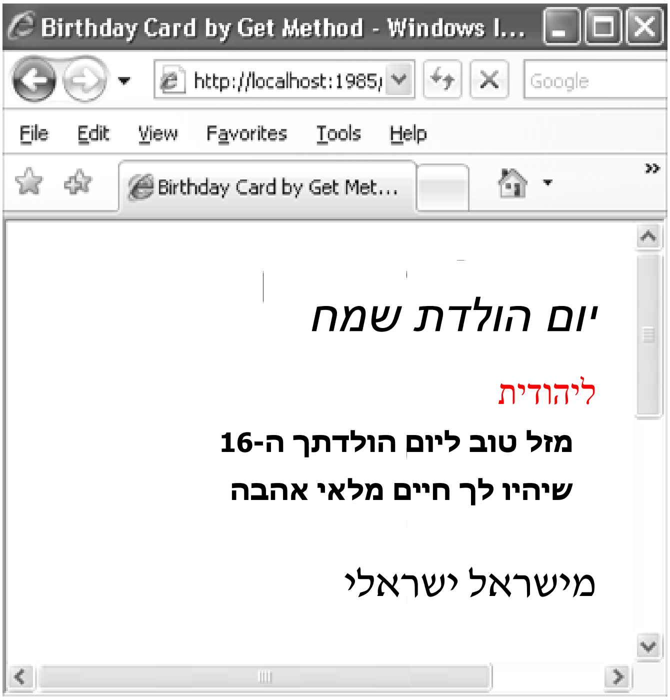
שיהיו לך חיים מלאי אהבה! (לילדים עד גיל 18)

שיהיו לך חיים מלאים שמחה ואהבה! (לאנשים עד גיל 70)

שיהיו לך חיים של אושר, בריאות, נחת ושמחה! (לאנשים מעל גיל 70)

הוסיפו לטופס אפשרות לבחירת צבעי הרקע, צבעי הטקסט, הוספת תמונות או הוספת אובייקטים אחרים, כרצונכם.

לפניכם ברכה לדוגמה שהוכנה לכבוד יום הולדתה ה-16 של יהודית.



איור 3-10
דף המציג ברכת יום הולדת

3.4 תכנות "מצב-חסר"

מהו תכנות "מצב-חסר" (Stateless Programming)?

בגרסת המשחק האחרונה שכתבנו (Game3.aspx), משתמש שזכה בניקוד גבוה הופנה לדף בונוס אשר הציג הודעה למשתמש. עם זאת, דף הבונוס לא הציג למשתמש לא את מספר הנקודות שבהן זכה ואף לא את שמו. לפיכך, נרצה להציג למשתמש המגיע לדף הבונוס את שמו ואת הניקוד שבו זכה.

נוסיף לדף Bonus.aspx את המשפט:

```
Response.Write(name + " " + points)
```

הניסיון לראות את הדף הזה בדפדפן יגרום שגיאה שכן המשתנים name ו-points הוגדרו בדף אחר, והדף Bonus.aspx אינו מכיר אותם.

יתרה מכך, גם הדף שבו הוגדרו הנתונים אינו זוכר את ערכם בזמן הטעינה מחדש. לדוגמה, אם נרצה לשנות את המשחק כך שהמשתמש יוכל לשחק כמה פעמים ולצבור את הניקוד שבו זכה, נזדקק למשתנה נוסף כדי לסכם את הזכיות. נכתוב אפוא דף בשם Game3a.aspx הדומה לדף Game3.aspx, ובו נגדיר משתנה שלם בשם sum וכן משפט לצבירת הניקוד ב-sum:

```
sum+=points;
```

בחלק התצוגה נציג את ערך המשתנה sum במקום את ערכו של points:

```
<%Response.Write("זכית בהגרלה האחרונה "+sum+" נקודות"+" "+name); %>
```

כתבו והריצו את הדף Game3a.aspx ולחצו על הכפתור 'נסו מזלכם' כמה פעמים; בכל פעם הקלידו שֵם בתיבת הטקסט כך שבכל לחיצה תזכו בשתי נקודות לכל הפחות.

שאלה למחשבה

האם הניקוד מחושב נכון? אם לא, מהי הסיבה לכך?

בעת הרצת הדף, המשתנה sum אינו צובר את הנקודות, אלא שומר אותן להגרלה הנוכחית בלבד, בדיוק כמו points. מדוע הדבר קורה?

כפי שהסברנו בפרק הראשון, יישומי Web מאופיינים ב'מצב-חסר'. כלומר, השרת אינו שומר נתונים על ההתקשרות עם הלקוח. במילים אחרות, לאחר שהשרת טיפל בבקשת HTTP, יצר את תגובת ה-HTTP ושלח אותה ללקוח, הוא אינו שומר מידע על ההתקשרות שהסתיימה. כל העצמים שהשרת הקצה במהלך ההתקשרות נהרסים; כל המידע שהמשתמש שלח לשרת וגם כל המידע שהשרת יצר ושלח ללקוח – נמחקים. לדוגמה, ייתכן שלקוח נכנס לדף מכונת המזל, שלח את שמו, קיבל את הניקוד המתאים וסיים את ההתקשרות עם השרת. כשאותו לקוח ייכנס שוב לאותו דף, השרת לא יזכור את שמו והמשתמש יצטרך לשלוח שוב את המידע הזה[4].

שאלה 3.12

תלמידים טענו כי כאשר מריצים את הקובץ Game3.aspx נשמר הטקסט המוצג בתחילת הדף (המתאר את כללי המשחק והמציג את השדות לקליטת נתוני המשתמש ושליחתם לשרת); בכל לחיצה על הכפתור 'נסו מזלכם', השרת מוסיף רק את שתי השורות המציגות את המספרים שנקלטו ואת הניקוד שחושב.

האם טענה זו נכונה? נמקו את תשובתכם.

במקרים רבים נרצה לשמור מידע על המשתמש. למשל, באתר משחק 'מכונת המזל' ברצוננו לשמור את הניקוד המצטבר ונתונים שונים, כגון שם משתמש וגילו; באתר קניות רוצים לשמור את עגלת הקניות של הלקוח כל עוד הוא גולש באתר ובוחר מוצרים; אתר בית-הספר צריך לשמור את מספר הזהות של התלמיד כדי לאפשר לו לראות את מערכת השעות, את השינויים החלים בה, את רשימת הציונים ועוד. במקרים כאלה נדרשת בשרת תוכנה שתנהל את המידע על פעולות המשתמש באתר, כלומר, תנהל את **מצב** (**state**) היישום.

קיימות כמה אפשרויות לשמירת מידע על המשתמש. כדי לבחור את האפשרות המתאימה, עלינו לדעת את כמות הנתונים שיש לשמור ואת טווח הזמן שבו יש לשמור נתונים אלה.

---

[4] כיום קיימים יישומי רשת ששומרים מידע על הלקוח בעזרת עוגיות (cookies), כך שנתוני הלקוח שיצא מהאתר וחזר אליו נשמרים. מידע זה נשמר בצד לקוח. שימוש במנגנון העוגיות חורג מתוכנו של ספר זה.

כאשר הנתונים שיש לשמור רבים ונדרש לשמור אותם לאורך זמן, נשתמש במסד נתונים המאפשר שמירה של כמות נתונים גדולה לאורך זמן. דוגמאות לנתונים כאלה הן סיסמאות הכניסה של המשתמשים, פרטי החשבונות של משתמשים ונתוני התלמידים בבית-הספר.

החלטה נוספת שיש להחליט בעת פיתוח דף שרת היא היכן יישמרו הנתונים: האם הנתונים יישמרו בשרת או במחשב הלקוח? היתרונות והחסרונות של כל שיטה מוצגים בטבלה שלפניכם.

טבלה 3-2 היתרונות והחסרונות בשמירת נתונים בצד הלקוח ובצד השרת

	שמירת הנתונים בצד הלקוח	שמירת הנתונים בצד השרת
היתרונות	• אין בזבוז של משאבי השרת (ניצול משאבי הלקוח)	• הנתונים מוגנים • הנתונים אמינים • לשרת יש שליטה מלאה על הנתונים • התעבורה ברשת פוחתת
החסרונות	• הנתונים לא מוגנים • הנתונים לא אמינים (לקוח יכול לשנות את הנתונים או לחסום אותם) • התעבורה ברשת גְדֵלה	• הטלת עומס על השרת

ניתן לראות כי היתרונות של שיטה אחת הם החסרונות של השיטה האחרת, ולהפך. בפרק זה נציג עצמים ושיטות לשמירת כמות קטנה של נתונים בשרת, ובפרק הבא נלמד כיצד לשמור נתונים במסד הנתונים בשרת.

ניהול מידע על משתמש יחיד (שימוש בעצם Session)

העצם **Session** (בעברית ישיבה, פגישה או מושב) הוא עצם המוגדר מראש ב-ASP שנועד לשמור מידע על משתמש (גולש) יחיד. לכל משתמש מוקצה עצם מהטיפוס Session, אשר יהיה זמין בעבורו כל עוד הוא פעיל באתר. בעצם Session ניתן לשמור מידע על המשתמש הדרוש להמשך הגלישה באתר.

העצם Session נוצר בעקבות בקשת HTTP, ונהרס כאשר המשתמש סוגר את הדפדפן, עובר לאתר אחר, או אינו פעיל באתר במשך פרק זמן מסוים (ברירת המחדל היא 20 דקות, אך אפשר להגדיר פרק זמן אחר).

להלן כמה מצבים שבהם נהוג להשתמש בעצם Session כדי לשמור מידע על משתמש יחיד:

- שמירת שם משתמש והניקוד המצטבר במשחק 'מכונת המזל'.
- שמירת שם משתמש בכניסה לאתר מאובטח לצורך זיהוי המשתמש בכל דפי האתר, ללא צורך בהקלדת שם וסיסמה בכל דף.
- שמירת פרטי עגלת הקניות של משתמש באתר קניות.

א. אחסון נתוני המשתמש בעצם מטיפוס Session

העצם Session מנהל מידע של לקוח מסוים שאליו ניתן לפנות באמצעות תסריטים כדי לאחסן ולאחזר את הנתונים של משתמש מסוים.

המידע מנוהל כאוסף של זוגות במבנה: <ערך-תכונה, שם-תכונה>, כאשר שם-תכונה הוא שם מזהה שמוצמד לערך מסוים. לדוגמה, המשתמש שגלש לאתר 'מכונת המזל' הקליד בטופס את השם 'Lior'. כדי לשמור נתון זה, נצמיד לו שם-תכונה username. בהמשך נוכל להשתמש בשם התכונה כדי לאחזר את השם של המשתמש. שם התכונה צריך להיות מחרוזת, והערך שלה נשמר בעצם מטיפוס Object[5] .

כדי לאחסן זוג כזה, נשתמש בפעולת ההשמה. למשל, כדי לאחסן שם של משתמש, למשל "Lior", תחת שם התכונה "userName", נרשום את המשפט הזה:

```
Session["userName"] =  "Lior";
```

ניתן גם להשים ערך משתנה מטיפוס מחרוזת, לדוגמה:

```
string name = "Lior"
Session["userName"] =  name;
```

אפשר גם לרשום

```
Session["sum"] = 0;
```

---

[5] או מחוון לעצם כלשהו. המחלקה Object היא מחלקת הבסיס בשפת C# והיא השורש בהיררכיה של כל שאר המחלקות (כולל מחלקות שמגדיר המשתמש).

נשתמש בעצם מהטיפוס Session כדי לשמור נתוני משתמש וכדי לכתוב מחדש את דף המשחק Game3.aspx (ראו סעיף 3.3), כך שאם הניקוד שקיבל המשתמש גדול מ- 9 יציג השרת בדף חדש – "דף הבונוס" את שם המשתמש, את הניקוד שבו זכה ותמונה של בלונים.

כדי שנוכל להשתמש בדף הבונוס בערכים שחושבו בדף המקורי, נשמור את הערכים האלה בעצם מהטיפוס Session. להלן קטע הקוד המתאים:

```
Session["name"] = name;             // Session-שמירת השם בעצם ה
 Session["points"] = points;        // Session-שמירת הניקוד בעצם ה
if (points > 10)
{
  Response.Redirect("bonus.aspx");
}
```

נוסף על כך, עלינו לשנות את הקובץ Bonus.aspx: נוסיף לחלק העיצובי תסריט קצר שבו נשתמש בעצם מהטיפוס Session כדי להכין את תגובת ה- HTTP:

```
<% Response.Write(Session["name"]+" זכית " + Session["points"] + " נקודות");%>
```

את גרסת המשחק החדשה נכתוב בדף Game4.aspx. לתסריט בחלק שבו אנו מכינים את תגובת השרת, נוסיף משפטי השמה לשמירת הניקוד ושם המשתמש בעצם מהטיפוס Session כדי שנוכל לזמן אותם מדף אחר. כמו כן, נוסיף משפט תנאי שבודק אם המשתמש לחץ על הכפתור send ואם הניקוד גדול מ-10 ובהתאם – מפנה לדף חדש השמור בקובץ bonus.aspx. להלן הקבצים המלאים.

```
  <!--  Game4.aspx -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- תסריט לבחירת שלושה מספרים וחישוב הניקוד -->
<script runat="server">
  int n1, n2, n3;                                  // הגדרת משתנה מטיפוס שלם לאחסון הניקוד
```

```
   string name;
   public void Page_Load()
    {
         Random rnd=new Random();          // הגדרת עצם ליצירת מספר אקראי
         n1 = rnd.Next(1,10);              // הגרלת מספרים אקראיים
         n2 = rnd.Next(1,10);
         n3 = rnd.Next(1,10);
    }
//פונקציה הבודקת אם השם אינו מחרוזת ריקה//
    public bool IsValidName(string name)
    {
        return (name != null && name != "");
    }

//פונקציה לחישוב הניקוד//
    public int CalculatePoints(string name)
    {
         int points;
        // חישוב הניקוד //
        if ((n1 == n2) && (n2 == n3))      // בדיקה אם שלושת המספרים זהים
           points= 10;
        else
           points= 2;                      //ניקוד בסיסי הוא 2 נקודות
          // תוספת ניקוד למי שהוסיף את שמו //
        if (IsValidName(name))      {
           points += 2;
        }
        return points;
    }
</script>

<!-- עיצוב הפלט להצגה ללקוח -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
```

```
    <title>Game Activity 4</title>
    <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
    <form id="form1" action="Game4.aspx" method="get" runat="server">
    <div>
        <h1>ברוכים הבאים לאתר מכונת המזל</h1>
        <p>
          באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות. <br />
          בכל לחיצה על הכפתור "נסו מזלכם", יוגרלו 3 מספרים בין 1 ל-9 <br />
          ניקודכם יתעדכן על-פי הערכים שהוגרלו <br />
          במידה שיוגרלו 3 מספרים זהים תזכו ב-10 נקודות <br />
          אם תקלידו את שמכם, תזכו ב-2 נקודות נוספות <br />
        </p>
        הקלידו את שמכם : <input type="text" name="userName" size="20" /> <br /><br />
        <input type="submit" value="נסו מזלכם" name="send"/>
        כפתור שליחת נתונים    <br /><br />
         <%
            int points;
           string name;
           if (Page.IsPostBack)
           {
               name = Request.QueryString["userName"];
               points = CalculatePoints(name);
               Session["userName"] = name;
               Session["points"] = points;
               if (points > 10) {
                   Response.Redirect("Bonus.aspx");
              }  else
               {
                   Response.Write(name + "הניקוד שקיבלת" + points + "<br />");
                   Response.Write("המספרים שהגרלת הם" + n1 + " " + n2 + " " + n3);
               }
           }
```

```
        %>
    </div>
    </form>
    </body>
</html>
```

בקובץ bonus.aspx נוסיף שורת פקודה לשרת – להדפיס את שם המשתמש ואת הניקוד שקיבל:

```
<!--  bonus.aspx -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 <script runat="server">
 </script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Bonus</title>
<link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
    <form id="form1"  runat="server">
    <div dir="rtl">
      מזל טוב <img alt="מזל טוב" src="Images/baloons.jpg"/>
      <h2>
        <% Response.Write(Session["userName"] + " זכית  ב– " + Session["points"] + "נקודות");%>
     </h2>
    </div>
    </form>
</body>
</html>
```

ב. שימוש בעצם מהטיפוס Session לצבירת הניקוד של משתמש

בסעיף הזה נשנה את הדף Game4.aspx ונשתמש בעצם מהטיפוס Session כדי לצבור את הניקוד של משתמש מסוים.

כפי שכבר ראינו, אי-אפשר להשתמש במשתנה רגיל כדי לשמור ערכים בעת המעבר בין דפים ואף בעת טעינה חוזרת של אותו הדף, משום שפרוטוקול HTTP הוא "מצב-חסר", ולכן ערך המשתנה sum לא נשמר מהגרלה להגרלה. בכל הגרלה הדף נטען מחדש ונוצרים משתנים חדשים.

לפיכך, כדי לבצע משימה זו נוסיף לאוסף של העצם Session את הזוג 'sum' כשם התכונה. בתכונה זו נאחסן את הניקוד המצטבר. בסיום, ערך התכונה של זוג זה יוצג לגולש. נתאר בסעיף זה את השינויים שנבצע בקובץ Game4.aspx.

כיצד יקבל Session["sum"] את ערכו? ראשית, יש להגדירו בפעם הראשונה שבה המשתמש גולש לאתר המשחק. לאחר מכן, כל עוד המשתמש אינו עוזב את האתר, הניקוד שהוא מקבל נצבר באמצעות הנתונים שנשמרו ב-Session הזה.

כדי להוסיף ל-Session את זוג הנתונים <ערך, שם-משתמש> בפעם הראשונה שהמשתמש גלש לאתר, נשנה את האירוע Page_Load ונוסיף הוראות שבודקות אם עדיין לא יצרנו זוג מתאים, כלומר נבדוק אם Session בשם sum נוצר או לא. אם לא, ניצור אותו ונאפס אותו.

```
if (Session["sum"]==null)
        Session["sum"] = 0;
```

כאמור, כאשר משתמש גולש לאתר, נוצר בעבורו עצם מהטיפוס Session. בונה האתר יכול להוסיף לאוסף הזה זוגות של שמות משתנים והפניות לעצמים כרצונו. עם זאת, עצם שלא הוגדר אינו קיים, ולכן ערכו null. כדי לבדוק אם עצם מטיפוס Session בשם sum עדיין לא נוצר, נבדוק אם ערכו הוא null. אם העצם לא נוצר, ניצור אותו ונציב בתוכו 0. יצירת העצם מתבצעת על-ידי מתן ערך התחלתי לעצם. המשפט Session["sum"]=0 יוצר את העצם ומאתחל אותו ל-0. כאשר לקוח ישלח בקשת HTTP נוספת, ערכו של עצם מהטיפוס Session בשם sum יישמר ולכן הוא יהיה שונה מ-null.

כדי לצבור את הניקוד, נוסיף את שורות הקוד הבאות לפעולת החישוב של הניקוד:

```
points = points + (int)Session["sum"];
Session["sum"]  = points;
```

הפעולה (int)Session["sum"] מאחזרת את ערך הניקוד (הקודם) שנשמר על ידי העצם Session וממירה אותו למספר מהטיפוס int. עלינו להמיר את הערך לטיפוס שלם משום שהערך הנשמר בעצם מהטיפוס Session הוא הפניה לעצם מטיפוס Object ש"עוטף" את המספר השלם. כעת, נוסיף לניקוד הקודם את ערכו של הניקוד שהתקבל בסיבוב הנוכחי של המשחק. לאחר מכן, את הערך המעודכן של סך כול הנקודות נשמור בעצם Session["sum"].

לסיום נציג את הניקוד המעודכן למשתמש. לשם כך נרשום בחלק בו אנו מעצבים את הפלט את המשפט הבא:

```
Response.Write( " בהגרלה האחרונה זכית ב" + Session["sum"] + "נקודות");
```

להלן התכנית המלאה השמורה בקובץ בשם Game4a.aspx. את הקטעים ששינינו צבענו באפור:

```
<! – Game4a.aspx -->
<%@ Page Language="C#" AutoEventWireup="true"   %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
  int n1, n2, n3 = 0;                          // הגדרת משתנים לאחסון המספרים שהוגרלו

  public void Page_Load()
  {
        if (Session["sum"]==null) {            // הגדרת Session לאחסון הניקוד
           Session["sum"] = 0;
           }
    Random rnd=new Random();                   // הגדרת עצם ליצירת מספרים אקראיים
        n1 = rnd.Next(1,10);                   // הגרלת מספרים אקראיים
        n2 = rnd.Next(1,10);
        n3 = rnd.Next(1,10);
```

```
 }

bool IsValidName(string name)
{

    return (name != null && name != "");
}

int CalculatePoints(string name)
{
   // חישוב הניקוד
   int tempPoints;
   tempPoints = (int)Session["sum"];            // אחזור הניקוד המצטבר עד כה
       if ((n1 == n2) && (n2 == n3))            // בדיקה אם שלושת המספרים זהים
           tempPoints += 10;
   else
             tempPoints += 2;
                                                // תוספת למי שהכניס את שמו
   if (IsValidName(name))
   {
      tempPoints += 2;
   }
   Session["sum"] = tempPoints;                 // עדכון הניקוד המצטבר ושמירתו
   return tempPoints;
}
</script>
<!--   החלק הבא הוא חל'ק העיצוב -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head >
   <title>Game Activity 4</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1" action="Game4a.aspx" method="get" runat="server">
   <div>
```

```
        <h1>ברוכים הבאים לאתר מכונת המזל</h1>
        <p>
          באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות. <br />
          בכל לחיצה על הכפתור "נסו מזלכם", יוגרלו 3 מספרים בין 1 ל-9 <br />
          ניקודכם יתעדכן על-פי הערכים שהוגרלו <br />
          במידה שיוגרלו 3 מספרים זהים תזכו ב-10 נקודות<br />
          אם תקלידו את שמכם, תזכו ב-2 נקודות נוספות <br />
         </p>
        <input type="text" name="userName" id="userName"  size="20" /> : הקלידו את שמכם <br
/><br />
        <input type="submit" name="send" value="נסו מזלכם" />
          כפתור שליחת הנתונים   <br /><br />
           <%
              int points;
             string name;
              if (Page.IsPostBack)
                  {
                 name = Request.QueryString["userName"];
                 points = CalculatePoints(name);
                 Session["userName"] = name;
                 Response.Write(name + " הניקוד שקיבלת" + points + "<br />");
                 Response.Write("המספרים שהוגרלת הם" + n1 + " " + n2 + " " + n3);
           }
           %>
    </div>
    </form>
</body>
</html>
```

שאלה למחשבה

עד כה השתמשנו בעצם Session לצורך הדפסות. מדוע לא השתמשנו בהמרה של ערכים למחרוזת?

בעת הדפסת ערכו של Session מודפסת למעשה המחרוזת המתארת את ערכו. אם ערכו היה מהטיפוס מחרוזת או מספר, כפי שראינו עד כה, אין צורך לציין במפורש את ההמרה כיוון שבשפת #C ישנה המרה אוטומטית של הערכים השמורים בעצם מטיפוס Session למחרוזת.

שאלה 3.13

שמרו את שם המשתמש ב-Session והציגו אותו בדף הבונוס (Bonus.aspx).
שימו לב, יש לבדוק אם המשתמש כבר מזוהה; אם לא, יש לבדוק אם הזדהה בסבב זה.

עד כה ראינו כיצד ניתן לשמור את שם המשתמש ואת הניקוד שלו. אולם כפי שניתן לראות הדף שמוצג למשתמש כולל תיבת טקסט בה ניתן להקליד את שם המשתמש גם לאחר שהמשתמש הקליד ושלח את שמו לשרת ושם זה נשמר בעצם מטיפוס Session. לפיכך, שיפור נוסף שנרצה לבצע הוא לשנות את הדף כך שבמידה ומשתמש הקליד את שמו, ושמו נשמר, בדף שהוא יקבל לא תוצג שוב תיבת טקסט לקליטת שם.

כדי לפתור בעיה זו, מספיק להתייחס ל- ["Session["userName במקום למשתנה name – הן כאשר נותנים שתי נקודות על ההזדהות והן כאשר מדפיסים את ההודעה ללקוח. עם זאת, הדף אשר יוצג למשתמש יציג גם את תיבת הטקסט לקליטת שמו, אף-על-פי ששמו כבר נשמר. נעדכן אפוא את הדף כך שמשתמש שהזדהה לא יתבקש עוד להזדהות, והמשתמש יזכה בנקודות על הזדהותו, בכל סבב של המשחק. לשם כך, נשמור את שם המשתמש ב-Session. בקוד הפעולה IsValidName נבדוק אם קיים ["Session["userName; אם הוא אינו קיים והמשתמש הקליד את שמו, נשמור את שם המשתמש ב-Session. משתמש מזוהה יקבל שתי נקודות נוספות לסבב.

```
if (Session["userName"] == null)                        // אם שם המשתמש לא נשמר עדיין
{
   name = Request.Form["userName"];   // נשמור את הערך מתיבת השדה
   if (name != "")                              // אם הוקלד שם
      Session["userName"] = name;           // נשמור את שם המשתמש בעבור הסבבים הבאים
}
if(Session["userName"] != null)                          // אם קיים ערך לשם המשתמש
```

```
points += 2;                          // נוסיף למשתמש שתי נקודות
```

כמו כן, נוסיף בחלק העיצובי של הדף תנאי הבודק אם קיים ה-Session כך שתיבת הטקסט להקלדת שם המשתמש תופיע רק אם המשתמש עדיין לא הזדהה:

```
<%if (Session["userName"] == null)
      Response.Write("<input type='text' name='userName' size='20' /> :הקלידו את שמכם
                     <br /><br />");
%>
```

לסיום, נציג לפני השתמש המזוהה את שמו בכל הודעה על הניקוד המצטבר:

```
<%Response.Write(" ניקודך המצטבר הוא "+Session["sum"]+" נקודות"+"  "
 +Session["userName"]); %>
```

נציג את הדף המתוקן במלואו (Game5.aspx) עם שינוי קל. בגרסה זו ובגרסאות הבאות של המשחק לא נַפנה את השחקן לדף הבונוס לאור צבירת ניקוד כלשהו.

```
 <!--  Game5.aspx -->
<%@ Page Language="C#" Debug="true" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   int points =0;
   int n1, n2, n3;
   string name;

   public void Page_Load()
   {
     if (Session["sum"] == null)
        Session["sum"] = 0;                  // איפוס סכום הזכייה
     points=(int)Session["sum"];             // סכום הנקודות עד כה

     Random rnd=new Random();                // הגדרת עצם ליצירת מספרים אקראיים
      n1 = rnd.Next(1,10);                   // הגרלת מספרים אקראיים
      n2 = rnd.Next(1,10);
```

```
        n3 = rnd.Next(1,10);

    }
    public bool IsValidName()
    {
        if (Session["userName"] == null)            // אם עדיין לא נשמר שם המשתמש
        {
       name = Request.QueryString["userName"];  // שמירת שם המשתמש שהתקבל בתיבת טקסט
        if (name != "")                              // אם הוקלד שם
            Session["userName"] = name;            // שמירת שם המשתמש לסבבים הבאים
        }
        return Session["userName"] != null;       // החזר ערך בוליאני 'אמת' אם קיים ערך לשם
    }
    public int CalculatePoints()
    {
        // חישוב הניקוד
       if ((n1 == n2) && (n2 == n3))                 // בדיקה אם שלושת המספרים זהים
           points += 10;
        else
           points += 2;

        // תוספת ניקוד למי שהקליד את שמו
        if (IsValidName())
        {
           points += 2;
        }
        return points;
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Game Activity 5</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
```

```
    <form id="form1" action="Game5.aspx" method="get" runat="server">
    <div style="text-align:center">
   <h1 style="color:Red"> `ברוכים הבאים לאתר 'מכונת המזל  </h1>
       <p style="color:Purple; font-size:larger" dir="rtl">
        <br /> באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות.
        <br /> בכל לחיצה של הכפתור 'נסו מזלכם', יוגרלו 3 מספרים בין 1 ל-9.
        <br /> ניקודכם יתעדכן על-פי הערכים שהוגרלו
        <br /> אם יוגרלו 3 מספרים זהים תזכו ב-10 נקודות
        <br  /> אם תקלידו את שמכם, תזכו ב-2 נקודות נוספות
       </p>
        <input type="submit" value=" נסו מזלכם" name="send"/>   כפתור לשליחת נתונים
        <br /><br />
<%  if (Request.QueryString["send"] != null)
       {
        points = CalculatePoints();                         // חישוב הניקוד החדש
         Session["sum"] = points;                           // שמירת הניקוד המעודכן
         Response.Write(Session["userName"] + " ניקודך המצטבר הוא " +
              Session["sum"] + " נקודות" +  "<br />");
          Response.Write("המספרים שהוגרלו: " + n1 + " " + n2 + " " + n3 + "<br />");
        }
         else {                                      // עדיין לא נשלח שם משתמש
          if (Session["userName"] == null)
             Response.Write("<input type='text' name='userName' size='20' /> : הקלידו את שמכם "
                                   +  "<br />");
        }
         %>
         <br /><br />
                  כפתור לשליחת נתונים
           <input type="submit" value=" נסו מזלכם" name="send"/>
      <br /><br />
</div>
    </form>
</body>
</html>
```

ג. ניהול עצמים מהטיפוס Session

שאלה למחשבה

למשך כמה זמן יישמרו ערכי "sum" ו-"userName" אשר שמרנו ב-Session?

ערכים אלה יישמרו כל עוד המשתמש לא עזב את האתר ובתנאי שפרק הזמן שבו המשתמש לא היה פעיל באתר לא חרג מן הדרוש. פרק זמן זה הוא תכונה של העצם Session, וברירת המחדל היא 20 דקות. בונה האתר יכול לשנות ערך זה על-ידי השמת ערך חדש (מספר שלם שמציין דקות) בתכונה Session.Timeout. אפשר לקצר או להאריך את פרק הזמן הזה. למשל, נקבע שפרק הזמן יהיה דקה אחת:

```
Session.Timeout = 1;
```

לאחר דקה אחת של חוסר פעילות באתר, גם אם מחשב המשתמש עדיין מחובר לאתר, ייהרס העצם Session וצבירת הנקודות תתחיל מחדש.

כדי להציג את פרק הזמן שנקבע לעצם מהטיפוס Session כברירת מחדל נכתוב את התסריט הזה:

```
<%  Response.Write("timeout " + Session.Timeout);%>
```

שאלה 3.14

שמרו את הקובץ Game5.aspx בשם Game5a.aspx. הוסיפו את המשפט Session.Timeout=1; בשורה הראשונה של האירוע Page_Load. הריצו את הדף, הזדהו ולחצו על הכפתור 'נסו מזלכם' כמה פעמים ברצף. עקבו אחר הניקוד המתעדכן. לאחר מכן, הרפו מן הכפתור 'נסו מזלכם' במשך דקה אחת, ואז לחצו עליו שוב. מהו מצב הניקוד כעת?

שאלה 3.15

שנו את דף המשחק כך ששחקן יוכל לצבור נקודות במשך שלושה סבבים, ואז יתאפס הצובר והמשחק יתחיל מחדש. לשם קבעו את פרק הזמן שנקבע לעצם מהטיפוס Session לזמן ארוך מאוד (למשל 30 דקות בהנחה ששלשה סבבים של משחק אורכים פחות מ- 30 דקות). שמרו את הדף הזה בשם Game5b.aspx.

אפשר להרוס עצם מהטיפוס Session באופן יזום על-ידי שימוש בפעולה Session.Abandon. פעולה זו הורסת את העצם מהטיפוס Session המתייחס לאותו משתמש. לדוגמה, נשנה את התכנית הקודמת כך שסכום הניקוד יצטבר בצובר, והמשתמש לא יוגבל לא בזמן המשחק ולא במספר הסבבים שהוא יכול לעשות. עם זאת, כאשר הניקוד של המשתמש יהיה גדול מ-50, יוצא השחקן מהמשחק.

נחזור לקובץ Game5.aspx: נמחק את המאפיין Timeout של אוסף ה-Session, ונוסיף בסיום האירוע Page_Load תנאי לבדיקה של מספר הנקודות ולהריסת העצם Session במקרה הצורך:

```
if ((int)Session["sum"]> 50)
        Session.Abandon();
```

שאלה 3.16

שנו את דף המשחק שכתבתם בשאלה 3.15 כך שלקוח שהניקוד שלו גדול מ-50 יוצא מהמשחק. שמרו דף זה בשם Game5c.aspx.

ניהול מצב היישום – ניהול מידע של משתמשים

נוסף על המידע הנשמר על כל משתמש בנפרד, נדרש לעתים לשמור מידע על היישום עצמו או על המשתמשים ביישום, שיהיה נגיש לכלל המשתמשים. לשם כך משתמשים בעצם מהטיפוס Application. לכל יישום Web המורכב ממספר דפי ASP יש עצם מטיפוס Application משותף.

בדומה לעצם מטיפוס Session, גם העצם Application (יישום) מנהל אוסף של זוגות במבנה: <ערך-תכונה, שם-תכונה> שבהם ניתן לאחסן הפניות לעצמים שיהיו נגישים לאותו יישום שרת לכלל הגולשים. שם התכונה צריך להיות מחרוזת והערך שלה יכול להיות עצם מטיפוס Object או עצם מטיפוס אחר. אופן העיבוד של מידע של יישום השמור בעצם מטיפוס Application דומה לאופן העיבוד של מידע השמור בעצם מהטיפוס Session.

העצם Application הוא אחד מהעצמים המובנים בסביבת ASP והוא נוצר כשמשתמש ראשון פונה ליישום מסוים. זמן החיים של Application הוא מרגע פתיחת היישום ועד סגירתו על-ידי אחרון המשתמשים. כמו כן, נפילת השרת שעליו רץ היישום, או אי-גלישה זמנית של משתמש כלשהו ביישום תסגור את היישום, והערכים שנשמרו ב-Application ייאבדו. בקשה חדשה ליישום תפתח את היישום מחדש וערכי ה-Application יאותחלו מחדש.

איזה מידע רצוי לשמור בעצם Application?

- מונה מבקרים: בכל כניסה של משתמש חדש נוסיף 1 למספר המבקרים באתר מראשית פעילותו.
- מונה גולשים בזמן אמת: בכל כניסה של משתמש חדש נוסיף 1 למספר הגולשים, ובכל סגירת Session נוריד 1 ממספר הגולשים.
- שמירת הודעות שיוצגו בכל דפי האתר.
- שמירת השיחות בחדרי שיחות סינכרוניים (צ'ט).

א. אחסון נתוני היישום (Application)

דוגמה – מונה מבקרים באתר

אתרים רבים מציגים בדף הבית את מספר המבקרים שביקרו באתר אי-פעם. לשם כך, משתמשים בעצם מטיפוס Application אשר מונה את מספר המבקרים ובשני דפים לפחות. נעשה זאת כעת. בשלב הראשון ניצור דף שרת שיאחסן בעצם מטיפוס Application ערך מספרי בשם siteCounter אשר ימנה את מספר המבקרים באתר. להלן משפט לאתחול ערך זה ל-0:

```
Applicatiom["siteCounter"] = 0;
```

נקרא לדף זה בשם SiteCounter.aspx והוא מציג הודעה אשר תציין אם איפוס המונה הצליח וכן קישור לדף נוסף (SiteCounterShow.aspx) שבו יוצג ערכו של המונה. להלן הקוד:

```
<!--SiteCounter.aspx-->
<%@ Page Language="C#" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
    public void Page_Load()
    {
        if (Application["siteCounter"] == null)
            Application["siteCounter"] = 0;
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Site Counter</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
    <form id="form1" runat="server">
    <div>
        <h3>המונה אותחל </h3>
        <a href="SiteCounterShow.aspx">עבור לאתר</a>
    </div>
    </form>
</body>
</html>
```

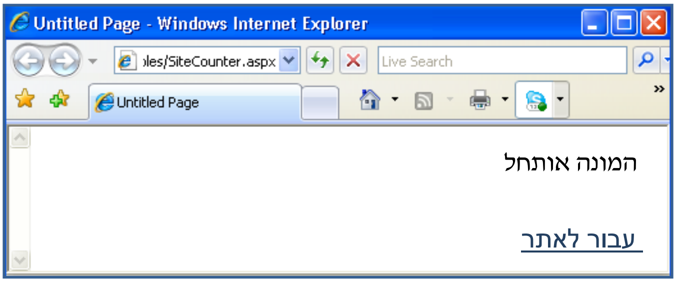
כעת ניצור קובץ בשם SiteCounterShow.aspx אשר יעדכן את מונה המבקרים (יוסיף לו 1) עם כניסתו של כל משתמש ויציג אותו.

שימו לב, נתון המאוחסן בעצם Application הוא עצם מהטיפוס Object ועל כן לא ניתן להוסיף לו 1. זכרו שיש לבצע המרה ל-int לפני עדכונו של האובייקט:

```
<!-- SiteCounterShow.aspx -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
    public void Page_Load()
```

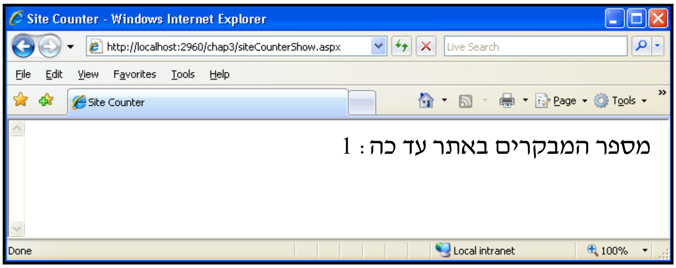
```
    {
        Application["siteCounter"] = (int)Application["siteCounter"] + 1;
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Site Counter</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
    <form id="form1" runat="server">
    <div>
    <%
        Response.Write(" : מספר המבקרים באתר עד כה" + Application["siteCounter"]);
    %>
    </div>
    </form>
</body>
</html>
```

כעת נריץ את הקובץ SiteCounter.aspx ונקבל את הדף הזה :



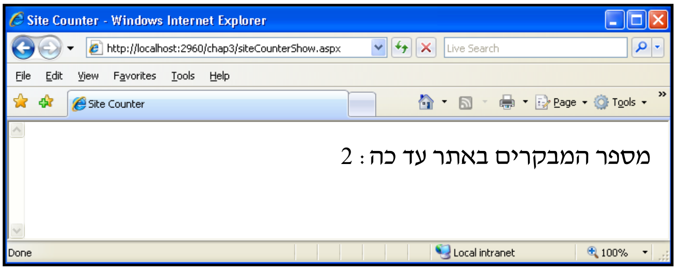
איור 3-11
דף המאתחל מונה מבקרים

כאשר ניכנס לאתר בפעם הראשונה נקבל את המסך הזה :



איור 3-12
דף המציג את מונה המבקרים

נטען שוב את הדף על-ידי רענונו (F5), ונקבל את המסך הזה :



איור 3-13
דף המציג מונה מבקרים מעודכן

כפי שניתן לראות בפתרון שכתבנו קיימת בעיה : מונה המבקרים התעדכן כשבצענו פעולת רענון. למעשה, מונה המבקרים שכתבנו אינו מונה את מספר המבקרים, אלא את מספר הבקשות שהדף מקבל. נניח כי משתמש יחיד נכנס לאתר וביצע רענון דף. הדפדפן שלח לשרת שתי בקשות HTTP : הבקשה הראשונה בעקבות הכניסה לאתר והבקשה השנייה בעקבות פעולת הרענון. לפיכך, מונה המבקרים מתעדכן ל-2 אף שלמעשה באתר נמצא רק מבקר אחד.

ניתן לשנות את המונה כך שימנה רק מבקרים חדשים באתר. כלומר, טעינה חוזרת של הדף לא תגדיל את המונה. כדי לזהות אם המשתמש הוא משתמש חדש או משתמש שכבר ביקר באתר וזו בקשה נוספת שלו, בדף SiteCounter.aspx נשתמש באוסף של עצם מהטיפוס Session כדי לשמור מידע על כל משתמש שגלש לאתר. נקבע את שם התכונה בשם 'user' (או כל שם אחר) והערך שנאחסן בה יהיה בוליאני מטיפוס בוליאני ('אמת' או 'שקר').

בכניסה הראשונה לאתר יעודכן ערכו של המשתנה Session["user"] ל'אמת'. ערכו של המשתנה מטיפוס Application יעודכן רק אם Session["user"]==false.

כדי למנוע ממשתמש לחזור לאחור (הכפתור Back) לדף SiteCounter.aspx ולהִיסָפֵר מחדש, האתחול Session["user"]=false יתבצע רק אם ה-Session לא מוגדר.

להלן השינוי הדרוש בקובץ SiteCounter.aspx :

```
if (Session["user"] == null)          // אם עדיין לא הוגדר משתמש
    Session["user"] = false;          // אתחל משתמש שקרי
```

ולהלן השינוי הדרוש בקובץ SiteCounterShow.aspx :

```
if(Session["user"]!= null && (bool)Session["user"]==false)
    {
        Application["siteCounter"] = (int)Application["siteCounter"] + 1;
        Session["user"] = true;
    }
```

עדכון המונה נעשה רק עבור משתמש חדש. משתמש חדש הוא זה שעבורו התכונה "user" לא הוגדרה או שהיא מוגדרת וערכה 'שקר'.

שאלה למחשבה

כיצד ייתכן שמשתמש ייכנס לאתר במצב שבו התכונה "user" אינה מוגדרת?

תשובה : בדפים שכתבנו הנחנו כי משתמש מפעיל תחילה דף SiteCounter שבו נוצרים עַצְמֵי Session ולאחר מכן הוא מופנה לדף SiteCounterShow המשתמש בעצמי Session שנוצרו בדף הקודם כדי להציג את מספר המבקרים. ואולם כאשר משתמש נכנס ישירות לדף SiteCounterShow.aspx ולא נכנס לאתר דרך SiteCounter.aspx, לא מוגדר בעבורו Session["user"]. כדי למנוע בעיה זו לא מספיק לבדוק אם (bool)Session["user"]==false, אלא ראשית יש לבדוק שהוא מוגדר, כלומר אם Session["user"]!=null.

לסיכום, נציג עתה את הדפים המעודכנים המונים את מספר המבקרים באתר :

```
<!-- SiteCounter.aspx -->

<%@ Page Language="C#" Debug="true"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   public void Page_Load()
   {
      if (Session["user"] == null)                    // אם עדיין לא הוגדר משתנה משתמש
         Session["user"] = false;                     // אתחל משתנה משתמש שקרי
      if (Application["siteCounter"] == null)         // אם עדיין לא הוגדר מונה מבקרים
         Application["siteCounter"] = 0;              // הגדרת מונה ואיפוסו
   }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Site Counter</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
   <form id="form1" runat="server">
   <div>
      <h3>אותחל המונה</h3>
      <a href="SiteCounterShow.aspx">לאתר</a>
   </div>
   </form>
</body>
</html>

<!--SiteCounterShow.aspx -->
<%@ Page Language="C#" Debug="true"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   public void Page_Load()
```

```
   {
       if (Session["user"] != null && (bool)Session["user"] == false)
       {
              Session["user"] = true;        //  הגדר משתנה משתמש
              Application["siteCounter"] = (int)Application["siteCounter"] + 1;
       }
   }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Site Counter Show</title>
  <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
   <form id="form1" runat="server">
   <div>
   <%
         Response.Write("מספר המבקרים באתר עד כה: " + Application["siteCounter"]);
   %>
   </div>
   </form>
</body>
</html>
```

ב. כניסה בו-זמנית של כמה משתמשים לאתר

עתה, משעדכנתם את מונה המבקרים, הריצו את הדף SiteCounter.aspx בשני חלונות דפדפן נפרדים על-ידי פתיחת חלון דפדפן נוסף והעתקת כתובת הדף. היכנסו לאתר באמצעות כל אחד מהדפדפנים. כיוון שפתחנו שני דפדפנים, היינו מצפים שבדפדפן הראשון מספר המבקרים שיוצג הוא 1 ובדף השני – 2, אך לא כך הדבר. מדוע?

נניח עתה כי ארבעה גולשים נכנסים בו-זמנית לאתר שיש בו מונה מבקרים. כל אחד מהגולשים יוסיף 1 למונה המבקרים, אך כיוון שארבעת הגולשים נכנסו לאתר בו-זמנית,

הם יתייחסו לאותו ערך של מונה המבקרים. לדוגמה, אם הערך של מונה המבקרים היה עד לכניסתם 100, אז בעבור כל אחד מהם השרת מחשב 100+1 ושומר את התוצאה במונה המבקרים. לאחר כניסתם יעודכן המונה לערך 101, אף-על -פי שמספר המבקרים הוא 104.

בעיה זו נקראת בעיית התנגשות, והיא מתרחשת כאשר מספר תהליכים פונים למשאב משותף. בדוגמה שלנו: המונה הוא המשאב המשותף. במקרה זה, כמה משתמשים עלולים לעדכן בו-זמנית את Application["siteCounter"]. כדי לפתור את בעיית ההתנגשות נצטרך לדאוג שכל משתמש יעדכן את ה-Application בזמן אחר. לשם כך, נשתמש בפעולה Application.Lock שמאפשרת רק למשתמש אחד לגשת לעצם Application ושחוסמת את הגישה של שאר המשתמשים לעצם הזה. למעשה, העצם נעול ואף משתמש אינו יכול לראות את ערכו ולשנותו. עתה אפשר לבצע עדכון ללא חשש מהתנגשות. בסיום העדכון, יש לשחרר את החסימה באמצעות הפעולה Application.UnLock, ובכך לאפשר למשתמש אחר לגשת לאותו עצם, לחסום אותו ולעדכנו.

הפעולות Application.UnLock ו-Application.Lock מאפשרות עדכון משאבים משותפים בצורה מסודרת וללא התנגשויות.

נשתמש בפעולות אלה ונעדכן את הקוד לעדכון מונה המבקרים של האתר כך שלא יתרחשו התנגשויות בגישה למשאב המשותף – Application["siteCounter"]:

```
public void Page_Load()
{
   if (Session["user"] != null && (bool)Session["user"] == false)
                                              // אם עדיין לא הוגדר משתנה משתמש
   {
      Session["user"] = true;                // הגדר משתנה משתמש בעל ערך אמת
      Application.Lock();                    // נעל עצם
                                              // הגדל ערך מונה המבקרים
       Application["siteCounter"] = (int)Application["siteCounter"] + 1;
      Application.UnLock();                  // שחרר את העצם הנעול
   }
}
</script>
```

שאלה 3.17

כתבו אתר למשחק המחשב 'נחש מה יצא לי'. דף הבית של האתר (היישום) יטיל ארבע קוביות משחק, כלומר, יגריל ארבעה מספרים בין 1 ל-6 וישמור אותם בעצמים מטיפוס Application. דף הבית יפנה את הגולש לדף המשחק.

בדף המשחק יתבקש המשתמש לנחש (בטופס) מהם הערכים של הטלות הקובייה (המשתמש יתבקש להכניס ארבעה ערכים בין 1 ל-6). לאחר שליחת הטופס יקבל המשתמש הודעה על מספר הניחושים הנכונים.

שאלה 3.18

הוסיפו לאתר המשחק 'נחש מה יצא לי' את שמו ואת התוצאה שהשיג משתמש אשר ניחש נכונה את כמות המספרים הגדולה ביותר.

שמרו את התוצאה הטובה ביותר ואת שמו של המשתמש שהשיג אותה במשתנים מהטיפוס Application. אם התקבלה תוצאה טובה יותר, שאלו לשמו של המשתמש ושמרו את שמו ואת התוצאה שהשיג.

שאלה 3.19

הוסיפו לאתר המשחק 'נחש מה יצא לי' מידע סטטיסטי, המציג את מספר הגולשים שניחשו נכונה מ-0 עד 4 ניחושים.

הוסיפו חמישה משתנים מהטיפוס Application, שכל אחד מהם מייצג ניחוש נכון – מ-0 ניחושים נכונים (כלומר, המשתמש לא ניחש אף מספר) עד 4 ניחושים נכונים. בכל פעם שגולש מנחש מספר, עדכנו את המשתנה המתאים. להלן דוגמה לפלט:

מספר הניחושים הנכונים	מספר הגולשים
0	2
1	3
2	1
3	4
4	2

שאלה 3.20

ניתן להשתמש בעצם Application לעריכת סקרים, בהם אנו שואלים את המשתמשים על נושא מסוים. המשתמשים יכולים לבחור אחת מכמה תשובות אפשריות שמציג הסקר. תוצאות הסקר יכולות להיות שמורות ב-Application , כך שלכל תשובה ייוחד Application משלה הצובר את מספר העונים על הסקר שבחרו בתשובה הזאת.

הוסיפו לאתר 'נחש מה יצא לי' סקר הבודק את שביעות הרצון של המשתמשים. הסקר יאפשר למשתמש לסמן את אחת מבין החלופות הבאות:

- התמכרתי
- מעביר את הזמן
- משעמם

לאחר שהמשתמש ענה על הסקר, יוצגו לפניו תוצאות הסקר עד כה.

3.5 כניסה מאובטחת ושימוש בסיסמה

דרכים להתמודד עם בעיות אבטחה באינטרנט

מאז שרשת אינטרנט נוצרה, בשנות ה-80 של המאה הקודמת, היא הפכה לאט לאט לרשת הגדולה בעולם, בה משתמשים כמעט בכל בית ועסק. בקליק קטן על העכבר ניתן להשיג מידע בכל נושא כמעט, לצאת למסע בעולם ובמקביל לבדוק מהי השעה בניו יורק, לדבר עם חברים, לשמוע מוזיקה, לראות סרטים ועוד. בכל יום יותר ויותר אנשים נעשים תלויים במידת מה במחשב האישי שלהם ומנהלים את סדר יומם דרכו אם באמצעות דואר אלקטרוני, יומן אלקטרוני, מסמכים הנשמרים על המחשב ועוד.

אולם יחד עם היתרונות הרבים עלינו לזכור כי קיימות סכנות רבות. החיבור לרשת האינטרנט, משמעו חיבור לכל מחשב אחר המחובר לרשת האינטרנט. התחברות לרשת האינטרנט עלולה לחשוף חומר אישי להתקפות זדוניות מכל מקום בעולם. ולכן, משתמשים המחברים את מחשביהם לרשת האינטרנט, חייבים להיות מודעים לסכנות, להשפעותיהן, וכיצד להתגונן מפניהם.

קיימים כיום מגוון של כלים שמסייעים לנו להתגונן מפני סכנות אלה. אחת הכלים הנפוצים הוא השימוש בחומת אש (Firewall), שהוא חלק ממארג שלם של אמצעים ושיטות לאבטחת האתרים. חומת אש של האינטרנט, היא תוכנה (ולעיתים ממומשת בחומרה) שמסייעת לסנן האקרים, וירוסים ותולעים שמנסים להגיע למחשב דרך האינטרנט. כאשר מותקנת חומת האש במחשב, היא עוקבת אחרי כל התקשורת שמקורה במחשב שלנו, והיא מונעת כניסת תעבורה לא רצויה למחשב. אם לא ביקשנו תעבורה נכנסת, חומת האש של החיבור לאינטרנט מנטרת את המִפתחים (ports) ומסייעת לחסום אותה לפני שהיא מגיעה למחשב. עבור שימושים מיוחדים כמו עבודה ברשת, אירוח משחקים מקוונים או אירוח שרת אינטרנט משלנו, ניתן לבחור את המִפתחים שאותן ברצוננו להשאיר פתוחות. זה מאפשר לאחרים להתחבר למחשב שלנו, אבל עלול גם לצמצם את רמת האבטחה.

אולם כמפתחים וכמשתמשים עלינו לבצע פעולות נוספות שיסייעו למנוע פרצות אבטחה. כדי לאבטח את האתר משתמשים בשיטות שונות לדוגמה:

א. שיטות לבקרת הגישה למידע

א.1. שימוש בסיסמאות היא השיטה המקובלת ביותר למניעת גישה למידע ממי שלא הוסמך לכך. לכל משתמש מורשה יש סיסמה שרק הוא אמור לדעת אותה. בדרך כלל כדי להגן על הסיסמה מפני אחרים, בזמן הקלדה הסיסמה מופיעים על הצג כוכביות או כל סימן אחר. הסיסמה שמשתמש בוחר צריכה להיות קשה לפענוח ובדרך כלל היא מורכבת מאוסף של אותיות וספרות. בסעיף זה נציג שימוש בסיסמה.

א.2 שיטה אחרת למניעת גישה למידע היא להשתמש בכרטיס מגנטי או אמצעי לזיהוי טביעת אצבעות של המשתמשים, זיהוי קול וכדומה. אמצעים אלו נמצאים בשימוש במקומות בהם המידע הוא רגיש וחשוב לשמור על סודיות.

ב. הגבלת זכויות השימוש

דרך נוספת למנוע מנזקים שיכולים להיגרם למידע של הארגון הוא להגדיר למשתמשים זכויות שונות של שימוש במידע, החל מהגדרה של משתמש שיכול לקרוא מידע מסוים, וכלה ברמה הגבוהה ביותר של משתמש שיכול גם לערוך, להוסיף ולמחוק תכנים. בצורה כזו הארגון יכול, מצד אחד, לתת למשתמשיו מידע הדרוש להם ומצד שני להגן על המידע

מפני שיבושים אפשריים. כיום קיימים כלים ממוחשבים רבים שמאפשרים בקלות יחסית להגביל את סוג ההרשאות למשתמש בודד או לקבוצה של משתמשים .

ג. הצפנת מידע

יישומי web שמבוססים על גישה של משתמשים מרשת האינטרנט חשופים לדליפת מידע. אחת הבעיות הדרכים המקובלות לשמור על סודיות מידע בעיקר על מידע רגיש (למשל פרטי כרטיס אשראי של לקוח שבצע עסקה בפורטל מסוג מחסר אלקטרוני) היא להשתמש בשיטות הצפנה. תוכנות המשתמשות בשיטות הצפנה מבצעות שתי פעולות:

1. להצפין מידע שנשלח באינטרנט כך שאחרים לא יוכלו לקרוא אותו
2. לפענח את המידע המוצפן.

ד. חתימה דיגיטאלית

משלוח מסכמים באינטרנט יצרה בעיה שנובעת מהקושי לזהות ולאמת את השולח. בעבר אחת השיטות להגברת האבטחה ולאמת שהמסמך הוא אותנטי הוא להשתמש בחתימה דיגיטאלית. כדי לזהות את הבעלים של מסמך שנשלח ברשת האינטרנט משתמשים בכלים ממוחשבים המטפלים בחתימה דיגיטאלית.

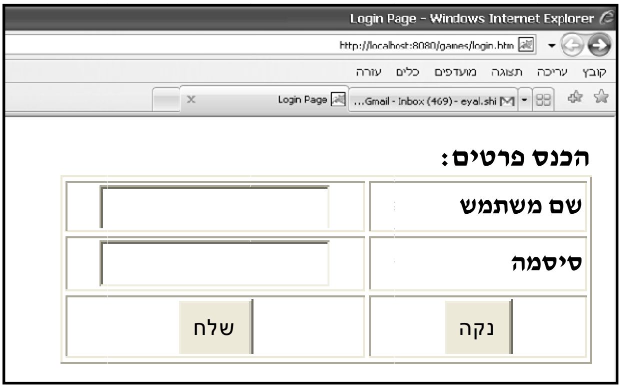
דף כניסה עם סיסמה

בסעיף זה נדגים כיצד לכתוב דפים שמאפשרים כניסה לאתר מוגבלת למשתמשים רשומים בלבד. משתמשים מתבקשים להכניס שם משתמש וסיסמה לפני שיורשו להיכנס לאתר. לדוגמה, נשנה את דף המשחק כך שבתחילה יתבקש המשתמש להקליד שם וסיסמה. אם המשתמש הקליד סיסמה תקינה, הוא יופנה לדף המשחק שבו יוצג שמו ויוגרלו מספרים.

האתר יורכב מהדפים הבאים:

א. Login.htm – קובץ המכיל טופס (המוצג באיור 3-14), שבו המשתמש נדרש לרשום שם וסיסמה בתיבת טקסט מסוג password. שימוש בערך הזה למאפיין type גורם לכך שהקלט שמוקלד לא יוצג בתיבה; במקום התווים שיוקלדו יוצגו כוכביות.

ב. User.aspx – קובץ המעבד את נתוני הטופס שנשלחו אליו מהקובץ Login.htm, ומציג ברכת שלום הכוללת את שם המשתמש שנקלט. השם והסיסמה יישלחו בשיטה post כדי לא לחשוף את הסיסמה.



איור 3-14
טופס לקליטת שם משתמש וסיסמה

לפניכם דף HTML המציג טופס לקליטת שם משתמש וסיסמה:

```
<!-- Login.htm-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Login Page</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
  <h1>הכנס פרטים :</h1>
  <form id="Login" action="User.aspx" method="post" >
    <table border="1" style="width:300px; text-align:center" dir="rtl">
     <tr>
       <td style="width: 200px">
         <input type="text" name="userName" id ="userName" size="20"  style="width: 200px"/>
       </td>
       <th style="width: 204px">
         <p style="text-align:right; width:150px; font-size:4mm; font-family:Arial">שם משתמש</p>
       </th>
     </tr>
```

```
    <tr>
      <td style="width: 200px">
        <input type="password" name="password" size="20" style="width: 200px"/>
      </td>
      <th style="width: 204px">
        <p style="text-align:right; width:150px; font-size:4mm; font-family: Arial">סיסמה</p>
      </th>
    </tr>
    <tr>
      <th style="width: 200px">
       <input type="submit" value="שלח" name="send"/>
      </th>
      <th style="width: 204px">
       <input type="reset" value="נקה" name="clear"/>
      </th>
    </tr>
  </table>
 </form>
</body>
</html>
```

הקובץ אשר יטפל בהתחברות הוא User.aspx. קובץ זה ישמור את שם המשתמש ב-Session ויציג ברכת שלום למשתמש. נדגיש כי את הסיסמה לא נשמור ב-Session ומובן שלא נציג אותה על המסך שכן הסיסמה נועדה אך ורק לצורך זיהוי המשתמש בעת ביצוע פעולת ההתחברות. בהמשך הגלישה אין לנו צורך בסיסמתו של המשתמש. בפרק הבא נלמד כיצד ניתן לשמור סיסמה במסד הנתונים, ולאחזר אותה כדי לבדוק שהמשתמש אכן הקליד סיסמה תקינה.

```
<! – User.aspx -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
```

```
    public void Page_Load()
    {
        string userName = Request.Form["userName"];
        Session["userName"] = userName;
    }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Home Page</title>
    <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body dir="rtl">
    <form id="form1" runat="server">
    <div>
        <%
            Response.Write("<h2>  שלום " + Session["userName"] + "</h2>");
        %>
    </div>
    </form>
</body>
</html>
```

בפתרון שהצענו טמונה בעיה. למשל , מה יקרה אם משתמש לא יקליד פרטים כלל וישלח טופס ריק? במקרה כזה המשתמש יקבל את הדף הזה:

איור 3-15
דף שמוצג כאשר משתמש אינו שולח שֵם משתמש

נצטרך להתמודד עם בעיה זו גם כאשר נרצה למנוע ממשתמש לא מורשה להיכנס לדפים מאובטחים – נעשה זאת באמצעות העצם Session.

כדי לפתור את הבעיה, נשנה את הקובץ User.aspx ונוסיף משפט תנאי שיוודא שרק גולש שהזדהה יוכל לראות את הדף הזה. ניסיון גלישה ללא מתן פרטי הזדהות יחזיר את הגולש לדף של טופס ההתחברות login.htm. נבדוק שהגולש אכן הזדהה, כלומר ששמו וסיסמתו אינם מחרוזת ריקה. עם זאת, נשמור ב-Session רק את שמו של הגולש.

בדף User.aspx נשנה את האירוע ()Page_Load :

```
public void Page_Load()
{
    Session["userName"] = Request.Form["userName"];
        string password = Request.Form["password"];

    if ((string)Session["userName"] == "" || password == "")
        Response.Redirect("login.htm");
}
```

הריצו את הקובץ login.htm ושלחו טופס ריק. האם עברתם לדף השמור בקובץ user.aspx? מובן שתשובתכם היא לאו. שנו עתה את כתובת ה-URL בשורת הכתובת שבדפדפן וכתבו User.aspx (במקום Login.htm). האם עברתם עתה לדף User.aspx? אכן כן. כיצד נמנע אפוא ממשתמש המכיר את שם הדף מלהיכנס לדף שאינו מורשה לראותו? בעת טעינת הדף (האירוע Page_Load) נבדוק שהמשתמש מזוהה, כלומר שהעצם Session בעבור שם המשתמש איננו null. לשם כך נוסיף לתנאי בדיקה נוספת. להלן האירוע Page_Load עם כל הבדיקות הדרושות:

```
public void Page_Load()
  {
    //אם שם המשתמש ריק, שלח את הגולש לדף הכניסה
    if ((Session["userName"] == null)   || ((string)Session["userName"] == "" || password == ""))
        Response.Redirect("Login.htm");
    Session["userName"] = Request.Form["userName"];            // אחזור שם משתמש
    string password = Request.Form["password"];                 // אחזור סיסמה
    if ((string)Session["userName"] == "" || password == "")    // אם שם משתמש ריק או
                                                                //סיסמה ריקה
```

```
        Response.Redirect("Login.htm");         // שלח גולש לדף כניסה
  }
```

עתה אנו יכולים לבנות אתר למשחק מכונת המזל המאפשר למשתמשים להיכנס לאתר רק עם שם משתמש וסיסמה. נשתמש בגרסת המשחק הכתובה בדף Game5.aspx.

שאלה 3.21

שנו את הדפים login.htm, Game5.aspx, כך שלאחר שהמערכת וידאה שהשם והסיסמה אינם ריקים, תתאפשר הפניה לדף המשחק Game5.apsx. לאחר הגרלת המספרים וחישוב הניקוד יוצגו למשתמש שמו והניקוד שבו זכה.

לסיכום, האתר שלנו מכיל כבר כמה דפים: דף המשחק Game5.aspx, דף ההתחברות Login.htm ודף המשתמשים המורשים user.aspx. נעדכן את דף המשחק כך שלא יבקש מהמשתמש להזדהות (שכן אפשר להגיע לדף רק לאחר הזדהות) אבל יבדוק אם המשתמש הזדהה כדי למנוע כניסה ישירה למשחק דרך רישום כתובת ה-URL של הדף. בכל מקרה, המשתמש יקבל שתי נקודות עם כל לחיצה על הכפתור 'נסו מזלכם'. כמו כן נוסיף לדף זה הפניה לדף הראשי, הוא Index.aspx, המוצג בהמשך. נשמור גרסה זו של המשחק בקובץ בשם Game.aspx. נוסיף לאתר דף בית אשר יתאר למשתמש את המשחק שבאתר ויזמינו להירשם לאתר. משתמש שהזדהה יוכל לגלוש מן הדף הזה לדף המשתמשים המורשים, ומשם יופנה ישירות לדף המשחק. בדף המשחק תתווסף האפשרות להתנתק מהמשחק. הניתוק יתבצע דרך הדף השמור בקובץ Logout.aspx אשר יהרוס את כל עַצְמֵי ה-Session של המשתמש ויפנה אותו מחדש לדף הבית. למעשה, הדפים השמורים בקבצים Logout.aspx ו-User.aspx הם דפי ניתוב בלבד ואינם מוצגים למשתמש.

להלן פירוט הקוד של דף הבית, דף המשתמש ודף ההתנתקות.

דף הבית (Index.aspx) יציג מידע שונה למשתמשים שונים. למשתמש שעדיין לא נרשם תוצג מטרת האתר והמערכת תבקש ממנו להזדהות על-ידי שם משתמש וסיסמה; למשתמש רשום יוצגו שתי אפשרויות: לגלוש לדף המשחק או להתנתק.

```
<!—Index.aspx →
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Home Page</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1" runat="server">
   <div style="text-align:center">
      <h1 style="color:Red">ברוכים הבאים לאתר מכונת המזל</h1>
      <%
         // משתמש שעדיין לא נרשם יופנה לדף הרשמה
         if (Session["userName"] == null)
         {
            Response.Write("<p style='color:Purple; font-size:larger' dir='rtl'>");
            Response.Write("באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות ");
            Response.Write(".<br /> לשם כך עליכם להירשם<br /></p>");
            Response.Write("<a href='Login.htm'>לטופס ההתחברות</a>");
         }
         // משתמש מחובר יוכל לגלוש למשחק או להתנתק
         else
         {
            Response.Write("שלום לך " + Session["userName"] + "<br />");
            Response.Write(" לפניך מספר אפשרויות:");
            Response.Write("<a href='Game.aspx'>לשחק במכונת המזל  </a><br />");
            Response.Write("<a href='Logout.aspx'>להתנתק</a><br />");
         }
```

```
        %>
      </div>
   </form>
</body>
</html>
```

להלן דף המשתמש לניתוב הגולש לדף המתאים (User.aspx) :

```
<! – User.aspx -->
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   public void Page_Load()
   {
      if (Session["userName"] == null)            // אם שם המשתמש עדיין לא נשמר
      {
                                                    // שמירת הערכים שהתקבלו מהטופס
         string name = Request.Form["userName"];
         string password = Request.Form["password"];
                                                    // אם לא הוקלדו נתונים
         if (name == null || name == "" || password == null || password == "")
            Response.Redirect("login.aspx");        // חזור לדף ההתחברות

         Session["userName"] = name;                // נשמור את שם המשתמש לסבבים הבאים
      }
      Response.Redirect("Game.aspx");               // עבור לדף המשחק
   }

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Home Page</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
```

```
</head>
<body dir="rtl">
   <form id="form1" runat="server">
   <div>

   </div>
   </form>
</body>
</html>
```

להלן דף ההתנתקות (Logout.aspx):

```
<!-- Logout.aspx -->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
   public void Page_Load()
   {
      Session.Abandon();
      Response.Redirect("Index.aspx");
   }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Logout Page</title>
   <link rel ="Stylesheet"  type="text/css" href ="Chap3.css" />
</head>
<body>
   <form id="form1" runat="server">
   </form>
</body>
</html>
```

3.6 פעילות מסכמת – בניית אתר למשחק מכונת המזל

נמשיך בבניית האתר לפרויקט המלווה את הספר הזה. כללי המשחק באתר הם הכללים המוגדרים בשאלה 3.3 ושמורים בקובץ Game1c.aspx.

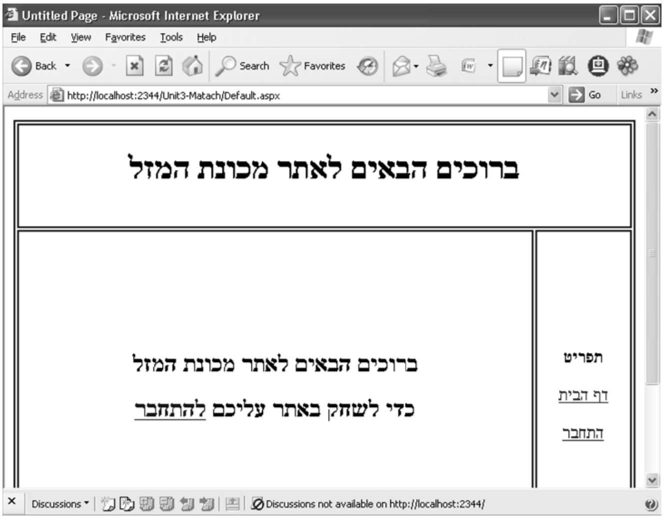
שלב א׳

בנו דף המבוסס על טבלה בעלת שתי שורות ושתי עמודות.

- בשורה הראשונה של הטבלה מזגו את שני התאים, וכתבו כותרת ברמה 1 ׳ברוכים הבאים לאתר מכונת המזל׳.
- בשורה השנייה של הטבלה ישמש תא אחד לתפריט שיטוט באתר, והתא האחר – לתוכן העמוד.
- האתר יכיל דף בית, דף משחק, דף התחברות, דף התנתקות ודף ניהול.

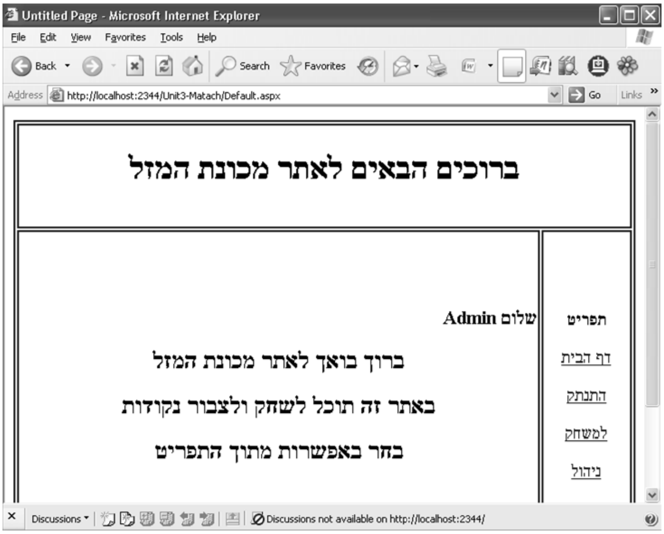
נגדיר את המשתמש ששמו admin ושסיסמתו 1234 כמנהל והוא לבדו יוכל לראות את דף הניהול. הגדרת Session["admin"] תיעשה בדף User.aspx לאחר שנבדק ערכם של שדות הטופס.

הכניסה לדף הבית תהיה חופשית ומשתמש שלא נרשם ייראה את הדף הזה:



איור 3-16
דף למשתמש באתר

מנהל האתר יראה את הדף הזה:



איור 3-17
דף מנהל אתר

משתמש רשום יראה דף הדומה לדף שיראה המנהל; בדף הזה לא יהיה קישור לדף הניהול ויופיע שמו של המשתמש.

להלן שלד של תכנית שלתוכו אתם יכולים להוסיף את ההוראות שלכם. תחילה, רשמו הערות בדף המתארות את השינויים בתוכן הטבלה שבניתם, ולאחר מכן הוסיפו הוראות אלה לדף. בדף שלהן מסומנות מספר שורות בצלליות. הוסיפו את ההערות והתכנים לשורות אלה כך שיהיה אפשר לזהות בקלות את התכנים שמציג כל הדף.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
   <title>Game Activity - Slot Machine</title>
   <link rel="Stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
 <form id="form1" action="index.aspx" method="post" runat="server">
 <div>
 <table>
   <tr>
```

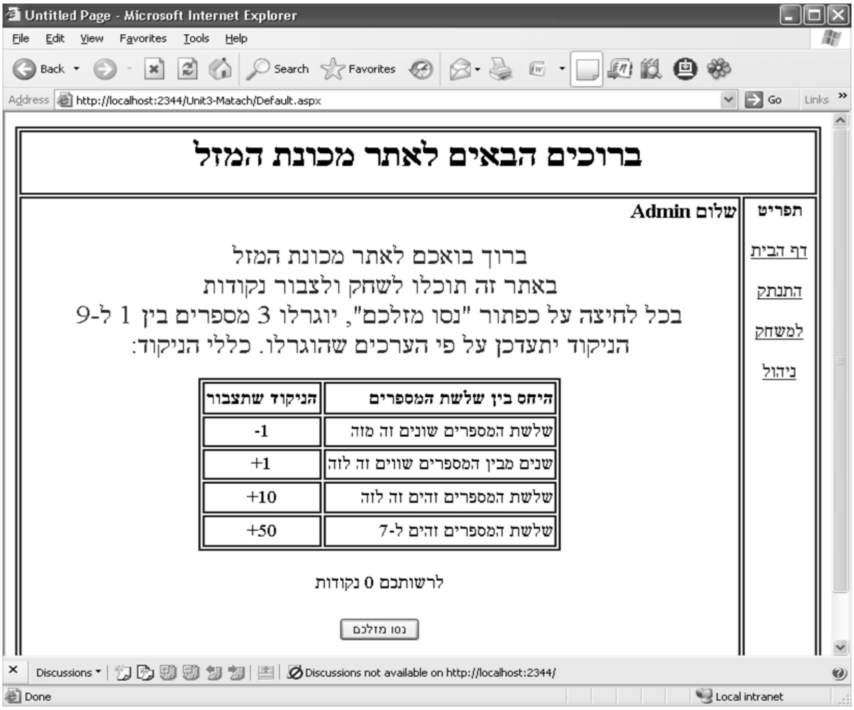
```
<!-- שורת הלוגו, המורכבת משתי עמודות -->
  <td colspan="2" style="height:90px"  dir="rtl">
     <br /><br />
     <h1>ברוכים הבאים לאתר מכונת המזל</h1>
   </td>
</tr>
<tr style="height:500px">
  <td style="width: 130px">
  <!—עמודת התפריט, קישורים לדפי האתר השונים על-פי הרשאות המשתמש -->
   <div>
     <h2>תפריט</h2>
       <!--כאן יופיעו הקישורים על-פי ההרשאות השונות-->
       <!--יש לבדוק אם המשתמש רשום / מנהל / לא רשום-->
      <a href ="Index.aspx">התחבר</a><br />
      <%
      if (Session["userName"] == null)
           Response.Write("<a href ='Login.aspx'>התחבר</a><br />");
      else
      {
           Response.Write("<a href ='Logout.aspx'>התנתק</a><br />");
           Response.Write("<a href ='Game.aspx'>למשחק</a><br />");
      }
      if(Session["admin"]!=null && (bool)Session["admin"]==true)
           Response.Write("<a href ='Admin.aspx'>ניהול</a><br />");
      %>
   </div>
  </td>
  <td>
  <!-- *************************************************** -->
  <%--  תא התוכן המשתנה מדף לדף  --%>
  <!-- *************************************************** -->
  <!--  כאן יוצג המידע המתאים לכל דף -->
  <!-- *************************************************** -->
  </td>
```

```
  </tr>
 </table>
 </div>
 </form>
</body>
</html>
```

שלב ב'

- הוסיפו לפרויקט את דפי ה-aspx הדרושים: Game.aspx ,Logout.aspx ,Login.aspx, Admin.aspx.
- העתיקו את התוכן הנמצא בין התגיות <body> ו-</body> של דף הבית לכל אחד מן הדפים המוצגים ללקוח; שנו את דף התוכן המשתנה כך:
  - o בדף ההתחברות צרו טופס המכיל שדות המיועדים לרישום שם המשתמש וסיסמתו וכן כפתורי submit , reset.
  - o בדף המשחק שבצו את המשחק כפי שעשיתם בקובץ Game1a.aspx.
  - o בדף הניהול הציגו למנהל האתר את מספר המשתמשים באתר ואת סך כול הנקודות שצברו המשתמשים. הנקודות של המשתמש יתווספו לסך הנקודות בעת יציאתו של המשתמש מהאתר.
  - o דף ההתנתקות אינו מוצג למשתמשים ועל כן אינו צריך להיות זהה במבנהו לדפי האתר.
- נוסף על עיצובו של כל דף, עליכם לכתוב את התסריט המתאים לאירוע Page_Load() של כל דף.

לפניכם דף המשחק כפי שהוא מוצג למנהל האתר:



איור 3-18

דף המוצג למנהל האתר

נספח ג':

יצירת דף aspx בסביבת Visual Studio

נספח זה מתאר כיצד יוצרים קובץ aspx ומריצים אותו כדף דינמי מתוך שרת בסביבת העבודה Microsoft Visual Studio Express 2008[6].

א. פתיחה של יישום השרת

בחרו בתפריט ראשי File ← New ← Web Site. בחלון שנפתח עשו את הפעולות האלה:

- בחרו ביישום רשת מסוג ASP.Net Web site
- קבעו את שם היישום ואת המקום שבו תשמרו אותו, לדוגמה, D:\asp\htmltest.
  אם הספרייה אינה קיימת, תופיע שאלה אם ליצור ספרייה. יש לבחור במקש OK.
- הגדירו את שפת התכנות Visual C#.

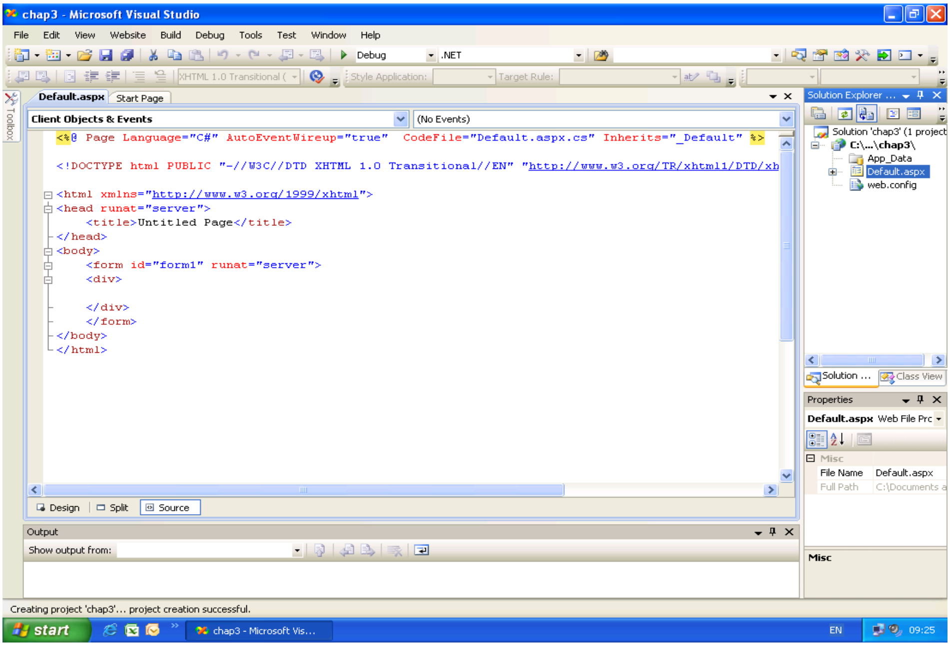
ב. יצירת קובץ aspx

לאחר שתגדירו יישום רשת חדש, תיפתח סביבת העבודה (איור 3-19) המכילה דף ASP.
סביבת העבודה מכילה כמה חלונות. חלון Solution Explorer (הנמצא לרוב בצד הימני של המסך) מציג את הקבצים שהיישום מכיל ובאילו מדריכים יישמרו הקבצים.
אם אינכם רואים את החלון הזה, בחרו בתפריט הראשי View ← Solution Explorer.

החלון הראשי הוא חלון העריכה המכיל תבנית של קובץ aspx. שם הקובץ הוא Default.aspx.

- כדי לשנות את שם הקובץ, בחרו: File← Save Default.aspx as
- כדי ליצור קובץ aspx חדש בתפריט ראשי, בחרו ב-
  File ← New← File ← Web form.

[6] סביבת פתוח זו ניתן להוריד מאתרי מיקרוסופט בכתובת http://www.microsoft.com/express/Downloads/



איור 3-19
סביבת העבודה של יישום שרת

ג. הרצה של קובץ aspx

- בתפריט הראשי בחרו :

Debug ← Start Without Debugging

ד. הפרדה בין החלק העיצובי ובין הקוד שמאחור (code behind)

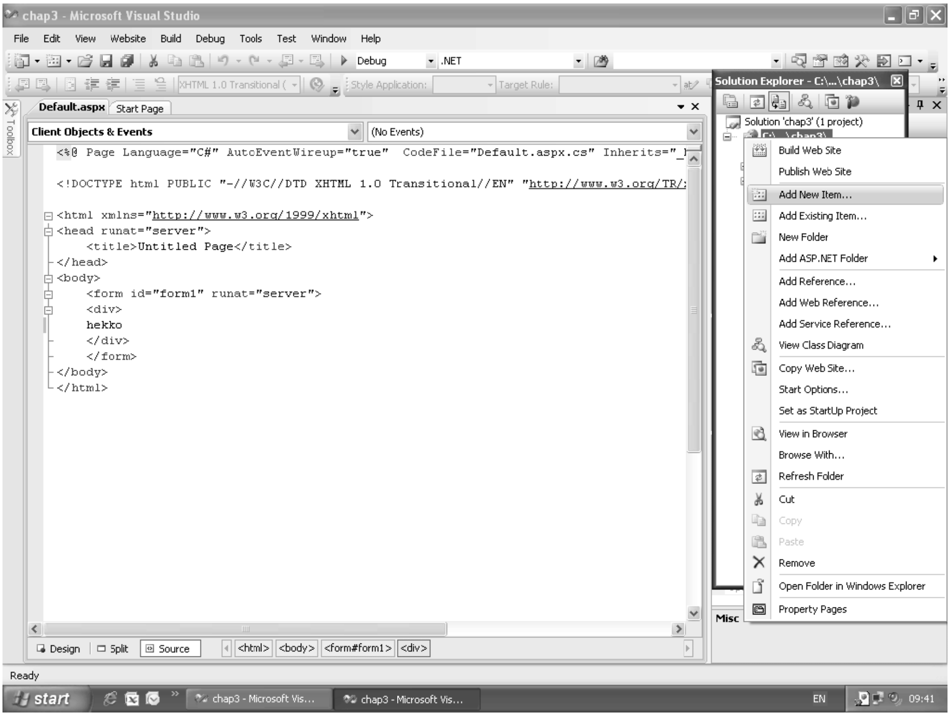
סביבת העבודה Visual Studio מאפשרת לחלק את תוכן דף ה-aspx לשני קבצים נפרדים : הקובץ האחד, שהסיומת שלו היא aspx, יכיל את החלק העיצובי (תגי HTML והתסריטים להכנת תגובת HTTP). הקובץ השני, שהסיומת שלו aspx.cs, יכיל את הגדרת התסריטים ואת החלק הביצועי וייקרא הקוד שמאחור (code behind).

בדוגמה שלנו, נשתמש בתכנית Game1.aspx ונתאר כיצד לחלק את הדף לשני קבצים. בקובץ שיכיל את החלק העיצובי שבו אנו מכינים את תגובת ה-HTTP והוא מתחיל מהתג

<html> ועד סיומו. הקובץ שיכיל את הקוד שמאחור יכלול את התסריט הכתוב בין התגים <script> ו- </script>.

יצירת קובץ המכיל את הקוד שמאחור (איור 3-20):

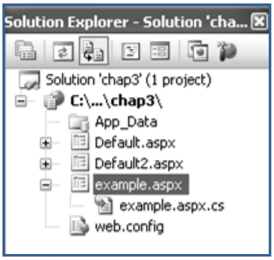
- בחלון Solution Explorer הקליקו עם הלחצן הימני בעכבר על שם הפרויקט.
- בחלון שנפתח בחרו: Add New Item.
- בחלון הנוסף שנפתח בחרו: Web Form.



איור 3-20

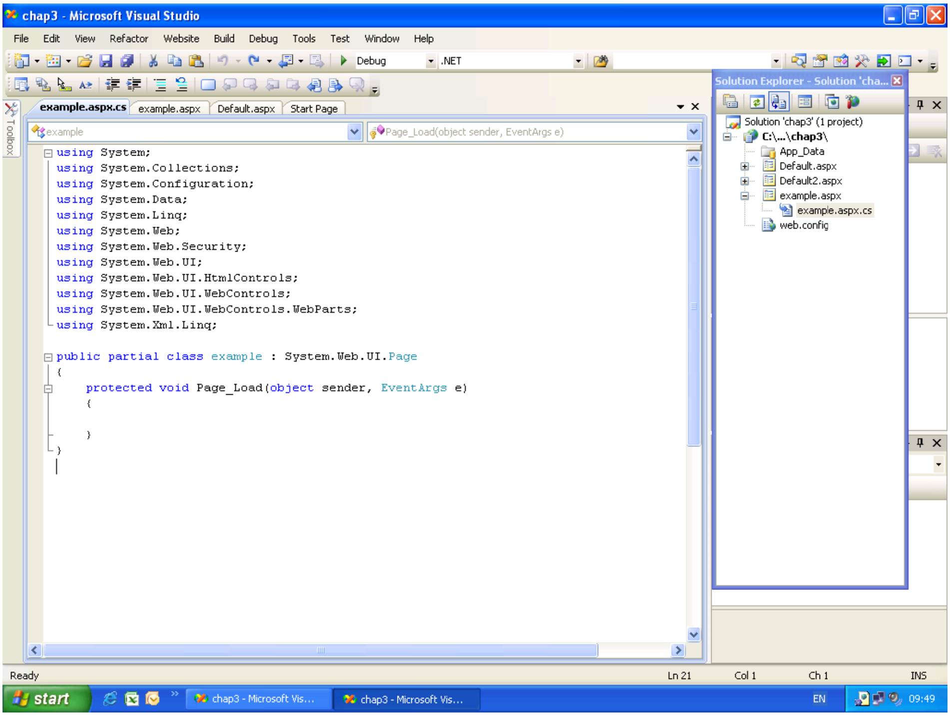
יצירת קובץ aspx המכיל את הקוד שמאחור

בחלון Solution Explorer ניתן לראות שנוצרו שני קבצים: לחצו על הסימן '+' שליד שם הקובץ.



איור 3-21
יצירת קובץ aspx וקובץ aspx.cs

- הקליקו על שם הקובץ עם הסיומת aspx.cs ותופיע התבנית של הקובץ כמתואר באיור 3-22.



איור 3-22
תבנית הקובץ של הקוד שמאחור

סביבת Visual Studio יצרה תבנית של מחלקה שבה נוכל להוסיף את הקוד שלנו.

- הוסיפו את הקוד של תכנית Game1.aspx לקובץ זה ושמרו

```
public partial class example : System.Web.UI.Page
{
   public int points, n1, n2, n3;                        //  הגדרת משתנה מטיפוס שלם לאחסון הניקוד
   protected void Page_Load(object sender, EventArgs e)
   {
      Random rnd = new Random();                   //  הגדרת עצם ליצירת מספר אקראי
      n1 = rnd.Next(1, 10);                                //  הגרלת מספרים אקראיים
      n2 = rnd.Next(1, 10);
      n3 = rnd.Next(1, 10);
      // חישוב הניקוד
      if ((n1 % 2 == 0) && (n1 % 2 == 0) && (n1 % 2 == 0))  //  בדיקה האם שלושת המספרים זהים
         points = 5;
      else
         points = 2;
   }
}
```

- לסיום שמרו את הקובץ עם הסיומת cs.
- כעת פתחו את הקובץ המכיל את החלק העיצובי על-ידי הקלקה על הקובץ עם הסיומת aspx.
- בתבנית דף aspx רִשמו את כל התגים הרשומים בחלק העיצובי של הקובץ Game1.aspx, החל מהתג <html >
- לסיום שמרו קובץ זה.

להלן קובץ Example.aspx שהתקבל:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Example.aspx.cs"
Inherits="Example" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
```

```
    <title>פרק 3 שאלה 1</title>
    <style type="text/css">
        div{text-align:center}
        h1 {color:red}
        p  {color:Purple; font-size:larger;
            text-align:right; direction=rtl}
    </style>
</head>
<body>
    <form id="form1" action="Example.aspx" runat="server">
    <div>
        <h1>ברוכים הבאים לאתר מכונת המזל</h1>
        <p>
          באתר זה תוכלו לשחק במכונת המזל ולצבור נקודות.<br />
          בכל לחיצה על הכפתור "נסו מזלכם", יוגרלו 3 מספרים בין 1 ל-9.<br />
          ניקודכם  יתעדכן על-פי הערכים שהוגרלו. <br />
          אם הוגרלו 3 מספרים זוגיים תזכו ב-5 נקודות.<br /><br />
       </p>
      <input type="submit" value="נסו מזלכם" name="Send"/>
      כפתור שליחת הנתונים
      <br /><br />
      <% Response.Write("המספרים שהוגרלו הם:  " + n1+ " "+n2+" "+ n3+ "<br />");
         Response.Write("זכית בהגרלה האחרונה" +points+"נקודות"); %>
    </div>
  </form>
</body>
</html>
```

וקובץ Example.aspx.cs שהתקבל:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
```

```
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class example : System.Web.UI.Page
{
    public int points, n1, n2, n3;                  // הגדרת משתנים מטיפוס שלם לאחסון הניקוד
    protected void Page_Load(object sender, EventArgs e)
    {
        Random rnd = new Random();           // הגדרת עצם ליצירת מספר אקראי
        n1 = rnd.Next(1, 10);                // הגרלת מספרים אקראיים
        n2 = rnd.Next(1, 10);
        n3 = rnd.Next(1, 10);
        // חישוב הניקוד
        if ((n1 % 2 == 0) && (n1 % 2 == 0) && (n1 % 2 == 0))    // בדיקה אם שלושת המספרים זהים
            points = 5;
        else
            points = 2;
    }
}
```

שימו לב: כדי שיהיה ניתן לגשת אליהם מקובץ אחר עלינו להגדיר משתנים ופונקציות להיות פומביים (public).

בנוסף, שימו לב כי ההנחיה הראשונה בקובץ עם הסיומת aspx

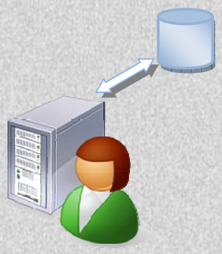
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Example.aspx.cs"
Inherits="Example" %>
```

כוללת הפניה לקוד שמאחור ואפשרות גישה לפעולות ומשתנים של הקוד שמאחור מן הדף המכיל את החלק העיצובי.

- כדי להריץ את היישום ב בתפריט הראשי בחרו :
  Debug ← Start Without Debugging

פרק 4

תכנות שרת –
שימוש במסדי נתונים



4.1 מבוא

בפרק הקודם למדנו כיצד לשמור כמות קטנה של מידע. לדוגמה, באתר 'מכונת המזל' שמרנו לכל משתמש את הניקוד שלו, ועדכנו אותו במידע הזה בכל התקשרות. ואולם, לעתים קרובות אתרי האינטרנט שומרים מידע רב, לדוגמה, אתרי קניות ששומרים מידע על פריטים שניתן לקנות ועל הלקוחות שערכו קניות דרך האתר, אתרי שידוכים ששומרים מידע על המועמדים לשידוך, אתרי משחקים שמנהלים ליגות ותחרויות משחקים וששומרים מידע על השחקנים והמשחקים, וכן אתר של אוניברסיטה השומר מידע על הקורסים, על המרצים ועל התלמידים.

אחד האמצעים לשמירת מידע רב ולאורך זמן הוא **מערכות ממוחשבות לניהול מסדי נתונים** (**Data Base Management System – DBMS**). בשוק קיימות מערכות ממוחשבות לניהול מסדי נתונים של חברות שונות, כגון Access, MySQL, Oracle, SQL Server. **מסד נתונים** (**Data Base – DB**) הוא אוסף של מאגרי מידע ממוחשבים המכילים את כל המידע הקשור לנושא מסוים והמנוהלים במרוכז, כיחידה אחת. מערכות ממוחשבות לניהול מסדי הנתונים הן תוכנות לניהול מסד נתונים, המאפשרות למשתמשים לגשת בצורה קלה ופשוטה לנתונים שמאוחסנים בו. מערכות לניהול מסד נתונים מאפשרות לעשות את הפעולות האלה: לבנות מאגרי מידע, לעדכן את מאגרי המידע ולהפיק מהם מידע.

היכן יישמר מסד הנתונים – בשרת או במחשב של הלקוח? לרוב, ארגונים רבים שומרים את מסדי הנתונים שלהם בשרת המשמש למטרה זו בלבד. הלא הוא שרת הנתונים. בצורה זו המידע מרוכז במקום אחד, ניתן לתחזקו, לעדכנו ולגבותו ובכך להבטיח את גמישות הנתונים. כמו כן, ניתן לאבטח את הנתונים על-ידי מתן הרשאות גישה שונות למשתמשים שונים. המידע במסדי הנתונים הוא זמין, וניתן לגשת לנתונים מכל מחשב.

אף-על-פי שארגונים משקיעים מאמץ בשמירת המידע, קורה שמידע אובד, אם בעקבות 'נפילה' של שרת הנתונים, אם בעקבות פריצה לשרת ואם בעקבות טעויות של עובדים. לשם כך, קיים לכל מסד נתונים **יומן** (הנקרא קובץ log) שבו נרשמות כל התנועות שנעשו במסד הנתונים, ובעזרתו ניתן לשחזר את המידע במקרה של אובדן המידע. הקובץ הזה שומר את כל הפעולות שנעשו על הנתונים. נוסף על כך, ארגונים גדולים מחזיקים גם שרת 'רדום', המכיל עותק מעודכן של מסד הנתונים. שרת זה 'מתעורר' כאשר שרת הנתונים 'נופל', או כאשר העומס על שרת הנתונים רב. לשיטת גיבוי הנתונים הזאת (שרת רדום) קוראים **מַרְאָה** (**mirror**) שכן השרת הרדום הוא בבחינת מראה של מסד הנתונים.

לסיכום, היתרונות העיקריים של מסד הנתונים הם:

- ריכוז המידע במקום אחד
- אמינות הנתונים
- זמינות הנתונים
- גמישות באחזור נתונים
- אבטחת הנתונים
- גיבוי ושחזור הנתונים

4.2 המודל הטבלאי

ישנם מודלים אחדים לייצוג מסדי נתונים. הנפוץ ביותר כיום הוא **המודל הטבלאי** (הנקרא גם '**המודל היחסי**'). במודל הטבלאי כל הנתונים מיוצגים באמצעות טבלאות וקשרים בין טבלאות. בספר הזה נעסוק בטבלה אחת, ולכן לא נעמיק בכל הקשור לקשרים בין הטבלאות. מבנה הטבלאות הוא אחיד: כל שורה בטבלה מייצגת ישות אחת (שעליה אנו שומרים מידע) וכוללת נתונים על תכונותיה. כל השורות בטבלה מייצגות קבוצה של ישויות.

לדוגמה, נתבונן באיור 4-1. מוצגת בו דוגמה לטבלה של תלמידים שעשויה להיות חלק ממסד נתונים של בית-הספר. תוכלו לראות שלכל הישויות בקבוצת התלמידים מוגדרות

תכונות משותפות: מספר הזהות, השם, השכבה, הכיתה, הרחוב והעיר. לכן נוח לייצג כל ישות (תלמיד) כשורה בטבלה.

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

איור 4-1
טבלה המייצגת את קבוצת הישויות – תלמידים

הטבלה מייצגת אפוא **קבוצת ישויות**. כל עמודה (הנקראת גם **שדה**) בטבלה מייצגת תכונה אחת של קבוצת הישויות. בתוך המשבצות של הטבלה רשומים הערכים של התכונות (הנתונים). כמובן, בעמודה של מספר הזיהוי לכל תלמיד יש ערך שונה. ערך זה מבחין בינו לבין תלמידים אחרים המיוצגים בטבלה.

לשורה בטבלה נהוג לקרוא גם **רשומה**. כל רשומה מורכבת משדות (עמודות) – כל **שדה** מכיל נתון אחד (ערך של תכונה אחת). לדוגמה, הרשומה הראשונה בטבלה מכילה את: הערך '1544' (שנמצא בשדה 'מספר זהות'), הערך 'יוסי כהן' (שנמצא בשדה 'שֵם'), הערך 'יב' (שנמצא בשדה 'שכבה') וכך הלאה. במילים אחרות, כל שדה בטבלה מייצג **תכונה** אחת מאוסף של כל התכונות המיוצגות. לכל שדה יש שם וטיפוס נתונים. באמצעות הגדרת השמות, השדות וטיפוסיהם מגדירים את מבנה הטבלה. מבנה זה נקרא גם '**סכימה** של טבלה'. שימו לב, הניסיון לרשום בשדה מסוים נתון מטיפוס אחר יגרום לשגיאה.

להלן טיפוסים אופייניים של נתונים:

- **טקסט** – רצף של תווים (אותיות, ספרות רווחים וכדומה) המכיל שם או תיאור.
- **מונה** – מספר סידורי המזהה כל שורה בטבלה. בדרך-כלל זהו מספר רציף ייחודי הגדל ב-1 באופן אוטומטי עם כל הוספה של רשומה חדשה לטבלה.
- **מספר** – מספר כלשהו, לדוגמה מספר שלם או מספר ממשי.
- **תאריך** – במבנה של יום, חודש ושנה.
- **מטבע** – מספר שלם המבטא ערך כספי.
- **כן/לא** – נתון שיכול להכיל שני ערכים בלבד, לדוגמה: אמת ושקר.
- **קול** – קטע קול המאוחסן במחשב במבנה מוגדר מראש.
- **תמונה** – תמונה המאוחסנת במחשב.
- **מזכר** – טקסט ארוך שמשתמשים בו כאשר רוצים להגדיר תכונה שערכיה הם טקסטים ארוכים (מאות תווים).

נהוג להשתמש בתכונה אחת, או בצירוף של כמה תכונות, לזיהוי חד-משמעי של ישות אחת בטבלה. לתכונה (או קבוצת תכונות) המשמשת לזיהוי חד-משמעי של הישות בטבלה קוראים **מפתח** (**key**). בטבלה שלעיל נוכל לבחור בתכונה **מספר זיהוי** כמפתח. אם בטבלה קיימים מספר מפתחות, נבחר אחד מהם כ**מפתח ראשי** (**primary key**). פעולות חישוב וחיפוש רבות נעשות על המפתח הראשי. כיוון שקל יותר לערוך חישובים על מספרים מאשר על מחרוזות, נהוג להגדיר מפתח ראשי באמצעות תכונה מספרית.

שאלה 4.1

הציעו מבנה של טבלה שבה יישמרו נתוני המשתמשים באתר 'מכונת המזל'. הטבלה צריכה להכיל את הנתונים האלה: שמו הפרטי של המשתמש, שם המשתמש שלו, הסיסמה, הגיל, המין, מספר הכניסות לאתר וסה"כ הנקודות שצבר. הגדירו את המפתח הראשי של הטבלה הזאת.

4.3 מבוא לשפת שאילתות מובנות (SQL)

כיצד מאחזרים נתונים ממסד הנתונים? משפט תכנות שמאחזר נתונים ממסד נקרא **שאילתה** (**query**). התוצאה של שאילתה היא טבלה המכילה את המידע המבוקש. המבנה של טבלת התוצאה יכול להיות זהה למבנה של הטבלה שעליה התבצעה השאילתה או להכיל רק חלק מן העמודות או השורות. כל מערכת DBMS כוללת שפת שאילתות שמאפשרת לאחזר ולהפיק מהמסד מידע לפי חתכים שונים. לדוגמה, באמצעות טבלה שמכילה את נתוני התלמיד נוכל לקבל את המידע שלהלן:

- השמות של כל התלמידים בשכבת י"ב
- השמות של כל התלמידים בשכבת י' שגרים בתל-אביב

כדי להפיק מידע ממסד הנתונים במודל טבלאי, אנו משתמשים בשפה הנקראת '**שפת שאילתות מובנות**' או בקיצור **SQL** (**Structured Query Language**). שפה זו מאפשרת לגשת לנתונים מבלי להתייחס לאופן שבו הם נשמרים במסד הנתונים. בשפה זו נגדיר מה אנו רוצים לעשות על מסד הנתונים – לשלוף מידע או לעדכן מידע. נגדיר באילו טבלאות נמצא המידע, אילו תכונות אנו מעוניינים לשלוף או לעדכן ומהם התנאים שתכונות אלו צריכות לקיים.

שפת SQL פותחה על-ידי חברת IBM בשנות ה-80 של המאה ה-20. כיום, כל חברה המפתחת מערכת לניהול של מסד נתונים כוללת גם את שפת SQL המבוססת על תקנים שנקבעו[1], אך לעיתים מרחיבה את השפה התקנית. כך נוצרו מספר "ניבים" של השפה וקיימים הבדלים בצורת הכתיבה של השאילתות בשפת SQL בין סביבה לסביבה.

את הפעולות לטיפול בנתונים ניתן לסווג לארבע קבוצות:

אחזור	SELECT	שליפת נתונים ממסד הנתונים
הוספה	INSERT	הוספת רשומות תוך כדי שמירה של כללי התקינות

---

[1] הסטנדרט לשפת SQL התקבל ב- 1987 על ידי ארגון התקנים הבינלאומי (International Organization for Standardization (ISO)). ראו באתר http://www.jcc.com/sql.htm

עדכון	UPDATE	עדכון רשומות קיימות, תוך כדי שמירה על כללי התקינות
מחיקה	DELETE	מחיקת שורות

כמו כן קיימות פעולות להגדרה ותחזוקה של מסד הנתונים למשל: CREATE , DROP.

שפת SQL אינה רגישה לגודל האות, ולדוגמה אם הפקודה UPDATE תיכתב כך: update, או כך UpDate או בשילוב אחר של אותיות קטנות וגדולות – המחשב יבין את הפקודה הזאת כפקודת עדכון. נהוג לרשום את המילים השמורות, כגון INSERT, SELECT, BETWEEN, OR, AND, LIKE, WHERE, UPDATE, DELETE, באותיות גדולות, ואת התכונות וערכיהן באותיות קטנות. נקפיד לקרוא לתכונות בשמות הזהים למשתנים המתאימים באתר. לפיכך, שם של תכונה יתחיל באות קטנה, וכל מילה חדשה תתחיל באות גדולה. בצורה זו userName יהיה שם התכונה המאחסנת את שם המשתמש במסד הנתונים, וכן שם המשתנה בדפי האתר אשר יאחסן את שמו של המשתמש. אם באתר קיים גם Session, השומר את שמו של המשתמש, גם ל-Session נקרא userName. דרך זו מקלה את ניפוי השגיאות באתר שאנו מקימים.

שפת SQL

בדוגמאות שבסעיף הזה נתייחס למסד נתונים המכיל מידע על המשתמשים באתר 'מכונת המזל'. מסד הנתונים הזה מורכב מטבלה אחת בשם tblPlayers, המכילה מידע על המשתמשים הרשומים באתר. הטבלה הבאה מציגה את התכונות המאפיינות כל אחד מהמשתמשים באתר המשחק:

טבלה 4-1 מבנה של טבלת המשתמשים באתר 'מכונת המזל'

שם השדה	תיאור
userID	המספר המזהה של המשתמש (**מפתח ראשי**)
userName	שם המשתמש
password	הסיסמה
firstName	השם הפרטי

lastName	שם המשפחה
address	הכתובת
gender	מין – הערך female (נקבה) או הערך male (זכר)
birthYear	שנת הלידה

בכתיב מקוצר נוכל לרשום את המבנה של הטבלה בצורה הזאת:

tblPlayers (<u>userID</u>, userName, password, firstName, lastName, address, gender, birthYear,)

בשיטת הכתיבה הזאת המפתח הראשי של הטבלה מודגש בקו תחתון.

א. שאילתה לאחזור נתונים – SELECT

הפעולה העיקרית המתבצעת על מסד נתונים היא אחזור (שליפת) הנתונים. שאילתת SQL לאחזור נתונים מורכבת משלושה רכיבים עיקריים – WHERE, FROM, SELECT. המבנה של שאילת אחזור הוא:

SELECT < שמות התכונות שיוצגו בתוצאת השאילתה >

FROM < שם הטבלה או הטבלאות שמהן יילקחו הנתונים >

WHERE < התנאים שהרשומות צריכות לקיים בתוצאת השאילתה >

תוצאת השאילתה יכולה להכיל שורות זהות. בכדי לבטל שורות כפולות וזהות מוסיפים את המילה DISTINCT אחרי המילה SELECT.

אחרי המילה SELECT אפשר לרשום במפורש את שמות השדות שמבקשים לאחזר, או לרשום * (כוכבית) המציינת שליפה של כל השדות של הטבלה.

בפסוקית WHERE ניתן לעשות את הפעולות האלה:

- להשתמש בקשרים הלוגיים AND, OR, ו-NOT
- להשתמש בפעולות ההשוואה <>, >=, <=, >, <, =
- לערוך השוואות בין שני ערכים (BETWEEN)

- להתאים מחרוזות על-ידי השימוש במילה LIKE. כדי להתאים בין תת-מחרוזות נשתמש בתו אחוז – % ; כדי להתייחס לתו מסוים כאל כל תו, נשתמש בתו קו תחתון – (_)
- לחפש בתוך קבוצת ערכים (IN)

דוגמאות לשאילתות אחזור

בדוגמאות שבסעיף להלן נתייחס לטבלת המשתמשים tblPlayers הבאה :

מספר	שם משתמש	סיסמה	שם פרטי	שם משפחה	כתובת	מין	שנת לידה
1	Yaron	yyy	Yaron	Zehavi	Electric Cave	male	1952
2	Elimelech	eee	Elimelech	Zorkin	Israel	male	1939
3	Potter	ppp	Harry	Potter	Hogwart	male	1986
4	Weasley	www	Ron	Weasley	Hogwart	male	1986
6	Granger	ggg	Hermione	Granger	Hogwart	female	1986
7	Bilbo	bbb	Bilbo	Baggins	Middle Earth	male	2890
8	Gandalf	zzz	Gandalf	Wizard	Middle Earth	male	NULL
14	Alice	eee	Alice	Liddell	Wonder Land	female	1865

איור 4-2
טבלת המשתמשים

1. אחזר את כל הנתונים בטבלת המשתמשים :

```
SELECT * FROM tblPlayers
```

תוצאת השאילתה הזאת היא טבלה זהה לטבלת המשתמשים.

2. אחזר את נתוני כל המשתמשים ממין נקבה :

```
SELECT *
FROM tblPlayers
WHERE gender='female'
```

כאשר מנגנון להרצת שאילתות מבצע שאילתה זו, הוא עובר על כל השורות של הטבלה tblPlayers, ועבור כל שורה הוא משווה את ערך התכונה gender עם הערך female. אם קיים שוויון, השורה מועברת לטבלת התוצאה של השאילתה.

3. אחזר את נתוני כל המשתמשים שגילם פחות מ-15 שנה[2]:

```
SELECT *
FROM tblPlayers
WHERE (2009-userBirthYear) < 15
```

4. אחזר את שמות המשתמשים ששמם הפרטי מתחיל באות H:

```
SELECT userName
FROM tblPlayers
WHERE firstName LIKE 'H%'
```

המילה LIKE מטרתה להציג את כל השדות המכילים ערך המתחיל באות/מחרוזת מסוימת, המסתיים באות/מחרוזת מסוימת או המכיל בתוכו אות/מחרוזת מסוימת. בדוגמה שלנו אנו מחפשים שמות פרטיים של משתמשים שמתחילים באות H.

5. אחזר את השמות ואת שנת הלידה של המשתמשים הגרים בהוגוורט (Hogwart). מיין את הנתונים על-פי שמם הפרטי, בסדר אלפביתי עולה:

```
SELECT userName, birthYear
FROM tblPlayers
WHERE address='Hogwart'
ORDER BY firstName ASC
```

הצירוף ORDER BY גורם למיון של השורות על-פי תנאי מסוים. ברירת המחדל למיון היא סדר עולה. עם זאת, ניתן לציין זאת במפורש על-ידי רישום ASC (קיצור של ascending). בדוגמה שלנו טבלת התוצאה תמוין לפי הסדר של ערכי התכונה firstName (סדר לקסיקוגרפי). כדי לציין סדר יורד נרשום DESC (קיצור של descending).

6. אחזר את שמות המשתמשים ששם משפחתם Cohen:

```
SELECT userName
FROM tblPlayers
WHERE lastName='Cohen'
```

---

[2] שימו לב בחישוב הגיל השתמשנו בערך קבוע 2009 כמייצג את השנה הנוכחית. קיימות פונקציות לאחזור תאריך ושנה אך בספר זה לא נציג אותן.

שאלה למחשבה

במסד הנתונים הקיים אין כרגע משתמשים ששם משפחתם הוא 'כהן'. כלומר אין אף רשומה המתאימה לשאילתה. מה תחזיר אם כן השאילתה?

7. אחזר את שמן הפרטי ושם משפחתן של כל המשתמשות שנולדו משנת 2000 ואילך:

```
SELECT firstName, lastName
FROM tblPlayers
WHERE birthYear > 2000 AND gender = 'female'
```

8. אחזר את נתוני המשתמשים ששם המשתמש שלהם מכיל את האות T:

```
SELECT *
FROM tblPlayers
WHERE userName LIKE  '%T%'
```

9. אחזר את נתוני המשתמשים שהכתובת שלהם מסתיימת ב-'Land':

```
SELECT *
FROM tblPlayers
WHERE address LIKE  '%Land'
```

10. אחזר את שמות המשתמשים ומינם, עבור משתמשים שהאות e היא האות הקודמת לאות האחרונה בשם המשתמש שלהם:

```
SELECT userName, gender
FROM tblPlayers
WHERE userName LIKE  '%e_'
```

שימו לב, כדי לציין כי לאחר האות e יש אות נוספת הוספנו את התו: _.

שאלה 4.2

לכל שאילתה שהוצגה רשמו את הטבלה שמתקבלת כתוצאה מהפעלתה על הטבלה שהוצגה באיור 4-2.

ב. שאילתה להוספת נתונים – INSERT

הוספת רשומה למסד הנתונים נעשית על-ידי שאילתת הוספה. ניתן לתת ערכים לכל שדות הטבלה או רק לחלק מן השדות, ובלבד שכל שדה שאינו יכול להיות ריק יקבל ערך. שדה מהטיפוס מונה יקבל את ערכו אוטומטית (על-ידי מנהל מסד הנתונים), ואין לתת לו ערך בעת כתיבת השאילתה.

בעת ההגדרה של שאילתת ההוספה יש לרשום את שמות השדות שאליהם יוכנסו הערכים, ואחר-כך לפרט את הערכים לפי סדר השדות שהוגדר. שימו לב, הערך של כל שדה חייב להתאים לטיפוס הנתונים של אותו השדה. לדוגמה, לשדה מספרי יש להכניס ערך מספרי. המבנה של שאילתת הוספה הוא:

```
INSERT INTO tblName(field1, ..., fieldn) VALUES (רשימת הערכים לפי סדר השדות)
```

נדגים להלן כיצד משתמשים במשפט INSERT INTO כדי להכניס רשומה חדשה לטבלה בשם tblPlayers.

א. הכנסת רשומה חדשה לטבלה עם פירוט הערכים של כל השדות האפשריים:

```
INSERT INTO tblPlayers (userName, password, firstName, lastName, address, gender,
birthYear) VALUES    ('Israel', 'myPass', 'Israel', 'Israeli', 'Herzelia', 'M', 1948)
```

כאשר מכניסים רשומה חדשה ומפרטים את כל ערכי השדות הקיימים בטבלה, ניתן לוותר על הפירוט של רשימת השדות, ובלבד שמכניסים את הערכים על-פי סדר ההופעה של העמודות בטבלה:

```
INSERT INTO tblPlayers  VALUES ('Israel', 'myPass', 'Israel', 'Israeli', 'Herzelia', 'M',1948)
```

ב. ניתן גם להכניס רשומה חדשה לטבלה מבלי להכניס ערכים לכל השדות:

```
INSERT INTO tblPlayers (userName, password, lastName, birthYear)
        VALUES ('Israel', 'myPass', 'Israeli',  1948)
```

הערה: הוספת מידע למסד אינה דורשת ביצוע פעולה מיוחד כדי לשמור אותו. המידע נשמר גם אם לא הורינו לו זאת במפורש.

ג. שאילתה לעדכון נתונים – UPDATE

שינוי של נתונים במסד הנתונים נעשה על-ידי שאילתת עדכון, באמצעות הפקודה UPDATE. עדכון יכול להתבצע על רשומה אחת, על מספר רשומות או על כל הרשומות שבטבלה.

העדכון ייעשה על שדה או שדות ברשומות המקיימות תנאי מסוים. המבנה של שאילתת עדכון הוא:

```
UPDATE tblName SET field1=value1 [field2=value2…] WHERE תנאי לוגי
```

אחרי המילה SET מופיעים ביטויי השמה במבנה: ערך = שם שדה. המנגנון לעיבוד השאילתות של ה- DBMS עובר על שורות הטבלה, ולכל שורה בודק את התנאי הלוגי שרשום אחרי WHERE. אם ערך התנאי הוא אמת, אזי מתבצעות ההשמות שרשומות אחרי המילה SET.

למשל, עדכון שמו של Moses ל-Moshe ייעשה על-ידי המשפט הזה:

```
UPDATE tblPlayers SET firstName= 'Moshe'  WHERE firstName= 'Moses'
```

דוגמה נוספת, בכדי להעלות את דירוגם של כל המשתמשים ברמת דירוג אחת נכתוב:

```
UPDATE tblPlayers SET degree= degree +1 WHERE 1=1
```

שימו לב כי בכדי לעדכן את כל הרשומות, כתבנו תנאי שהוא נכון תמיד. התנאי 1=1 נכון ללא תלות בערכי השורה, ולכן כל שורות הטבלה יעודכנו. דרך אחרת לרשום את התנאי הלוגי הזה היא: WHERE true. נוסף על כך, אין אפשרות לבטל פעולת עדכון לאחר שנעשתה.

ד. שאילתה למחיקת נתונים – DELETE

מחיקה של נתונים ממסד הנתונים נעשית על-ידי שאילתת המחיקה DELETE. ניתן למחוק רשומה אחת, כמה רשומות או את כל הרשומות שבטבלה.

מחיקת רשומה יחידה נעשית על-ידי משפט מחיקה הכולל את התנאי הנכון עבור רשומה אחת בלבד. כדי לזהות רשומה אחת בוודאות, נגדיר תנאי המתייחס למפתח או לשדה שהוגדר כייחודי. לדוגמה:

```
DELETE FROM tblPlayers WHERE userID = '1'
```

במקרה כזה, השדה userID הוא מפתח, ולכן רק שורה אחת תימחק. בצורה דומה ניתן למחוק כמה רשומות לפי תנאי. לדוגמה:

```
DELETE FROM tblPlayers WHERE address='Hogwart'
```

השאילתה האחרונה תמחוק את כל השורות שבהן ערך השדה address הוא Hogwart.

מחיקת כל הרשומות שבטבלה נעשית על-ידי המשפט הזה:

```
DELETE FROM tblPlayers WHERE true
```

או על-ידי המשפט:

```
DELETE FROM tblPlayers
```

משפטים אלה מחקו את המידע שבטבלה, אך לא את הטבלה עצמה.

למחיקת טבלה נשתמש במשפט DROP TABLE במבנה הזה:

```
DROP TABLE tblName
```

ה. שאילתות לטיפול במסד נתונים ובטבלה

עד כה הכרנו את כוחה של שפת SQL לטפל במידע (Data Manipulation Language) – הוספה / עדכון / מחיקה / שליפה. נוסף על כך, שפת SQL מכילה משפטים להגדרה ותחזוקה של המסד. ניתן לרשום משפט SQL שיוצר מסד נתונים חדש או טבלה חדשה.

כדי ליצור מסד נתונים חדש, נשתמש במשפט CREATE DATABASE אשר יציין את שם מסד הנתונים. המבנה של המשפט מוצג להלן:

```
CREATE DATABASE  (database name)
```

לדוגמה, ניצור מסד נתונים בשם game:

```
CREATE DATABASE game
```

כדי ליצור טבלה נשתמש במשפט CREATE TABLE, אשר יציין את שם הטבלה, את שמות השדות וטיפוסיהם. המבנה של המשפט מוצג להלן :

```
CREATE TABLE tblName (column1  type1 , column2  type2, ...)
```

לדוגמה, ניצור טבלה בשם tblPlayers המתאימה לאיור 4-2 :

```
CREATE TABLE tblPlayers (userID INT, userName NVARCHAR(30), password
NVARCHAR(30), firstName NVARCHAR(30), lastName NVARCHAR(30), address
NVARCHAR(30), gender BIT, birthYear INT);
```

להגדרת הטבלה tblPlayers השתמשנו בטיפוסי הנתונים האלה : NVARCHAR המציין ערך מטיפוס מחרוזת בעלת גודל משתנה, INT המציין ערך מטיפוס מספר שלם, BIT, המציין ערך מטיפוס בוליאני. לטיפוסי נתונים נוספים ראו נספח ד'[3].

שאלה 4.3

לפניכם מסד של נתונים של חברה להשכרת רכב. מסד הנתונים מורכב מטבלה אחת בשם tblRentalCar , המכילה מידע על רכבים להשכרה. לכל רכב נשמרים הנתונים האלה : מספר הרישוי (licenseNo), שנת הייצור, סוג הרכב, מספר הדלתות, מספר הנוסעים (כמה אנשים הוא יכול להכיל – 2, 5 או יותר), צבע הרכב וקטגוריית ההשכרה (A, B, C, D)

```
tblRentalCar (licenseNo, year, type, doorNo, passNo, color, category)
```

רשמו 10 שאילתות לאחזור נתונים מטבלה זו.

שאלה 4.4

בספר זה נשתמש במערכת לניהול מסד נתונים שנקראת SQL Server. נספח ד' מציג את אופן ההתקנה והשימוש בסביבת עבודה זו. השתמשו בנספח זה כדי לבנות את

---

[3] פירוט על טיפוסי הנתונים שבמסד נתונים SQL Server נמצא באתר http://msdn.microsoft.com/en-us/library/ms187752.aspx

מסד נתונים של חברה להשכרת רכב המורכב מטבלה אחת בשם tblRentalCar, כמפורט בשאלה 4.3. הוסיפו את הנתונים של 10 רכבים והפעילו את השאילתות שהגדרתם בשאלה 4.3.

סיכום

שפת SQL היא שפה לניהול נתונים שבה מגדירים במשפטי תכנות מה מבקשים לבצע על הנתונים. לשם כתיבת שאילתות SQL, יש להכיר את המבנה של מסד הנתונים – שמות הטבלאות והשדות, אך אין צורך לדעת מהו סוג מסד נתונים שבו עובדים.

4.4 עבודה עם מסד נתונים

בכדי להפיק מידע ולעדכן מסד נתונים, נשתמש בטכנולוגיית ADO.Net (ActiveX Data Object). טכנולוגיית **ADO** כוללת קבוצה של עצמים שמספקים גישה, ברמת היישום, למסד הנתונים. הטיפול בנתונים הוא אחיד, ללא תלות במערכת ניהול מסד הנתונים-DBMS (Database Management **S**ystem**)**, ונעשה באמצעות מישק. באמצעות העצמים הללו אפשר ליצור דפי אינטרנט דינאמיים שמכילים מידע שהופק ממסד הנתונים על-פי דרישת המשתמש. לדוגמה, נוכל להשתמש בטכנולוגיה ADO כדי לפנות מדף ASP למסד נתונים בשם Game למשל ולאחזר נתונים מטבלת tblPlayers שבנינו בסעיף הקודם, נוכל להציג את המשתמשים הרשומים לאתר 'מכונת המזל', להוסיף משתמשים חדשים ולעדכן את נתוניהם.

הגישה לנתונים במסד תלויה במערכת לניהול מסדי נתונים. בשנים האחרונות פותחו טכנולוגיות שמטרתן לפשט את העבודה עם מסדי הנתונים, הן בשלב הפיתוח והן בשלב התחזוקה. טכנולוגיות אלה הן המתווך בין היישום שאנו מפתחים לבין מסד הנתונים. המתווך הזה הנקרא ***'מתפעל'*** (**driver**), והוא ייחודי לכל מסד נתונים.

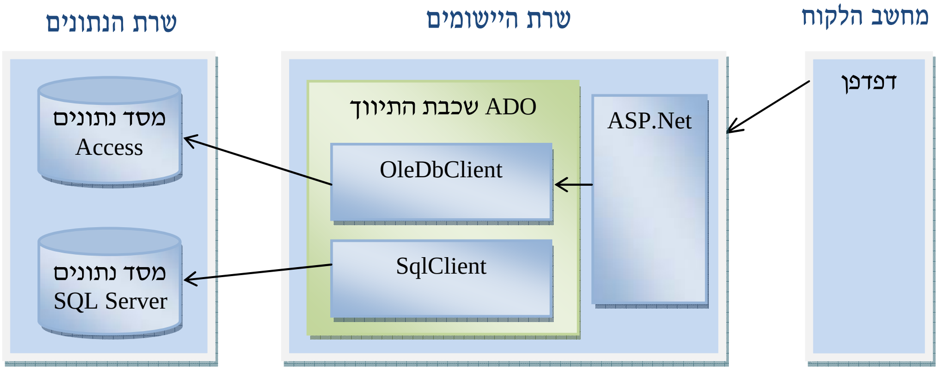
יישום טיפוסי של רשת מורכב משלושה מחשבים שונים לפחות:

1. המידע של הארגון נשמר **בשרת נתונים** שמיועד לשמירת מסדי נתונים בלבד, והגישה אליו מוגבלת.

2. השרת מורץ ב**שרת היישומים** של הארגון. בשרת הזה נשמרים דפי ה-ASP, דפי ה-HTML שמוצגים ללקוח והמתפעלים של מסדי הנתונים.
3. הלקוח רץ על מחשב שלישי אשר פונה אל שרת היישומים לשם קבלת המידע הדרוש לו.

אם המידע נשמר במסד הנתונים, שרת היישומים פונה לשרת המידע ומבקש ממנו את המידע הנחוץ לו. המידע עובר עיבוד ונשלח ללקוח.

להלן, סכמת עבודה אופיינית של יישום רשת[4]:



איור 3-4
מודל של יישום רשת המשתמש במסד נתונים

ADO.Net תומכת בשתי שיטות עבודה של יישום הרשת עם מסד נתונים:

- **מקושרת** (**connected**) – היישום מחובר באופן קבוע למסד הנתונים כך שהעיבוד נעשה על נתוני המסד עצמו.
- **לא-מקושרת** (**disconnected**) – ישום המחשב מתחבר למסד הנתונים, מאחזר את המידע ומתנתק. את העיבוד מבצע היישום על הנתונים שנשמרו אצלו. בסיום העיבוד היישום מתחבר שוב למסד הנתונים ומעדכן אותו.

---

[4] OleDBClient ו- SqlClient הם ממשקי משתמש, שתוכננו על ידי מיקרוסופט, המאפשרים גישה בצורה אחידה למגוון של מקורות נתונים.

לכל אחד מאופני העבודה עם מסד הנתונים ישנם יתרונות וחסרונות. העבודה בשיטה הלא-מקושרת קלה יותר, אינה מנצלת משאבי שרת גדולים, אך עלולה להשתמש בנתונים לא עדכניים. העבודה בשיטה המקושרת מסובכת מעט יותר, מעמיסה על השרת, אך מבטיחה שהנתונים שנעבד יהיו עדכניים. ההחלטה על הגישה למסד תלויה בצורכי הארגון.

מערכות אינטרנט רבות יאפשרו אף הן עבודה עם מסד הנתונים שלהן בשיטה הלא-מקושרת בכדי לאפשר גישה למשתמשים רבים ככל הנדרש. משתמש, המעוניין במידע מתוך מסד הנתונים של המערכת, יקבל עותק של המסד, ישלוף ממנו את הנתונים הדרושים לו, ויפנה מקום למשתמש אחר. מגבלת ההתחברות למסד הנתונים היא מספר המשתמשים בכל רגע נתון. במערכות אינטרנט נפח התעבורה הצפוי אינו ידוע מראש. ייתכנו מקרים שבהם מספר המשתמשים המנסה להתחבר למסד הנתונים הוא קטן, וייתכנו מקרים שבהם מספר המשתמשים הפונים למסד בו-זמנית ייצור עומס רב כל-כך על המערכת עד כדי פגיעה בביצועיה ואפילו עד כדי קריסת השרת. במקרים אלה לשיטה הלא-מקושרת יש יתרון, משום שזמן ההתקשרות למסד הנתונים הוא קצר מאוד.

ישנן מערכות שחייבות להיות מעודכנות בכל רגע נתון, כגון מערכת לניהול חשבונות בנק. ברור כי לא ייתכן שהבנק יעבוד עם מערכת אשר נתוניה נכונים ליום האתמול, ואף לא לדקה קודמת. מערכת ניהול בנק חייבת אפוא לעבוד בשיטה המקושרת, המאפשרת גישה לנתונים העדכניים. לפיכך, ייתכנו שתי מערכות שונות לייצוג נתוני חשבונות הבנק. מערכת אחת באינטרנט, המאפשרת ללקוחות הבנק לראות את נתוני החשבונות שלהם, ומערכת אחרת, פנימית, המיועדת לעובדי הבנק בלבד. ברור כי המערכת שבה עובדים עובדי הבנק צריכה להיות מסונכרנת עם מסד הנתונים בכל רגע נתון, ולכן היא תפעל בשיטה המקושרת. אבל המערכת המיועדת ללקוחות הבנק, המופעלת על-ידי אתר האינטרנט, יכולה לפעול בשיטה הלא-מקושרת.

כדי לפתח יישום רשת שפועל על מסד נתונים, אנו צריכים לעשות כמה פעולות. אנו נעשה הפעולות האלה באמצעות העצמים שמספקת טכנולוגיית ADO. תחילה נציג את השלבים הדרושים כדי לפתח את היישום בשיטה הלא-מקושרת, ובהמשך נתאר כיצד ניתן לפתח יישום בשיטה המקושרת. בכל שיטה נציג את תהליך העיבוד, העצמים והפעולות שבהם נשתמש, ונדגים על-ידי דוגמאות הקשורות לאתר 'מכונת המזל' שהצגנו בפרק הקודם.

4.5 תהליך האחזור ועיבוד הנתונים בשיטה הלא-מקושרת

4.5.1 יישום השיטה הלא-מקושרת

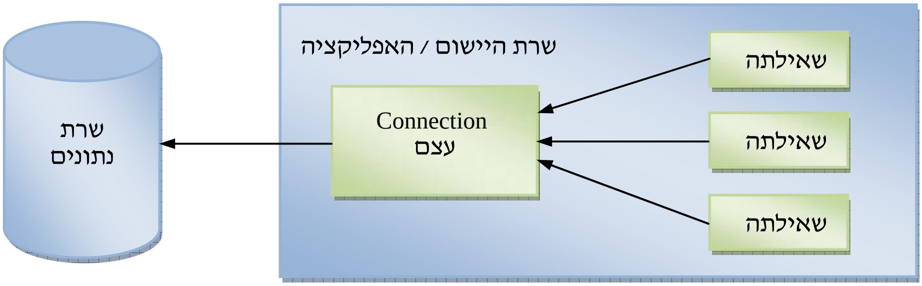
הצעדים הדרושים לשם התחברות למסד הנתונים, עיבוד הנתונים ועדכונם במערכות לא-מקושרות הם אלה:

1. התחברות למסד הנתונים
2. טעינת המידע ממסד הנתונים לזיכרון
3. התנתקות ממסד הנתונים
4. עיבוד הנתונים בזיכרון, כולל הפעולות האלה:
   - הצגת הנתונים
   - שינוי הנתונים
   - הוספת הנתונים
   - מחיקת הנתונים
5. התחברות למסד הנתונים
6. עדכון מסד הנתונים
7. התנתקות ממסד הנתונים

נתאר בהרחבה את העצמים והפעולות שנשתמש בהם כדי לממש את פעולות 1 עד 4 (כולל).

א. התחברות למסד הנתונים

כדי לעבוד עם מסד נתונים, צריך ראשית לדעת את מיקומו ואת שמו. מסד הנתונים יכול להימצא במחשב המקומי (המתפקד כשרת), על שרת מקומי של הארגון ואף על שרת בארץ אחרת. יתרה מכך, אפשר שעל אותו מחשב יהיו כמה מסדי נתונים. לפיכך, נרצה לדעת את כתובת ה-IP של מחשב השרת, את המיקום של מסד הנתונים בשרת, את סוג המסד ואת שמו. העצם המייצג מידע והמאפשר לגשת אליו הוא מטיפוס המחלקה Connection. עצם מטיפוס המחלקה הזאת מנהל את ההתחברות למסד הנתונים.



איור 4-4
התחברות למסד נתונים

אחת מתכונותיה של המחלקה Connection מוגדרת על-ידי מחרוזת ההתחברות למסד הנתונים. מחרוזת ההתחברות (Connection String) מגדירה את מיקומו, את שמו ואת סוגו של מסד הנתונים. בעת הגדרת ההתחברות למסד הנתונים, יש להעביר את מחרוזת ההתחברות לבנאי של עצם ההתחברות. למחלקה זו יש שתי פעולות חשובות. האחת יוצרת התחברות למסד הנתונים והשנייה מאפשרת התנתקות ממנו.

סוג ה-DBMS קובע עם אילו עצמים ספציפיים נעבוד. למרבית מערכות ה-DBMS קיימות מחלקות המאפשרות עבודה ייחודית עם רכיבים ספציפיים שעברו אופטימיזציה המתאימה למסד הנתונים שבהן הן מטפלות. לדוגמה, בעבודה עם מערכת SQL Server נשתמש ברכיבים של המחלקה SqlConnection.

בעבודה עם מסד נתונים SQL Server הגדרת ההתחברות תיעשה על-ידי יצירת עצם מטיפוס המחלקה הזו:

```
SqlConnection connection = new SqlConnection();
```

נציג את המאפיינים והפעולות הדרושים בעבודה עם עצם מהטיפוס SqlConnection.

המחלקה SqlConnection	
מאפיינים (Properties)	
מחרוזת ההתחברות למסד הנתונים	ConnectionString
בנאים	
בנאי בְּרֵרת מחדל שאינו מקבל פרמטרים	SqlConnection()
בנאי המקבל כפרמטר את מחרוזת ההתחברות	SqlConnection(string connectionString)
פעולות	
פעולה האחראית על התחברות למסד הנתונים	virtual void Open()
פעולה האחראית על התנתקות ממסד הנתונים	virtual void Close()

ב. הגדרת הפקודה שנשלחת למסד הנתונים

המחלקה Command מגדירה את השאילתה הנשלחת למסד הנתונים לשם עבודה עם הנתונים. העצם מהטיפוס Command מטפל בקישור בין התכנית למסד הנתונים, כלומר משתמש בעצם מהטיפוס Connection שמתאר על מי תורץ השאילתה, ולאחר שנעשתה ההתחברות למסד הנתונים הנכון, הוא מריץ את השאילתה. ניתן לומר כי המחלקה Command עונה על השאלה: מה רוצים לעשות על מסד הנתונים? ואילו המחלקה Connection עונה גם על השאלה: על מי רוצים לבצע את הפעולה?

נציג את המאפיינים והבנאים של עצם מהטיפוס SqlCommand המתאים לעבודה עם מערכת SQL Server.

המחלקה SqlCommand	
מאפיינים (Properties)	
מחרוזת המכילה משפט SQL לביצוע	string CommandText
עצם ההתחברות למסד הנתונים	SqlConnection Connection
בנאים	
בנאי בררת מחדל	SqlCommand()

בנאי המקבל כפרמטרים מחרוזת המגדירה שאילתה ועצם התחברות	SqlCommand(string cmdTxt, SqlConnection connection)

ג. ייצוג של טבלה בזיכרון

במודל נתונים הפועל בשיטה הלא-מקושרת (disconnected), העבודה עם מסד הנתונים איננה ישירה. לפיכך, הפעלת שאילתה על מסד הנתונים מחזירה כתוצאה עצם המכיל טבלאות, עמודות, קשרים וכו'. המבנה הזה יאוחסן בזיכרון המחשב ופעולות השליפה, העדכון והמחיקה יתבצעו מול המבנה הזה, ולא ישירות מול מסד הנתונים. העצם המאחסן את המבנה הזה נקרא DataSet. בסיום העבודה יש לעדכן את מסד הנתונים כך שיכיל את כל השינויים שנעשו על ה-DataSet.

תפקיד המחלקה DataSet הוא לנהל את המידע המאוחסן בזיכרון ולבצע עליו פעולות. המחלקה אינה מכירה כלל את מסד הנתונים ואין לה יכולת גישה למסד הנתונים, ובפרט היא אינה יכולה לקרוא או לעדכן את מסד הנתונים. יתרה מכך, עצם מהטיפוס DataSet יכול להכיל מידע שמקורו אינו בהכרח מסד נתונים. למשל, כפי שנציג בפרק 6, מקור הנתונים יכול להיות קובץ XML.

מבנה המחלקה DataSet יהיה תמיד זהה לזה של מסד הנתונים (או לחלק ממנו – תלוי בתוצאת השאילתה), והתנהגותה אף היא תהיה זהה להתנהגותו, כלומר, המחלקה תקיים את אותם האילוצים אשר מקיים מסד הנתונים. לדוגמה, אם ננסה להוסיף רשומה עם מפתח ראשי שקיים כבר או ננסה לעדכן רשומה שאיננה קיימת, אזי ה-DataSet ישלח הודעת שגיאה כפי שהיה שולח מסד הנתונים.

מהו המבנה של מסד הנתונים שמייצג DataSet? מופע של המחלקה DataSet מכיל אוסף שנקרא Tables. זהו אוסף של טבלאות (אחת או יותר) שבו כל טבלה היא עצם מטיפוס המחלקה DataTable. עצם מטיפוס המחלקה DataTable מורכב מאוסף של עמודות (עצמים מטיפוס המחלקה Columns) המגדירות את מבנה הטבלה, שורות (עצמים מטיפוס המחלקה Rows) המגדירות את המידע ומפתחות שהם מערך של עמודות המייצגות את מפתח הטבלה.

ד. גישור בין מסד הנתונים ובין ההעתק שלו בזיכרון

כאמור, עצם מהטיפוס DataSet מאחסן העתק של מסד הנתונים, אך אין לו גישה אל מסד הנתונים הזה. תפקידו של העצם הזה הוא לנהל את המידע בלבד. על הקישור אל מסד הנתונים, הכולל טעינת הנתונים מן המסד לזיכרון ועדכון המסד בסיום העיבוד, אחראית המחלקה DataAdapter. עצם מהטיפוס הזה אחראי על הקשר בין מסד הנתונים לבין מאגר הנתונים השוכן בזיכרון. יתרה מכך, עצם זה יוצר את ה-DataSet.

המחלקה DataAdapter היא מחלקת שירות המספקת גישור למסד הנתונים, ובאמצעותה נבנה ה-DataSet, מתבצעות השאילתות ומעודכנים הנתונים. בכדי להשתמש בעצם מהטיפוס DataAdapter, יש להעביר לו מידע על מסד הנתונים, מיקומו והפעולות שמעוניינים לבצע עליו. לפיכך, בעת יצירת עצם מהטיפוס DataAdapter יועבר לבנאי עצם מהטיפוס Command.

למחלקה DataAdapter יש ארבעה מאפיינים המייצגים את ארבעת סוגי השאילתות (עדכון, הוספה, מחיקה, בחירה). לכל ארבעת המאפיינים מוגדר חיבור משותף, שכן כולם פונים לאותו מסד הנתונים. למעשה, מחלקה זו מגדירה את ארבעת הפעולות הבסיסיות ב-SQL ומשתמשת בעצם מטיפוס המחלקה בשם SqlCommandBuilder (המוצג בסעיף ד' להלן) בכדי לעדכן את מסד הנתונים.

נדגיש כי לעצם מהטיפוס DataAdapter יש תפקיד כפול:

1. לבנות עצם מהטיפוס DataSet אשר יכיל את המידע הנדרש מתוך מסד הנתונים ואשר יישמר בזיכרון התכנית.
2. לעדכן את מסד הנתונים בסיום העיבוד.

כמו במחלקות הקודמות, נבחר את המחלקה שבה נשתמש בהתאם למסד הנתונים שאיתו אנו עובדים: לעבודה עם SQL Server נבחר במחלקה SqlDataAdapter.

המחלקה DataAdapter כוללת שתי פעולות מרכזיות:

1. הפעולה Fill מייבאת או טוענת את הנתונים לתוך DataSet על-פי שאילתת SELECT שהגדיר העצם מהטיפוס Command. הפעולה משאירה את מצב הקישור

למסד הנתונים כפי שהיה לפני שהופעלה. אם החיבור למסד הנתונים היה סגור, הפעולה תתחיל את ההתחברות למסד ובסיום תתנתק ממנו. אם החיבור היה פתוח, אזי הפעולה תשאיר אותו פתוח.

הפעולה Fill דואגת לפתיחה ולסגירה של מסד הנתונים ואנו לא צריכים ליזום פעולות אלה. מתי בכל זאת ניזום בעצמנו את ההתחברות וההתנתקות ממסד הנתונים, ולא נסמוך על הפעולה Fill שתבצע זאת בעבורנו? כאשר נעבוד עם מספר רב של טבלאות, ונרצה להקל את העומס על השרת. אם נפעיל מספר מתאמים בזה אחר זה, אזי אם נשאיר לפעולה Fill את משימת ההתחברות אל מסד הנתונים וההתנתקות ממנו, התהליך יתרחש מספר פעמים, כמספר המתאמים שנפעיל. כדי להימנע מכך, נתחבר אל מסד הנתונים באופן יזום, באמצעות הפעולה Open, ואחר-כך נשתמש בפעולה Fill מספר פעמים כנדרש. מובן שיהיה עלינו להתנתק ממסד הנתונים באמצעות הפעולה Close, שכן הפעולה Fill לא תעשה זאת בעבורנו. הפעולה מאפשרת להגדיר שם לוגי לטבלה הנוצרת ב-DataSet. השם הזה אינו חייב להיות זהה לשם הטבלה שבמסד הנתונים.

לסיכום – פעולה זו נעשית בתחילת העיבוד ומבצעת:

- התחברות
- הרצה של הפקודה SELECT של ה-DataAdapter
- בנייה של טבלת DataSet שאליה יובאו הנתונים ממסד הנתונים, על-פי השאילתה
- התנתקות

2. הפעולה Update מאפשרת לעדכן את מסד הנתונים בהתאם לשאילתות עדכון, הוספה או מחיקה אשר התבצעו על עצם מהטיפוס DataSet. הפעולה מקבלת שני פרמטרים - הראשון הוא שם ה- DataSet שממנו מעדכנים את הנתונים, והשני הוא שם הטבלה של מקור הנתונים ב-DataSet. שימו לב: **עדכון מסד הנתונים אפשרי בתנאי שהוגדר מפתח ראשי – PrimaryKey**. המפתח הראשי הוא עצם מהטיפוס מערך של עמודות. במקרה שבו המפתח מוגדר על-ידי ערך יחיד, יוגדר מערך באורך 1 .

לסיכום – פעולה זו נעשית בסיום העיבוד והיא מבצעת:

- התחברות
- הרצה של פקודת העדכון המתאימה
- התנתקות

נסכם את המידע על המחלקה SqlDataAdapter המתאים לעבודה עם מערכת SQL Server:

המחלקה SqlDataAdapter	
מאפיינים (Properties)	
מאפיין המגדיר את השאילתה אשר על-פיה יבחרו הרשומות **שיעודכנו** במקור הנתונים	UpdateCommand
מאפיין המגדיר את השאילתה אשר על-פיה ייבחרו הרשומות **שיתווספו** למקור הנתונים	InsertCommand
מאפיין המגדיר את השאילתה אשר על-פיה ייבחרו הרשומות **שיימחקו** מה-DataSet	DeleteCommand
מאפיין המגדיר את השאילתה אשר על-פיה ייטענו הנתונים לזיכרון	SelectCommand
בנאים	
בנאי ברירת המחדל	SqlDataAdapter()
בנאי המקבל כפרמטר את השאילתה כעצם מטיפוס SqlCommand	SqlDataAdapter(SqlCommand cmd)
פעולות	
פעולה הבונה DataSet וטוענת אותו בנתונים כתוצאה מהפעלת שאילתת SELECT. הפעולה מחזירה את מספר השורות שהוטענו לזיכרון. הערה: אם הקישור למסד הנתונים סגור, הפעולה מאפשרת התחברות והתנתקות ממסד הנתונים	int Fill(DataSet ds)
פעולה הבונה DataSet וטוענת אותו בנתונים כתוצאה מהפעלת שאילתת SELECT. הפעולה מגדירה שם לוגי (הפרמטר tblName) לטבלה שיצרה, שאינו בהכרח שם הטבלה במסד	int Fill(DataSet ds, string tblName)

הנתונים. הפעולה מחזירה את מספר השורות שהוטענו לזיכרון הערה: אם הקישור למסד הנתונים סגור, הפעולה מתחילה ומסיימת את הקשר למסד הנתונים	
פעולה המעדכנת את מסד הנתונים על-פי הנתונים העדכניים הנמצאים ב-DataSet. פעולת העדכון יכולה להיות אחת מהפעולות הבאות: הוספה, שינוי וביטול. הפעולה מחזירה את מספר הרשומות שהושפעו מפעולת העדכון	int Update(DataSet ds)

אחזור נתונים ממסד הנתונים בשיטה הלא-מקושרת

תחילה נדגים כיצד ניצור דפי שרת שבאמצעותם ניתן לאחזר נתונים ממסד נתונים באמצעות שאילתות SELECT וכיצד נציג את הטבלה המתקבלת ללקוח. בכל הדוגמאות המוצגות בהמשך נשתמש בטבלה בשם tblUsers שהמבנה שלה הוא כדלקמן:

tblUsers(userID, userName, password, firstName, lastName, address, phone, cellPhone, email, gender, birthYear, isManager)

התכונות gender, isManager יוגדרו כ-bit, כלומר מטיפוס בוליאני; התכונות birthYear ,userID, יוגדרו כ-int, כלומר מספרים שלמים; התכונות userName, password, firstName, lastName, address, email יוגדרו כ-nvarchar, כלומר כמחרוזות מגודל משתנה באורך מקסימלי של 50 תווים, והשדות phone, cellPhone יוגדרו כ-nvarchar, כלומר כמחרוזות מגודל משתנה באורך מקסימלי של 10 תווים.

א. קריאת כל הרשומות שבטבלה tblUsers והצגתם לפני המשתמש

כדי לקרוא את כל הרשומות בטבלה יש להפעיל את השאילתה

```
SELECT * FROM tblUsers
```

וליצור עצם מהטיפוס DataSet שיכיל עותק של הטבלה. לסיום, נכתוב פונקציה ListOfUsers() שממירה את נתוני הטבלה למחרוזת שתישלח ללקוח כתגובת HTTP. נתאר את שלבי הפתרון של משימה זו ובסיום נציג את התכנית המלאה.

שלב 1 – הגדרה של צורת ההתחברות למסד הנתונים

בכדי לעבוד עם מידע הקיים במסד הנתונים, נגדיר תחילה את ההתחברות למסד. לשם כך נגדיר עצם מהטיפוס Connection. נבחר את המחלקה שממנה ניצור את העצם הזה, על-פי מסד הנתונים שאיתו נעבוד.

בספר זה אנו משתמשים במסד הנתונים SQL Server, ולכן ניצור עצם מהטיפוס SqlConnection בשם conn, שיגדיר את צורת ההתחברות של היישום למסד:

```
SqlConnection connection = new SqlConnection();
```

בכדי לציין לאיזה מסד נתונים היישום מתחבר, נגדיר את מחרוזת ההתחברות של העצם הזה, כלומר, נציין את סוג מסד הנתונים, מיקומו ושמו, לדוגמה:

```
connection.ConnectionString =  @"Data Source=.\SQLEXPRESS;
                              AttachDbFilename=localhost\dbName.mdf;
                              Integrated Security=True";
```

סוג מסד הנתונים

שם מסד הנתונים

השרת והנתיב שבו נמצא מסד הנתונים

הגדרת האבטחה של מסד הנתונים

<u>הערה</u>: הסימון @ לפני מחרוזת מציין כי יש להתעלם מתווים מיוחדים בעלי משמעות כמו "\n" המסמן שורה חדשה.

כמובן, ניתן קודם להגדיר את מחרוזת ההתחברות ולאחר מכן לשלוח אותה לבנאי של העצם מהטיפוס SqlConnection:

```
string connectionString =
        @"Data Source=.\SQLEXPRESS; AttachDbFilename=localhost\dbName.mdf;
              Integrated Security=True";
              SqlConnection connection = new SqlConnection(connectionString);
```

מחרוזות התחברות למסד נתונים הן בעלות מבנים שונים, התלויים בסוג המסד ובסוג ההתחברות. לעתים, נתחבר למסד נתונים הנמצא באותו שרת ובאותה התיקייה של היישום, ולעתים למסד נתונים הנמצא בשרת מרוחק המצריך שם וסיסמה. באתר – www.connectionstrings.com נמצא הגדרות למחרוזות התחברות מטיפוסים שונים ולמסדי נתונים שונים.

נבחר במחרוזות ההתחברות של ה-SQLServer. המחרוזת הראשונה היא מחרוזת קישור סטנדרטית:

Standard Security

```
Data Source=myServerAddress; Initial Catalog=myDataBase; User Id=myUsername;
Password=myPassword;
```

המחרוזת דורשת את שם המחשב, את שם ה-DB, את שם המשתמש וסיסמה.
אם עובדים ברשת פנימית מאובטחת ואין צורך בשם וסיסמה, המבנה של מחרוזת ההתחברות הוא כזה:

Trusted Connection

```
Data Source=myServerAddress; Initial Catalog=myDataBase; Integrated Security=SSPI;
```

והוא דורש את הנתונים האלה: מיקום השרת, שם ה-DB והגדרות אבטחה. על אופן יצירת מחרוזת ההתחברות ב SQL Server ראו נספח ד בפרק זה.

עתה, לאחר שהגדרנו את מסד נתונים שעמו נעבוד, נגדיר מה אנו רוצים לבצע על מסד הנתונים הזה. ניצור עצם מהטיפוס Command בשם cmd.

```
SqlCommand cmd = new SqlCommand();
```

נגדיר על-ידי העצם הזה את שאילת SQL שאנו רוצים לבצע על מסד הנתונים. את השאילתה נקבע כערך המאפיין CommandText של מחלקה זו. בדוגמה שלנו, הצגת כל המידע בטבלת tblUsers.

```
cmd.CommandText = "SELECT * FROM tblUsers";
```

כעת, נחבר בין השאילתה שאנו רוצים לבצע לבין החיבור הנדרש, כלומר נשים במאפיין Connection של העצם מהטיפוס SqlCommand את עצם ההתחברות למסד הנתונים שהגדרנו קודם לכן (בדוגמה שלנו בעצם בשם connection מטיפוס המחלקה SqlConnection) :

```
cmd.Connection = connection;
```

עתה, ניתן להתחבר אל מסד הנתונים על-ידי הפעולה :

```
connection.Open();
```

הערה : כאשר עובדים עם טבלה אחת, ניתן לוותר על הפעולה הזאת משום שהיא תתבצע כחלק מהפקודה Fill הטוענת את הנתונים לזיכרון. אם בכל זאת בחרנו להתחבר למסד הנתונים באופן יזום, אז עלינו לוודא שהתנתקנו ממנו בסיום טעינת הנתונים.

שלב 2 – טעינת המידע ממסד הנתונים לזיכרון

נניח שמסד הנתונים הריץ את השאילתה ומצא שהיא מחזירה x רשומות העונות על השאילתה. כעת, עלינו לטעון את המידע מן המסד אל היישום. לשם כך נגדיר עצם מהטיפוס SqlDataAdapter :

```
SqlDataAdapter  adapter = new SqlDataAdapter();
```

המידע שיש לטעון מוגדר בעצם מהטיפוס Command ולכן נציב במאפיין SelectCommand של העצם מטיפוס SqlDataAdapter ערך זה :

```
adapter.SelectCommand = cmd;
```

אפשר כמובן לבצע שתי פעולות אלה בו-זמנית על-ידי :

```
SqlDataAdapter adapter = new SqlDataAdapter(cmd);
```

המידע שיאוחזר ממסד הנתונים יישמר בעצם מהטיפוס DataSet. נגדיר אפוא עצם מהטיפוס הזה :

```
DataSet ds = new DataSet();
```

האחזור, העדכון או המחיקה יתבצעו כאמור על הנתונים השמורים בעצם הזה. בסיום העבודה יעדכן עצם מהטיפוס SqlDataAdapter את מסד הנתונים.

התחברות למסד הנתונים (אם החיבור לפני ביצוע הפעולה סגור), מילוי ה-DataSet בטבלאות, עמודות ומידע וכן ההתנתקות ממסד הנתונים (אם הפעולה היא זו שדאגה לחיבור), יתבצעו על-ידי הפעולה:

```
adapter.Fill(ds);
```

פעולה זו מאפשרת קישור עם מסד הנתונים, טוענת ממסד הנתונים, ששמו הופיע במחרוזת ההתחברות (בדוגמה שלנו dbUsers.mdf), את כל השורות שבטבלה tblUsers ומכניסה אותם לטבלה בשם Users בעצם ds מהטיפוס DataSet.

שימו לב כי שם הטבלה ב-DataSet אינו חייב להיות זהה לשמה במסד הנתונים.

שלב 3 – התנתקות ממסד הנתונים

במקרה שבו התחברנו אל מסד הנתונים באופן יזום, ולא השארנו זאת לפעולה ()Fill, נתנתק עתה ממסד הנתונים על-ידי הפעולה:

```
connection.Close();
```

שלב 4 – עיבוד הנתונים בזיכרון

בשלב הזה יש באפשרותנו לעבד את הנתונים כרצוננו. אנו יכולים להציג את הנתונים על-פי חתכים, לעדכן, להוסיף ולמחוק נתונים על עותק הנתונים הנמצא בזיכרון המחשב שבו אנו עובדים. נדגיש כי מחיקת הנתונים אינה מוחקת אותם בפועל, אלא רק מסמנת אותם כמחוקים, כך שהם לא יתקבלו כתוצאה מביצוע שאילתת SELECT. הנתונים יימחקו בפועל רק כאשר מסד הנתונים יעודכן.

המידע נשמר בזיכרון בעצם מהטיפוס DataSet בתוך האוסף Tables אשר יכול להכיל כמה טבלאות. הגישה אל הטבלאות תיעשה על-ידי שם הטבלה או על-ידי אינדקס המציין את מיקומה באוסף הטבלאות (אינדקס הטבלה הראשונה הנו 0, אינדקס הטבלה השנייה הנו 1 וכך הלאה). לשם כך נגדיר עצם מהטיפוס DataTable ונשמור בו הפניה לטבלה:

```
DataTable dataTable = ds.Tables[0];
```

או

```
DataTable dataTable = ds.Tables["tblUsers"];
```

שלב 5 – הצגת הנתונים

כדי להציג את הנתונים שבטבלה, נתייחס לטבלה tblUsers. ניצור מחרוזת אשר תשרשר את כל הנתונים בזה אחר זה על-ידי הלולאה foreach. משפט foreach מאפשר ביצוע חזרות, בדומה למשפט for. השימוש במשפט foreach מתאים לבצוע חזרות על פני אוספים למשל מערכים, ובמקרים אלו הכתיבה שלו היא מקוצרת ולא דורשת (כמו במשפט for) להגדיר את הערך לקידום המונה ואת ערכו הסופי. למעשה משמעותו של משפט foreach הוא: בצע לכל איבר מטיפוס X באוסף ששמו Y.

נגדיר את המחרוזת ונאתחל אותה למחרוזת ריקה:

```
string resultStr = "";
```

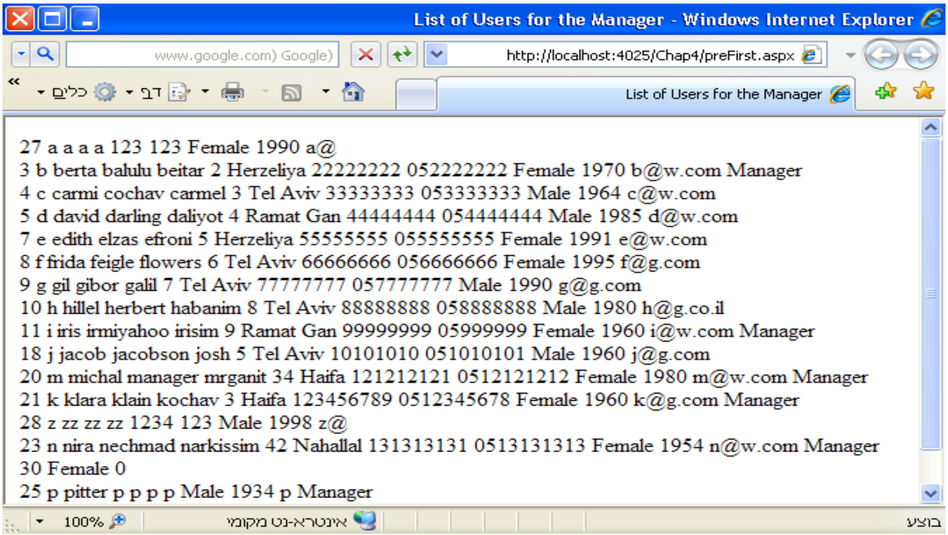
כדי לעבור על כל השורות בטבלת התוצאה, נגדיר משפט foreach שבו נשתמש בעצם בשם dr, שהוא מטיפוס המחלקה DataRow, כדי לעבור על כל השורות הטבלה ds.Tables[0], כלומר על אוסף השורות ds.Tables[0].Rows:

```
foreach (DataRow dr in ds.Tables[0].Rows)
{

  את כל ערכי השדות שנרצה להציג נשרשר למחרוזת אחת. //
  resultStr += dr["userId"];
  resultStr += dr["userName"];
  resultStr += dr["firstName"];
  resultStr += dr["lastName"];
  resultStr += dr["address"];
  resultStr += dr["phone"];
  resultStr += dr["cellPhone"];

  עלינו להציג גם את ערכו של השדה מְגָדֵר. נזכיר כי השדה הזה הוגדר כשדה מהטיפוס //
  bit, כלומר ערכו הוא בוליאני – הערך True ניתן למשתמש ממין נקבה. לפיכך, //
  נשרשר למחרוזת התצוגה "Female" או "Male" בהתאמה, על-פי ערך השדה. //

  if ((bool)dr["gender"])
    resultStr += "Female";
  else
```

```
    resultStr += "Male   ";
resultStr += dr["birthYear"];
resultStr += dr["email"];

 //      "Manager" גם ערכו של השדה isManager הוא בוליאני ועל כן נשרשר את המחרוזת
 //      עבור מנהל ורווחים – עבור משתמש שאינו מנהל.

if ((bool)dr["isManager"])
   resultStr += "Manager";
else
       resultStr += "    ";

          //      לבסוף, נשרשר תג ירידת שורה בכדי שהצגת הנתונים תהיה קריאה.
resultStr += "<br />";
}
```

את המחרוזת שיצרנו ניתן להציג כחלק מן המידע בדף שיישלח ללקוח.

נריץ את היישום הכולל בקשה להצגת המחרוזת resultStr ונקבל:



איור 4-5
הצגת טבלת משתמשים

נשים לב כי הצגת המידע במחרוזת מבלי שסידרנו אותו בטבלה או בכל אופן אחר מקשה את קריאות המידע. בהמשך נציג דרך להצגת המידע בצורה קריאה.

נציין כי כאשר ניגשנו לערכים שבשדות, ציינו את שם השדה (העמודה). עם זאת, כפי שניתן לגשת לטבלה הן על-פי שמה והן על-פי מיקומה באוסף הטבלאות, כך ניתן לגשת לשדות הן על-פי שמם והן על-פי מיקומם בשורה. בצורה דומה ניתן לגשת בשתי דרכים לרשומות (שורות) ולשדות (עמודות) שבטבלה. להלן כמה דוגמאות:

טבלה 4-2 דוגמאות לאופן הגישה לאיברים בטבלה

האיבר בטבלה שאליו ניגשים	אופן הגישה לאיבר הזה
השורה הרביעית (שמספרה הסידורי 3) בטבלה הראשונה (שמספרה הסידורי 0)	ds.Tables[0].Rows[3]
השורה הראשונה (שמספרה הסידורי 0) בטבלה השלישית (שמספרה הסידורי 2)	ds.Tables[2].Rows[0]
השורה הראשונה בטבלה ששמה tblUsers	ds.Tables["tblUsers"].Rows[0]
העמודה החמישית (שמספרה הסידורי 4) בשורה הרביעית (מספר 3) שבטבלה הראשונה (מספר 0).	ds.Tables[0].Rows[3][4]
עמודה ששמה userName בשורה השלישית שבטבלה הראשונה	ds.Tables[0].Rows[2]["userName"]
הערך הנמצא בעמודה userAddress בשורה 11 בטבלה ששמה tblUsers	ds.Tables["tblUsers"].Rows[10] ["userAddress"].ToString()

נשתמש בדף התבנית של הפעילות המסכמת בסוף פרק 3 – אתר המשחק 'מכונת המזל'. תבנית זו הגדירה דף המכיל טבלה בת שתי שורות. השורה הראשונה היא שורת לוגו והשורה השנייה מכילה שתי עמודות – האחת עמודת התפריט והשנייה עמודת התוכן. את טבלת המשתמשים נציג בעמודת התוכן של דף הניהול.

להלן התכנית המלאה, המציגה את המידע הקיים במסד הנתונים בצורה קריאה. בתכנית זו נשתמש בשרשור תגי HTML למחרוזת המייצגת את המידע ממסד הנתונים, וכן נוסיף שורת כותרת שתקל על הבנת סוג המידע המוצג.

הדף מתחיל בשלוש הנחיות המכילות הגדרה של שפת התכנות והקישור לאוסף המחלקות שבהן נשתמש לעיבוד מסד הנתונים SQL Server. שימו לב, כדי לשפר את הקריאות ואת המבניות של התרחיש בדף זה, אנו משתמשים בפונקציה ListOfUsers שמבצעת את המעבר על כל השורות של הטבלה שהתקבלה מביצוע שאילתת האחזור SELECT, משרשרת את נתוני הטבלה ותגי HTML ומחזירה מחרוזת הכוללת את נתוני הטבלה.

```
<!--   firstSqlExample.aspx -->
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Data.SqlClient" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
   DataSet ds = new DataSet();
   protected void Page_Load(object sender, EventArgs e)
   {
                                        //שלב ראשון: הגדרה של צורת ההתחברות למסד הנתונים
       string connectionString =
        @"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|
             \dbUsers.mdf; Integrated Security=True;User Instance=True";
             SqlConnection connection = new SqlConnection(connectionString);

                                        // שלב שני: טעינת הנתונים ממסד הנתונים לזיכרון
      SqlCommand cmd = new SqlCommand();
      cmd.CommandText = "SELECT * FROM tblUsers";
      cmd.Connection = connection;
      connection.Open();
      SqlDataAdapter adapter = new SqlDataAdapter(cmd);

      adapter.Fill(ds);
                                        // שלב שלישי: ההתנתקות ממסד הנתונים
      connection.Close();
   }
```

```
//פונקציה המבצעת את השאילתה: הצגת נתוני המשתמשים
public void ListOfUsers()
{
    //שלב רביעי: עיבוד הנתונים בזיכרון
    //הכנת מחרוזת המכילה את נתוני הטבלה שיש להציג

    string allUsersString = "<h1>רשימת המשתמשים </h1>";
    allUsersString += "<table>";
    // שורת הכותרת של הטבלה
    allUsersString += "<tr><td>Id</td><td>User Name</td><td>First Name</td>
        <td>Last Name</td><td>Address</td><td>Phone No.</td><td>Cell Phone</td>
        <td>Email</td><td>Gender</td><td>Birth Year</td><td>Is Manager</td> </tr>";

    // לולאת מעבר על כל שורות הטבלה
foreach (DataRow row in ds.Tables[0].Rows)
    {
        allUsersString += "<tr>";
        allUsersString += "<td>" + row["userID"] + "</td>";
        allUsersString += "<td >" + row["userName"] +  "</td>";
        allUsersString += "<td >" + row["firstName"] + "</td>";
        allUsersString += "<td >" + row["lastName"] + "</td>";
        allUsersString += "<td >" + row["address"] + "</td>";
        allUsersString += "<td >" + row["phone"] + "</td>";
        allUsersString += "<td >" + row["cellPhone"] + "</td>";
        allUsersString += "<td >" + row["email"] + "</td>";
        allUsersString += "<td >";
        if ((bool)row["gender"])
            allUsersString += "Female";
        else
            allUsersString += "Male";
        allUsersString += "</td>";
        allUsersString += "<td >" + row["birthYear"] + "</td>";
        if ((bool)row["isManager"])
            allUsersString += "Manager";
```

```
          else
               allUsersString += " ";                              //הצגת רווחים
          allUsersString += "</tr>";
       }
       allUsersString += "</table>";
       Response.Write(allUsersString);
    }
</script>
<%-- עיצוב דף התשובה --%>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>List of Users for the Manager</title>
   <link rel="Stylesheet" type="text/css" href="CSS/StyleSheet.css" />
</head>
<body>
<form id="form1" action="firstSqlExample.aspx" method="post" runat="server">
     <table;>
       <%--    שורת הלוגו המורכבת משתי עמודות  --%>
       <tr>
          <td colspan="2" dir="rtl">
             <h1>ברוכים הבאים לאתר מכונת המזל</h1>
          </td>
       </tr>
       <tr style="height:500px">
         <td style="width: 130px">
           <!-- עמודת התפריט, קישורים לדפי האתר השונים, על-פי הרשאות הגולש -->
          <div>
          <h2>תפריט</h2>
          <!-- כאן יופיע הקישורים על-פי ההרשאות השונות -->
        </div>
     </td>
          <%--   תוכן הדף   --%>
          <td  style="width: 88%; dir="rtl">
             <center>
```

```
<%-- ****************  הצגת פרטי המשתמשים  *************** --%>
<% ListOfUsers(); %>
</center>
</td>
</tr>
</table>
</form>
</body>
</html>
```

Is Manager	Birth Year	Gender	Email	Cell Phone	.Phone No	Address	Last Name	First Name	User Name	Id
	1990	Female	@a	123	123	a	a	a	a	27
Manager	1970	Female	b@w.com	052222222	22222222	beitar 2 Herzeliya	balulu	berta	b	3
	1964	Male	c@w.com	053333333	33333333	carmel 3 Tel Aviv	cochav	carmi	c	4
	1985	Male	d@w.com	054444444	44444444	daliyot 4 Ramat Gan	darling	david	d	5
	1991	Female	e@w.com	055555555	55555555	efroni 5 Herzeliya	elzas	edith	e	7
	1995	Female	f@g.com	056666666	66666666	flowers 6 Tel Aviv	feigle	frida	f	8
	1990	Male	g@g.com	057777777	77777777	galil 7 Tel Aviv	gibor	gil	g	9
	1980	Male	h@g.co.il	058888888	88888888	habanim 8 Tel Aviv	herbert	hillel	h	10
Manager	1960	Female	i@w.com	05999999	99999999	irisim 9 Ramat Gan	irmiyahoo	iris	i	11
	1960	Male	j@g.com	051010101	10101010	josh 5 Tel Aviv	jacobson	jacob	j	18
Manager	1980	Female	m@w.com	0512121212	121212121	mrganit 34 Haifa	manager	michal	m	20
Manager	1960	Female	k@g.com	0512345678	123456789	kochav 3 Haifa	klain	klara	k	21

איור 4-6

הצגה של טבלת המשתמשים בצורה קריאה

ב. אחזור חלק מהנתונים השמורים בזיכרון

כאשר מעוניינים לאחזר רק חלק מנתוני הטבלה, אפשר לבחור באחת משתי גישות: הגישה האחת – הפעלת שאילתת אחזור על מסד הנתונים ויצירת עצם חדש מהטיפוס DataSet, המכיל את הנתונים המבוקשים בלבד. הגישה השנייה – הפעלת שאילתת אחזור על ההעתק

של מסד הנתונים הנמצא בזיכרון, כלומר על ה-DataSet שבו נמצאת הטבלה כולה. את הטבלה מייצג עצם מהטיפוס DataTable ושורה בטבלה מיוצגת על-ידי עצם מהטיפוס DataRow[5].

כדי לאחזר נתונים ממסד הנתונים בזיכרון המיוצג על-ידי DataSet, עלינו לציין את שם הטבלה (עצם מהטיפוס DataTable) ולהשתמש בפעולה Select(string strWhere) (של המחלקה DataTable) אשר תקבל כפרמטר את הפסוקית WHERE של שאילתת SQL שאנו רוצים להפעיל על הטבלה. כאמור, הפסוקית WHERE מסננת רשומות, על-פי תנאי כלשהו מתוך טבלה יחידה. פעולה זו מחזירה מערך של עצמים מהטיפוס DataRow המייצגים את השורות בטבלה שענו לתנאי.

לדוגמה, אם נרצה לבחור את כל המשתמשים הנמצאים בטבלה tblUsers, אשר נולדו בשנת 1990, נגדיר מערך של DataRow אשר יקבל את תוצאת הפעולה Select על הטבלה tblUsers :

```
DataRow[] ar =  ds.Tables["tblUsers"].Select("birthYear = 1990");
```

תוצאת השאילתה SELECT מחזירה את כל הרשומות המקיימות את התנאי המועבר כארגומנט.

את שמות המשתמשים שנבחרו על-ידי הפעולה SELECT נציג באמצעות לולאת foreach הבאה :

```
foreach (DataRow dr in ar)
        Response.Write(dr["lastName"].ToString());
```

לולאת foreach מאפשרת לעבור על קבוצה של עצמים מאותו טיפוס. נשתמש בהוראה זו כדי לעבור על כל שורה (dr) בטבלה (ar).

---

[5] שתי מחלקות אלה הן חלק מהספריה **System.Data**

ניתן כמובן לסנן רשומות גם על-ידי תנאי מורכב, לדוגמה, רשימת כל המשתמשים (ממין נקבה) אשר נולדו בשנת 1990 :

```
DataTable myTable = ds.Tables["tblUsers"];
DataRow[ ] ar = myTable.Select("birthYear = 1990 AND gender = 1");
```

שאלה 4.5

א. שנו את התכנית כך שתציג את כל המשתמשים ששנת לידתם גדול מ- 1980.

ב. שנו את התכנית כך שתציג את השם הפרטי, שם המשפחה והכתובת של המשתמשים.

4.5.2 עדכון נתונים ממסד נתונים בשיטה הלא-מקושרת

בשיטה הלא-מקושרת, עדכון הנתונים מתבצע תחילה מול עותק הטבלה השמור בזיכרון (עצם מהטיפוס DataSet). נוסף על כך, כדי לשמור את השינויים שביצענו במסד הנתונים, עלינו לדאוג לעדכן את מסד הנתונים. נתאר תחילה כיצד ניתן לעדכן את העותק של הטבלה בזיכרון ולבצע את שלוש פעולות העדכון : שינוי שדות, הוספת רשומות וביטול רשומות. ולסיום נתאר כיצד מעדכנים את מסד הנתונים.

א. שינוי / עדכון הנתונים

עדכון של ערכים בטבלה יכול להיעשות ישירות על-ידי ציון השדה שרוצים לשנות וציון השינוי המבוקש. לדוגמה, עדכון הערך של שם משפחתו של המשתמש השלישי בטבלת המשתמשים ל-"Levi" ייעשה כך :

```
ds.Tables[0].Rows[2]["lastName"] = "Levi"
```

ואפשר גם כך :

```
ds.Tables[0].Rows[2][3] = "Levi"
```

ב. הוספת רשומה

הוספת רשומה חדשה לטבלה נעשית בשלשה שלבים :

1. יצירת עצם מטיפוס המחלקה DataRow המייצג שורה חדשה בטבלה. לשם כך נשתמש בפעולה NewRow של המחלקה DataTable המחזירה עצם מטיפוס DataRow.

```
DataRow dr = someTable.NewRow();
```

ובדוגמה שלנו נרשום:

```
DataRow dr = ds.Tables["tblUsers"].NewRow();
```

2. קביעת ערכים לכל אחד מהשדות של השורה החדשה:

```
dr["lastName"] = "xxxxxxx";
dr["firstName"] = "yyyy";
    . . .
```

3. הוספת השורה לטבלה

עתה משיצרנו את השורה החדשה, יש להוסיפה לאוסף השורות שבטבלה על-ידי השימוש בפעולה Add:

```
ds.Tables["tblUsers"].Rows.Add(dr);
```

כיצד נדע אילו שדות נמצאים בשורה?

נציב במאפיין command.Text שאילתה עם תנאי שלעולם לא יתקיים, למשל:

```
SqlDBCommand command = new SqlDBCommand ();
command.Text = "SELECT * FROM tblUsers WHERE (False);
```

מובן כי תוצאת שאילתה זו היא טבלה ריקה, שכן ביקשנו שורות המקיימות תנאי שקרי ואין שורות כאלה בכל טבלה שהיא. מדוע אם כן נרצה להריץ את השאילתה הזאת? בכדי לקבל את מבנה הרשומה, כלומר את מערך העמודות, שיספק לנו מידע על שמות השדות הדרושים לנו ברשומה החדשה.

למעשה, מערכת לניהול מסד הנתונים SQL Server, המקבלת שאילתה שפסוקית ה-WHERE שלה תמיד שקר (false), לא מריצה את השאילתה על כל הרשומות, אלא רק מחזירה את העמודות.

שימו לב: לא ניתן ליצור שורה חדשה בטבלה בלי להתייחס לטבלה עצמה. כלומר ניסיון לרשום את המשפט הבא יגרור שגיאה:

```
DataRow dr = new DataRow();                    // שגויי!
```

הוראה זו לא תעבור את שלב ההידור שכן יצירת שורה חדשה דורשת, בין היתר, להגדיר את השדות כך שיהיו זהים לשדות שבטבלה. לפיכך, רק הטבלה יכולה להקצות שורה.

ג. מחיקת רשומות מהטבלה

מחיקת רשומה מהטבלה נעשית על-ידי איתור הרשומה שנועדה למחיקה ועל-ידי הפעולה Delete הפועלת על הרשומה שנרצה למחוק. לדוגמה, כתוצאה מביצוע הפעולה:

```
ds.Tables[0].Rows[3].Delete();
```

תסומן כמחוקה הרשומה הרביעית (שמספרה הסידורי 3) שבטבלה הראשונה (שמספרה הסידורי 0) בעצם מהטיפוס DataSet.

אף-על-פי שהרשומה לא נמחקה ממש, אם נציג עתה את המידע הנמצא בטבלה הראשונה, לא תוצג הרשומה שזה עתה סימנו כמחוקה, לא מה-DataSet – ועל אחת כמה וכמה שלא ממסד הנתונים. המחיקה המוחלטת תיעשה רק בעת עדכון מסד הנתונים.

ניתן, כמובן, גם לאתר את הרשומה למחיקה תוך כדי ביצוע שאילתת אחזור. נחפש את כל הרשומות בטבלה tblUsers שבהן שמו הפרטי של המשתמש הוא יהויכין. התשובה לשאילתה הזאת יכולה להכיל כמה רשומות, על כן נשמור את תוצאת השאילתה במערך של שורות:

```
DataRow[] arRows = ds.Tables["tblUsers"].Select("firstName = 'יהויכין'");
```

עתה נסמן את השורה הראשונה במערך התוצאה כשורה מחוקה:

```
arRows[0].Delete();
```

זכרו כי השורה לא נמחקה בפועל, אלא רק סומנה כשורה מחוקה.

ד. עדכון מסד הנתונים

כאמור, העדכונים בוצעו בשלב הקודם בעצם מהטיפוס DataSet בלבד. כדי לעדכן את טבלת הנתונים (להוסיף, לשנות או למחוק נתונים) נבצע את הפעולות שלהלן:

1. נעדכן את נתוני הטבלה בזיכרון. כלומר, נעדכן את העצם מטיפוס DataSet המייצג נתוני טבלה.

2. נבנה פקודה לעדכון בעזרת עצם מטיפוס CommandBuilder. את פעולות עדכון מסד הנתונים בשיטה הלא-מקושרת (disconnected) מבצעת המחלקה CommandBuilder שמטרתה לבנות את שלוש שאילתות העדכון (הוספה / מחיקה / שינוי). עצם ממחלקה זו מקבל כפרמטר פקודת אחזור, מטיפוס המחלקה – SelectCommand המגדיר את השאילתה הנדרשת לביצוע.

שימו לב: שימוש במחלקה הבונה פקודה (CommandBuilder), אפשרי כל עוד העדכונים מתבצעים על טבלה אחת בלבד אשר הוגדר בעבורה מפתח ראשי.

כדי לבנות עצם מטיפוס CommandBuilder, עלינו להשתמש בעצם מהטיפוס DataAdapter ולאחר מכן לבנות את פקודת העדכון של מסד הנתונים המתאימה לצרכינו:

```
SqlDataAdapter adapter = new SqlDataAdapter();
SqlCommandBuilder builder = new SqlCommandBuilder(adapter);
```

כדי להוסיף שורה או שורות לטבלה נפעיל את בונה הפקודה להוספה:

```
adapter.InsertCommand = builder.GetInsertCommand();
```

כדי לשנות תוכן טבלה נפעיל את בונה הפקודה לשינוי:

```
adapter.UpdateCommand = builder.GetUpdateCommand();
```

וכדי לבטל שורה (או שורות) מטבלה נפעיל את בונה הפקודה למחיקה:

```
adapter.UpdateCommand = builder.GetDeleteCommand();
```

3. לסיום נבצע את העדכון במסד הנתונים בעזרת הפעולה Update של עצם מטיפוס SqlDataAdapter

```
adapter.Update(ds, "Users");
```

ה. התנתקות ממסד הנתונים

ההתנתקות ממסד הנתונים תיעשה על-ידי פעולת העדכון עצמה, כמו בניתוק באמצעות הפעולה Fill. זכרו, אם השתמשתם בפעולה ()Open להתחלת הקשר למסד הנתונים, עליכם להשתמש בפעולה Close כדי לסיים קשר זה.

דוגמה מסכמת לעדכון מסד נתונים בשיטה הלא-מקושרת

לסיכום, נציג יישום המציג דף שבו ניתן לבחור אחת מן האפשרויות הבאות: הצגת פרטי המשתמשים, עדכון פרטי המשתמשים וביטול רשומה. בדוגמה זו נשתמש באפשרות נוספת שקיימת בסביבת העבודה Visual Studio המאפשרת להפריד תוכן קובץ aspx לשני קבצים: כל תגי ה-HTML והתסריטים לעיצוב תגובת HTTP נשמרים בדף שהסיומת שלו aspx וכל הקוד בשפת #C המכיל הגדרות של משתנים ופונקציות נשמר בדף שהסיומת שלו aspx.cs. תוכלו להיעזר בנספח ג' שלעיל כדי ללמוד כיצד להשתמש באפשרות זו.

נבנה אתר חדש ובו נגדיר מסד נתונים הכולל את הטבלה tblUsers.

א. הצגה של הפרטים האישיים של כל המשתמשים

כדי לאחזר ולהציג את הפרטים האישיים של כל המשתמשים, נשתמש בשאילתת אחזור הזאת:

SELECT userID, userName, firstName, lastName, adderss, birthYear FROM tblUsers

פעולה זו נועדה כדי ליצור עותק של הטבלה בזיכרון בעצם מהטיפוס DataSet, כפי שהצגנו קודם לכן, ונשתמש בעותק זה כדי לשלוף ולהציג למשתמש את הנתונים. לשם כך נשתמש בכמה פונקציות:

- הפונקציה GetDataSet שמקבלת מחרוזת המייצגת שאילתת SQL ומחזירה עצם מטיפוס DataSet שמכיל טבלה שהיא תוצאה של השאילה שהופעלה.
- הפונקציה ListOfUsers שמשתמשת בפונקציה GetDataSet כדי ליצור עותק של טבלה בזיכרון שמכילה את הנתונים שאוחזרו מהשאילתה המבוקשת.
- הפונקציה PrintDataSet שמעצבת את הפלט המוצג.

שימו לב כי באירוע Page_Load אנו מגדירים את כל הפעולות הדרושות להתחברות למסד הנתונים.

נוסיף לאתר דף בשם Disconnected.aspx ובו נציג את טבלת המשתמשים :

```
<!-- Disconnected.aspx -->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Disconnected Page</title>
   <link rel="stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
   <form id="form1" runat="server">
       <div>
        <h1>Disconnected</h1>
        <h2>רשימת המשתמשים</h2>
             <% ListOfUsers(); %>
      </div>
   </form>
</body>
</html>
```

הפעולה ListOfUsers תיכתב בקוד שמאחורי הדף, כלומר בדף Disconnected.aspx.cs. למעשה, נכתוב בדף זה כמה פעולות, אשר יאפשרו גישה למסד הנתונים.

```
<!-- Disconnected.aspx.cs -->
using System.Data.SqlClient;              // קישור לספריה שמכילה מחלקות לטיפול במסד הנתונים
public partial class Disconnected : System.Web.UI.Page
{

                                          // הגדרת עצמים שיוכרו בכל הדף
public DataSet ds = new DataSet();
public SqlDataAdapter adapter = new SqlDataAdapter();
```

```
protected void Page_Load(object sender, EventArgs e) //לא הוגדרו פעולות לביצוע
{    }

        // פעולה המחזירה את מחרוזת ההתחברות למסד הנתונים //
        // בסביבת העבודה על המחרוזת להיות כתובה בשורה אחת. //
        // |DataDirectory| שימו לב לשימוש ב- //
        // המגדיר את מיקומו של מסד הנתונים ביחס למיקום הפרויקט //
        // ומאפשר להעביר את הפרויקט בין מחשבים מבלי לשנות את //
        // מחרוזת ההתחברות //
public string ConnStr()
{

  return @"Data Source=.\SQLEXPRESS;
        AttachDbFilename=|DataDirectory|\DataBase.mdf;
        Integrated Security=True;User Instance=True";
}

        // DataSet פעולה המקבלת שאילתה ומחזירה //
public DataSet GetDataSet(string strSql)
{
  ds = new DataSet();
        // שלב ראשון: הגדרת צורת ההתחברות למסד הנתונים //
  SqlConnection connection = new SqlConnection(ConnStr());
        // שלב שני: טעינת הנתונים ממסד הנתונים לזיכרון //
  adapter = new SqlDataAdapter(strSql, connection);
  connection.Open();
  adapter.Fill(ds);
        // שלב שלישי: התנתקות ממסד הנתונים //
  connection.Close();
  return ds;
}
        // DataSet פעולה המקבלת //
        // ומדפיסה את הנתונים שבו, //
        // בטבלה מתאימה לטבלת המשתמשים //
```

```
public void PrintDataSet(DataSet ds)
{
  string allStr = "<table>";
  allStr += "<tr><td>user ID</td><td>User Name</td><td>First Name</td>
<td>Last Name</td><td>Address</td><td>Birth Year</td></tr>";
  foreach (DataRow row in ds.Tables[0].Rows)
  {
       // לכל משתמש מודפסים ערכי השדות שלו
    allStr += "<tr>";
    allStr += "<td>"+row["userID"].ToString()+"</td>";
    allStr += "<td>"+row["userName"].ToString()+ "</td>";
    allStr += "<td>"+row["firstName"].ToString() + "</td>";
    allStr += "<td>" + row["lastName"].ToString() + "</td>";
    allStr += "<td>" +row["address"].ToString() +"</td>";
    allStr += "<td>"+row["birthYear"].ToString()+ "</td>";
     allStr += "</tr>";
  }
  allStr += "</table>";
  Response.Write(allStr);
}

       // פעולה המדפיסה בטבלה את כל המשתמשים השמורים ב–DataSet
public void ListOfUsers()
   {
       // הגדרת השאילתה
      string sqlAllUsers = "SELECT userID, userName, firstName, lastName, address,
                                        birthYear  FROM tblUsers";
       // יצירת DataSet
      ds = GetDataSet(sqlAllUsers);
       // הדפסת המשתמשים
      PrintDataSet(ds);
   }
}
```

ב. הצגת נתוני המשתמשים הגרים בכתובת מסוימת

באופן דומה, אם נרצה להציג רק חלק מן המשתמשים, למשל, את אלה הגרים בהוגוורט(Hogwart), נכתוב פעולה חדשה המגדירה שאילתה מתאימה, יוצרת בעבורה DataSet ומדפיסה אותו. נוסיף אם כן את הפעולה: LivesAt(string address), המקבלת מחרוזת המייצגת את הכתובת. הפונקציה LiveAt תשמש גם כן בפונקציה PrintDataSet להצגת הפלט.

```
public void LivesAt(string address)
 {
        string sql = string .Format("SELECT * FROM tblUsers  WHERE address = '{0}'",address);
        ds = GetDataSet(sql);
        PrintDataSet(ds);
 }
```

שימו לב, כי בפעולה זו השתמשנו בפעולה Format של string. פעולה זו מקבלת כמה פרמטרים. הראשון שבהם הוא מחרוזת המקצה מקום לערכים אשר יוצבו בתוך המחרוזת ושאר הפרמטרים הם הערכים שיש להציב שהוקצה. סדר הערכים קובע היכן יוצבו במחרוזת. הערך הראשון יוצב במקום {0}, השני במקום {1} וכו'. בצורה זו אין צורך לשרשר מחרוזות וערכים, אלא לשמור מקום לערכים על-ידי סימון מיקום: {i} ולהעביר את הערכים בסדר הנכון.

בחלק בו נעצב את תגובת ה- HTTP נכתוב תסריט שמזמן את הפונקציה LiveAT:

```
<body>
  <form id="form1" runat="server">
  <div>
    <h1>Disconnected</h1>
    <h2> רשימת המשתמשים</h2>
    <span>
      <% ListOfUsers(); %>
    </span>
```

```
    <h2>רשימת המשתמשים הגרים בהוגוורט</h2>
    <h3>שימוש ב–DataSet חדש, ייחודי למשימה זו</h3>
    <span>
      <% LivesAt("Hogwart"); %>
    </span>
  </div>
  </form>
</body>
```

שאלה 4.6

א. עדכנו את הדף בשרת (Disconnected.aspx, Disconnected.aspx.cs) כך שיבצע ויציג את כל המשתמשים שגרים ב-'Hogwart' ושנולדו לפני שנת 2000.

ב. עדכנו את הדף בשרת (Disconnected .aspx, Disconnected.aspx.cs) כך שיבצע ויציג את השמות הפרטיים ושמות המשפחה של כל המשתמשים שגרים ב- 'Hogwart'.

ג. הצגת נתוני המשתמשים לפי שנת הלידה

בדוגמאות הקודמות, בנינו DataSet חדש המתאים לכל שאילתה. בדוגמה זו נתאר כיצד ניתן לשלוף את הנתונים מתוך עותק של מסד הנתונים השמור בעצם מהטיפוס DataSet (הפעלת השאילתה SELECT * FROM myTable). כדי להציג רק חלק מן הנתונים, נבצע את הבחירה מתוך ה-DataSet בעזרת הפעולה Select של העצם מהטיפוס DataTable. ראשית נגדיר את מחרוזת השאילתה ולאחריה נקרא לפעולה Select על-ידי הטבלה ds.Tables[0]. פעולה זו תקבל את המחרוזת. בגישה זו אנו מנצלים את יתרונות הגישה הלא-מקושרת משום שאנו יכולים לאחזר נתונים לפי צרכינו מעותק של הטבלה הנמצא בזיכרון ומבלי שניאלץ לגשת שוב למסד הנתונים.

כדי לאחזר את כל המשתמשים שנולדו לפני שנת 2000, נוסיף פעולה המקבלת מספר המציין את שנת הלידה (בדוגמה שלנו, 2000). לשם כך, נוסיף פעולה בשם OlderThen. פעולה זו מזמנת את הפעולה GetDataSet, יוצרת עותק של הטבלה בזיכרון, ולאחר מכן

מסננת את הטבלה ויוצרת מערך של DataRow. המערך הזה יתמלא בשורות המתאימות לשאילתת SELECT אשר פסוקית התנאי שלה (הפסוקית WHERE) מועברת לשאילתה.

את הדפסת הנתונים שנבחרו נבצע בעזרת הפעולה PrintDataSet, אך הפעם נעביר מערך של DataRow ולא DataSet. על כן נוסיף גם פעולה זו לקוד.

את הפעולות נוסיף בדף הקוד Disconnected.aspx.cs :

```
public void OlderThen(int year) {
   string sql = "SELECT * FROM  tblUsers";
   ds = GetDataSet(sql);
   string where = string.Format("birthYear < {0}", year);
   DataRow[] ar = ds.Tables[0].Select(where);
   PrintDataSet(ar);
 }

public void PrintDataSet(DataRow[] ar)
{
            // DataSet זהה לפעולה שקיבלה
  foreach (DataRow row in ar)
  {
            // DataSet זהה לפעולה שקיבלה
  }
            // DataSet זהה לפעולה שקיבלה
}
```

ובדף התצוגה נרשום :

```
<body>
  <form id="form1" runat="server">
  <div>
    <h1>Disconnected</h1>
    <h2>רשימת המשתמשים </h2>
```

```
    <span>
      <% ListOfUsers(); %>
    </span>
    <h2>רשימת המשתמשים שנולדו לפני שנת 2000</h2>
    <h3>שימוש בבחירת שורות מתוך DataSet המחזיק עותק של כל הטבלה
    </h3>
      <% OlderThen(2000); %>
  </div>
  </form>
</body>
</html>
```

שאלה למחשבה

חשבו מה יקרה אם המשתמש יבקש פעולה נוספת שנועדה לאחזר את כל המשתמשים שנולדו לפני שנת 2000 ואת כל המשתמשים שגרים בכתובת מסוימת.

נזכור כי יישום שרת-לקוח הממומש בפרוטוקול HTTP הוא חסר מצב. כלומר, כל עוד הבקשות הללו הועברו כבקשת HTTP אחת, אנו יכולים לגשת שוב ושוב לעותק של מסד הנתונים שבזיכרון. אולם אם הבקשות הללו ממומשות כבקשות HTTP נפרדות, אין לנו גישה אל הנתונים שנשמרו ב-DataSet. כדי לפתור בעיה זו, ניתן לשמור את ה-DataSet של הטבלה בעצם מהטיפוס Session או מהטיפוס Application (בהתאם לצורך) ובכך להימנע מלגשת למסד הנתונים, אלא רק לשם עדכונו. לדוגמה, נוכל להשתמש בפעולה GetDataSet(sql) שיוצרת עצם מהטיפוס DataSet שאותו נשמור בעצם מהטיפוס Application:

```
Application["myTable"] = ds;
```

שאלה 4.7

שנו את התכנית כך שתאפשר למשתמש לבחור ולבצע בכל פעם אחת

מהשאילתות שפורטו בסעיפים א' עד ג'. הפעם, השתמשו בעצם מהטיפוס DataSet המכיל עותק של הטבלה השמור בעצם מהטיפוס Application ובצעו את השאילתות מתוך עותק זה.

ד. עדכון מסד הנתונים

כפי שהסברנו קודם לכן, כדי לעדכן את מסד הנתונים בשיטה הלא-מקושרת, נשתמש בפעולה Select לסינון הרשומות שיש לעדכן ובעזרת עצם מהטיפוס SqlCommandBuilder הבונה את שאילתות העדכון – הוספה / מחיקה / עדכון נבצע את העדכון בפועל. תחילה נדגים כיצד להוסיף נתוני משתמש חדש, ולאחר מכן – כיצד לעדכן נתונים ולבסוף כיצד לבטל רשומה.

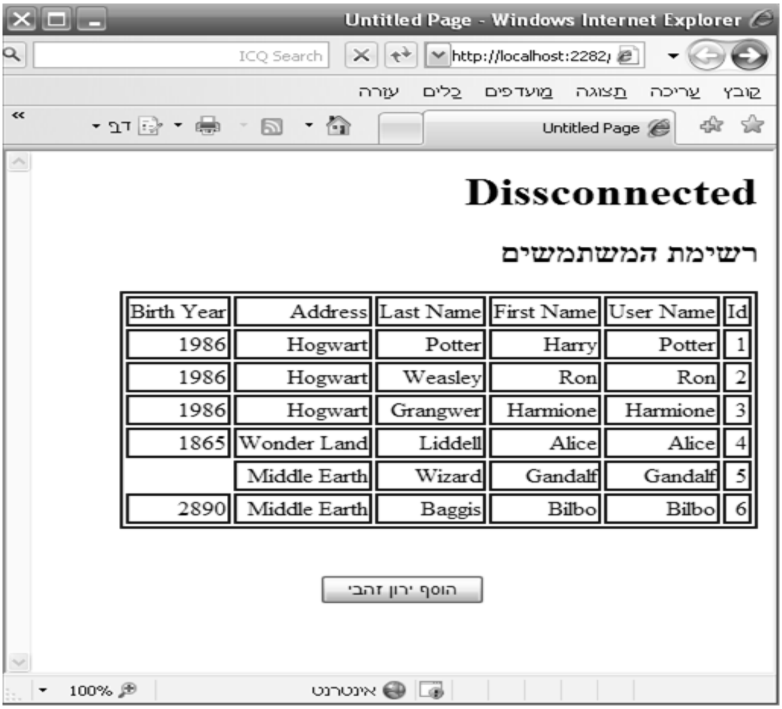
ד.1 הוספת נתוני משתמש חדש

כדי להוסיף את פרטיו של משתמש חדש בשם 'Yaron Zehavi', נוסיף לאחר הצגת טבלת המשתמשים כפתור להוספת משתמש:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
  <title>Disconnected Page</title>
    </head>
   <body>
    <form id="form1" runat="server">
       <div>
  <h1>Disconnected</h1>
  <h2> רשימת המשתמשים</h2>
  <span>
   <% ListOfUsers(); %>
  </span>
  <br /><br />
  <center>
    <input type="submit" id="addYaron" name="Send1"
               value=" הוסף ירון זהבי" runat="server" "/>
               <% if (Request.QueryString["Send1"]!= null){
                        Insert();
```

```
}
%>
  </center>
        </div>
    </form>
  </body>
</html>
```

נקבל את הדף הזה:



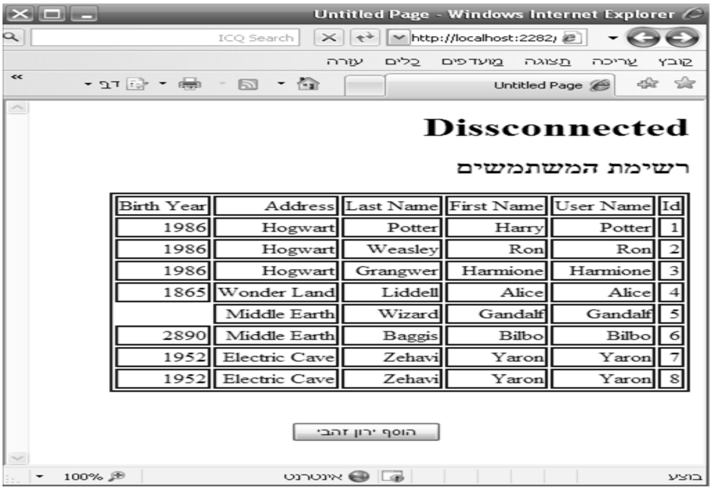
איור 7-4
אחזור נתוני משתמשים באמצעות השאילתה SELECT

כעת, נוסיף לקוד פעולה המוסיפה משתמש ששמו 'ירון זהבי'.

```
public void Insert()
{
        // קבלת עותק של הטבלה
        ds = GetDataSet("SELECT * FROM tblUsers");
```

```
// הגדרת שורה חדש לטבלה
DataRow dr = ds.Tables[0].NewRow();
// מתן ערכים לכל אחד מן השדות של השורה החדשה
 dr["userName"] = "Yaron";
 dr["firstName"] = "Yaron";
 dr["lastName"] = "Zehavi";
 dr["address"] = "Electric Cave";
 dr["birthYear"] = 1952;
dr["password"] = "yyy";
// הוספת השורה לטבלה
ds.Tables[0].Rows.Add(dr);
// עדכון מסד הנתונים
SqlCommandBuilder builder=new SqlCommandBuilder(adapter);
adapter.InsertCommand = builder.GetInsertCommand();
 adapter.Update(ds);
}
```

לחיצה על הכפתור תוסיף את המשתמש ירון זהבי למסד הנתונים. לחיצה נוספת תוסיף אותו שוב. הנה לפניכם החלון לאחר לחיצה כפולה על הכפתור 'הוסף ירון זהבי':



Birth Year	Address	Last Name	First Name	User Name	Id
1986	Hogwart	Potter	Harry	Potter	1
1986	Hogwart	Weasley	Ron	Ron	2
1986	Hogwart	Grangwer	Harmione	Harmione	3
1865	Wonder Land	Liddell	Alice	Alice	4
	Middle Earth	Wizard	Gandalf	Gandalf	5
2890	Middle Earth	Baggis	Bilbo	Bilbo	6
1952	Electric Cave	Zehavi	Yaron	Yaron	7
1952	Electric Cave	Zehavi	Yaron	Yaron	8

איור 4-8
טבלה לאחר שהוספה לה הרשומה של 'ירון זהבי'

שאלה למחשבה

כיצד ניתן למנוע מצב שבו נוסיף את אותו משתמש למסד הנתונים יותר מפעם אחת?

אם ננסה להוסיף משתמש עם ערך מפתח ראשי שקיים כבר בטבלה, נקבל הודעת שגיאה. כדי למנוע הוספת כפולה של משתמש, עלינו לבדוק תחילה באמצעות שאילתה מתאימה אם המשתמש אינו קיים בטבלה ורק אז להוסיפו כשורה חדשה.

שאלה 4.8

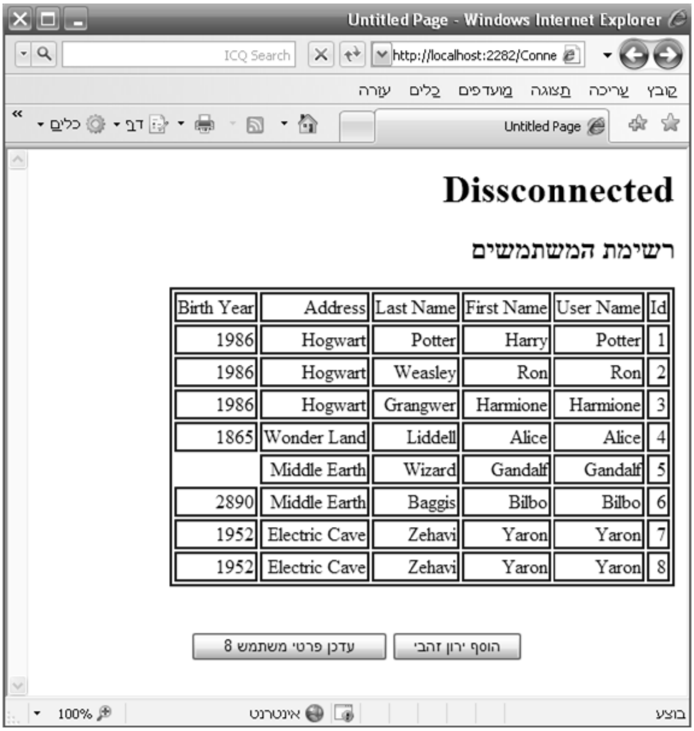
עדכנו את הפונקציה להוספת משתמש שתבדוק ושתוסיף את נתוני המשתמש רק אם אינו קיים עדיין בטבלה.

ד.2 עדכון נתוני משתמש

נשתמש בדוגמה שבה הוספנו פעמיים את הנתונים 'ירון זהבי' לטבלה, ונדגים כיצד ניתן לעדכן את נתוני הרשימה. בהתאם, נעדכן את המופע השני של 'ירון זהבי', ונשנה אותו ל- 'אלימלך זורקין'. נוסיף כפתור ליד הכפתור הקיים :

```
<body>
 <form id="form1" runat="server">
 <div>
   <h1>Disconnected</h1>
   <h2>רשימת המשתמשים</h2>
    <% ListOfUsers(); %>
   <br /><br />
   <center>
     <input type="submit" id="addYaron" name="addYaron"
            value="הוסף ירון זהבי" runat="server" />
    <input type="submit" id="editYaron8"   name="editYaron8"
             value="עדכן פרטי משתמש 8"  runat="server"   />
   </center>
 </div>
         <% if (Request.QueryString["addYaron "]!= null){
           Insert();
```

```
}
   if (Request.QueryString["editYaron8"]!= null){
Update();
}

%>
  </form>
</body>
```



איור 9-4

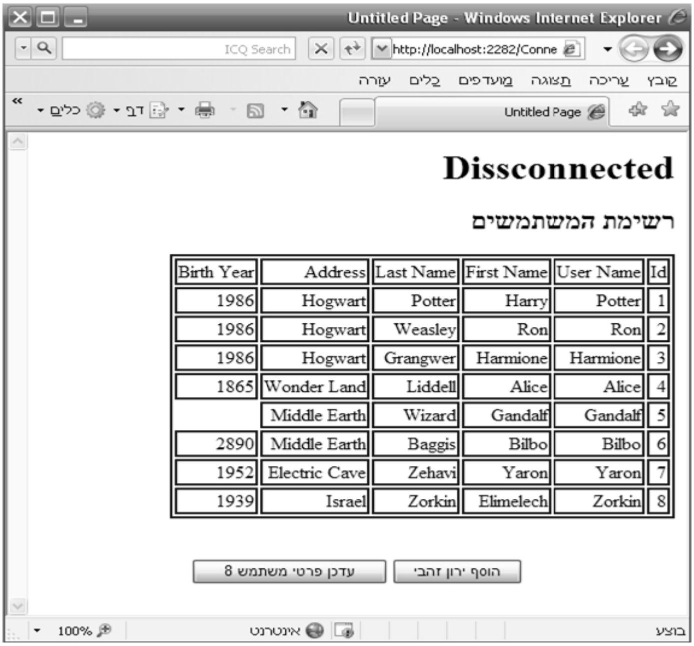
הוספת שורה ועדכון של נתוני משתמש מס' 8

כעת נוסיף פונקציית עדכון. שימו לב כי השורה שיש לעדכן נמצאת בטבלה בשורה מס' 7, ושלא כמו בפעולת ההכנסה, אין צורך להגדיר שורה חדשה, ומובן שיש לזמן את פעולת העדכון של ה-SqlCommandBuilder.

```
public void Update()
{
      // קבלת עותק של הטבלה
```

```
        ds = GetDataSet("SELECT * FROM tblUsers");
        // הגדרת השורה אותה רוצים לעדכן
        DataRow dr = ds.Tables[0].Rows[7];
        // מתן ערכים חדשים לחלק מהשדות של השורה
        dr["userName"] = "Zorkin";
        dr["firstName"] = "Elimelech";
        dr["lastName"] = "Zorkin";
        dr["address"] = "Israel";
         dr["birthYear"] = 1939;

        // עדכון מסד הנתונים
        SqlCommandBuilder builder=new SqlCommandBuilder(adapter);
        adapter.UpdateCommand = builder.GetUpdateCommand();
        adapter.Update(ds);
}
```

והנה התוצאה:



Birth Year	Address	Last Name	First Name	User Name	Id
1986	Hogwart	Potter	Harry	Potter	1
1986	Hogwart	Weasley	Ron	Ron	2
1986	Hogwart	Grangwer	Harmione	Harmione	3
1865	Wonder Land	Liddell	Alice	Alice	4
	Middle Earth	Wizard	Gandalf	Gandalf	5
2890	Middle Earth	Baggis	Bilbo	Bilbo	6
1952	Electric Cave	Zehavi	Yaron	Yaron	7
1939	Israel	Zorkin	Elimelech	Zorkin	8

איור 4-10
הוספת שורה ועדכון של נתוני משתמש מס' 8

ד.3 מחיקת נתוני משתמש

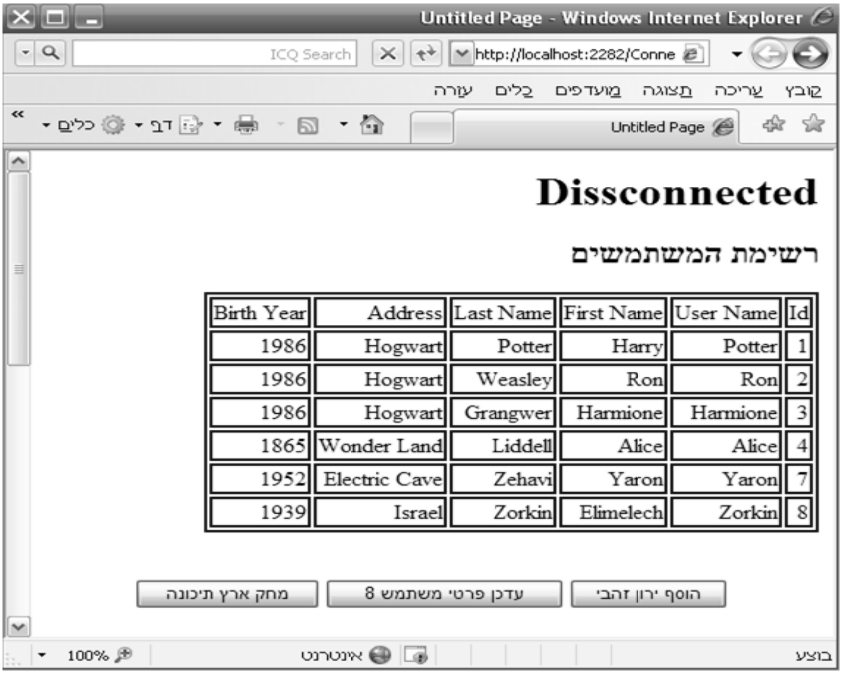
לסיום, כדי למחוק נתוני משתמש, נוסיף כפתור למחיקת המשמשים הגרים בארץ תיכונה:

```
<input type="submit" id="delete'Middle Earth'" name=" delete'Middle Earth "
    value="עדכן פרטי משתמש 8" runat="server" />
</center>
</div>
    <% if (Request.QueryString[" delete'Middle Earth "]!= null){
        Delete();
}
```

ונוסיף קוד של הפעולה למחיקה:

```
public void Delete()
{
    // קבלת עותק של הטבלה
    ds = GetDataSet("SELECT * FROM tblUsers");
    // הגדרת השורה אותה רוצים למחוק
    DataRow dr = ds.Tables[0].Rows[7]
    // או הגדרת השורות שאותן רוצים למחוק
    DataRow[] ar = ds.Tables[0].Select("address='Middle Earth'");
    // מעבר על השורות, שורה אחר שורה
    for (int i = 0; i < ar.Length; i++)
    {
    // מחיקה
    ar[i].Delete();
    }
    // עדכון מסד הנתונים
    SqlCommandBuilder builder=new SqlCommandBuilder(adapter);
    adapter.DeleteCommand = builder.GetDeleteCommand();
    adapter.Update(ds);
}
```

ולאחר לחיצה על הכפתור נקבל:



איור 4-11

טבלה לאחר ביצוע כל העדכונים

ואכן בילבו וגונדהלף נמחקו מרשימת המשתמשים.

4.6 תהליך האחזור ועיבוד הנתונים בשיטה המקושרת

עד כה עבדנו עם מסד הנתונים בשיטה הלא-מקושרת. בשיטה זו שלפנו מידע ממסד הנתונים על-ידי סדרת ההוראות אשר החזירה עצם מהטיפוס DataSet, שהוא עותק של הנתונים המתאימים הנמצאים במסד הנתונים. את עיבוד הנתונים ביצענו על העצם מהטיפוס DataSet ובסיום העבודה עדכנו את מסד הנתונים בשינויים שבוצעו.

לעתים, אנו מעוניינים בשליפת נתון אחד ממסד הנתונים, למשל מספר הזיהוי של משתמש, כמות המשתמשות ששמן מאוחסן בטבלה, מספר המשתמש בעל הניקוד הגבוה ביותר וכו'. במקרים אלה מיותר ליצור עצם מהטיפוס DataSet המכיל את הנתון היחיד.

העצם מהטיפוס SqlCommand מאפשר לנו לקבל את הנתון הזה על-ידי השימוש בפעולה ()ExecuteScalar. פעולה זו מקבלת כפרמטר מחרוזת המכילה שאילתת בחירה המחזירה ערך יחיד. הערך המוחזר הוא עצם מהטיפוס Object, ובכדי להשתמש בערכו יש לבצע המרה כנדרש.

צורת עבודה זו עם מסד הנתונים נקראת השיטה המקושרת (connected) שכן בשיטה זו אנו עובדים עם מסד הנתונים עצמו ולא עם ההעתק שלו. בשיטה זו לא נעבוד עם העצם DataAdapter אלא עם העצם Command.

להלן פעולה המקבלת מחרוזת שמייצגת שאילתת בחירה המחזירה עצם יחיד מתוך מסד הנתונים. הפעולה מזמנת את הפעולה המחזירה את מחרוזת ההתחברות:

```
public Object GetScalar(string sql)
{
      // הגדרת מחרוזות ההתחברות
    string connectionString = ConnStr();
      // יצירת עצם מהטיפוס SqlConnection
    SqlConnection connection = new SqlConnection(connectionString);
      // יצירת עצם מהטיפוס SqlCommand
    SqlCommand command = new SqlCommand(sql, connection);
      // התחברות אל מסד הנתונים
    connection.Open();
      // הפעלה של שאילתת הבחירה והחזרה של עצם התוצאה
    Object obj = command.ExecuteScalar();
      // התנתקות ממסד הנתונים
    connection.Close();
      // החזרת הערך
    return obj;
}
```

נשים לב כי פעולה זו מחזירה עצם ועלינו לבצע המרה לשם קבלת הערך.

פעולות אלה ייכתבו בקוד, כלומר בקובץ בעל סיומת aspx.cs.

גם את עדכונו של מסד הנתונים ניתן לבצע בשיטה המקושרת. לשם כך, נשתמש בפעולה ExecuteNonQuery של העצם SqlCommand. פעולה זו מבצעת שאילתת עדכון (INSERT / UPDATE / DELETE) על בסיס הנתונים, בהתאם לפעולה המופיעה במחרוזת השאילתה. אין צורך בכתיבת סדרת הוראות שונה בעבור כל אחד מן הסוגים של שאילתות העדכון.

להלן סדרת ההוראות לעדכון מסד הנתונים בשיטה המקושרת:

```
public int InsertUpdateDelete(string sql)
   {
       string connectionString = ConnStr();
               // משתנה מספר השורות שישתנו בעקבות הפעלת השאילתה
         int rowsAffected;
              // שלב ראשון: הגדרה של צורת ההתחברות למסד הנתונים
         SqlConnection connection = new SqlConnection(connectionString);
              // שלב שני: טעינת הנתונים ממסד הנתונים לזיכרון
         SqlCommand command = new SqlCommand(sql, connection);
              // התחברות אל מסד הנתונים
         connection.Open();
              // ביצוע שאילתת העדכון והחזרת מספר השורות שהושפעו
         rowsAffected = command.ExecuteNonQuery();
              // שלב שלישי ההתנתקות ממסד הנתונים
         connection.Close();
}
```

עתה, נניח כי יצרנו טופס רישום בדף Registration.aspx הכולל שדות טקסט בעבור שם המשתמש המבוקש, הסיסמה, שם המשפחה והשם הפרטי.

```
<form  id="regForm" method="post"
  action="Registration.aspx">
    <table>
        <tr>
          <th><input type="text" id="userName" />
          </th>
          <td>שם משתמש</td>
        </tr>
```

```
      <tr>
        <th><input type="password" id="pwd" /></th>
        <td>הקש סיסמה</td>
      </tr>
      <tr>
        <th><input type="text" id="firstName" />
        </th>
        <td>שם פרטי</td>
      </tr>
      <tr>
        <th><input type="text" id="lastName" />
        </th>
        <td>שם משפחה</td>
      </tr>
      <tr>
    <td colspan="2">
        <input type="reset"  value="נקה"/>
        <input type="submit"  value="שלח"/>
        </td>
      </tr>
   </table>
</form>
```

בעת טעינת דף Registration.aspx, עלינו לבדוק אם שם המשתמש תפוס. כיצד נעשה זאת? נוכל לעשות זאת באחת משתי דרכים. נגדיר ונמלא את העצם מהטיפוס DataSet בנתוניו של משתמש בעל שם המשתמש המבוקש. אם בעצם זה תהיינה שורות (למעשה, בדיוק שורה אחת) , אזי שם המשתמש תפוס.

בצורה דומה ניתן להחזיר רק את מספר הזהות (id) של המשתמש בעל שם המשתמש המבוקש. הרי מטרתנו היא רק לדעת אם השם הזה תפוס או לא, ואין לנו צורך במידע על המשתמש. לפיכך, נשתמש בפעולה ExecuteScalar ונבדוק אם הוחזר עצם או לא (הוחזר null).

בכל מקרה שבו שם המשתמש אינו תפוס, נוסיף למסד הנתונים את המשתמש החדש עם כל פרטיו.

נציג את שתי הדרכים.

להלן קוד הפעולה הבודקת אם שם המשתמש תפוס והמשתמשת במבנה ה-DataSet. הפעולה מקבלת את שם המשתמש.

```
public bool IsUserNameExists(string name)
{
    // הגדרת השאילתה
    string sql = "SELECT * FROM tblUsers WHERE (userName='"  "+ name +"  "')";

    // DataSet –יצירת ה
    DataSet ds = GetDataSet(sql);

    // 0–בדיקה אם מספר השורות שונה מ
    return (ds.Tables[0].Rows.Count != 0);
}
```

להלן קוד הפעולה הבודקת אם שם המשתמש תפוס והמשתמשת במבנה ExecuteScalar. הפעולה מקבלת את שם המשתמש.

```
public bool IsUserNameExists(string name)
{
    // הגדרת השאילתה
    string sql = "SELECT uId FROM tblUsers WHERE (userName='" + name + "')";

    // יצירת העצם
    Object obj = GetScalar(sql);

    // null בדיקה אם העצם לא
    return (obj != null);
}
```

מכאן ואילך אין חשיבות לצורת הבדיקה של שם המשתמש.

להלן קוד הפעולה InsertUser המקבלת כפרמטרים את שם המשתמש, את שמו הפרטי, את שם משפחתו ואת הסיסמה, ומוסיפה אותו למסד הנתונים:

```
  public int InsertUser(string userName, string password, string firstName, string lastName)
  {
     sql = string.Format("INSERT INTO tblUsers (uName,uPwd,uFirst,uLast) VALUES
                              ('{0}','{1}','{2}','{3}')",userName, password, firstName, lastName);
     return InsertUpdateDelete(sql);
}
```

להלן קוד האירוע Page_Load. אירוע זה ישמור את נתוני הטופס, יזמן את הפעולה הבודקת אם שם המשתמש תפוס, ואם לא – יזמן את הפעולה המוסיפה משתמש למסד הנתונים:

```
protected void Page_Load(object sender, EventArgs e)
{
      // שמירת הנתונים מתוך הטופס
      string userName = Request.Form["userName"];
      string password = Request.Form["password"];
      string firstName=Request.Form["firstName"];
      string lastName = Request.Form["lastName"];
      if (IsUserNameUsed(userName))
        Response.Redirect("RegistrationFail.aspx");
      else
        InsertUser(userName, password, firstName, lastName);
}
```

דוגמה מסכמת לעדכון מסד הנתונים בשיטה המקושרת

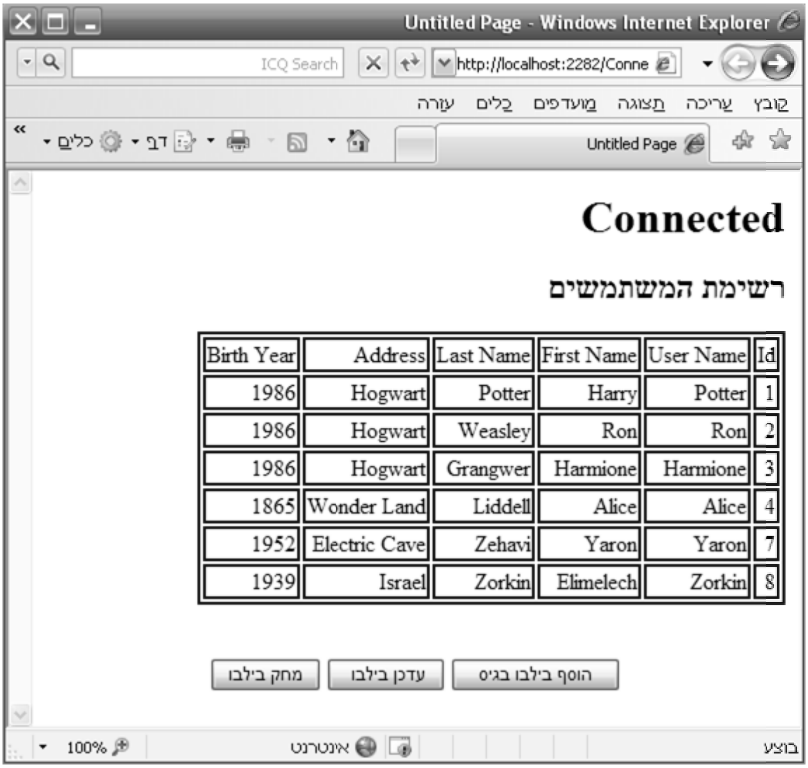
לאתר שבנינו בסעיף הקודם (דוגמה מסכמת בשיטה הלא-מקושרת) נוסיף דף עבור השיטה המקושרת למסד הנתונים. דף זה יציג את הטבלה כולה ושלושה כפתורים – להוספה, לעדכון ולמחיקה של המשתמש בילבו בגיס, שמחקנו כבר בדוגמה הקודמת.

לפניכם קוד העיצוב של הדף:

```
<!-- Connected.aspx-->
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Untitled Page</title>
   <link rel="Stylesheet" type="text/css" href="StyleSheet.css" />
</head>
<body>
   <form id="form1" runat="server">
      <div>
         <h1> Connected</h1>
         <h2> רשימת המשתמשים</h2>
         <span><% ListOfUsers(); %></span>
         <br /><br />
         <center>
            <input type="submit" id="addYaron" name="addYaron"
                value=" הוסף בגיס בילבו"  runat="server " />
            <input type="submit" id="edit"  name =" edit"
                value="עדכן בילבו  " runat="server" />
            <input type="submit" id="deleteMiddle" name="deleteMiddle"
                value=" מחק בילבו" runat="server"   />
         </center>
      </div>
<% if (Request.QueryString["addYaron "]!= null){
          Insert();
}
   if (Request.QueryString["edit"]!= null){
          Update();
}
   if (Request.QueryString["deleteMiddle "]!= null){
          Delete();
}
```

```
%>
   </form>
</body>
</html>
```

להלן הדף כפי שמוצג בדפדפן :



איור 4-12

עדכון טבלה בשיטה המקושרת

הקוד שמאחורי הדף (דף ה-Connected.aspx.cs) יכלול את הפעולות להגדרת מחרוזת ההתחברות, החזרת DataSet, הדפסת נתוני DataSet ובקשה להדפסת כל טבלת המשתמשים, שכבר ראינו בדף הלא מקושר. להלן חתימות הפעולות בלבד. הפעולות זהות לאלה שאתם כבר מכירים.

```
public string ConnStr() {…}
public DataSet GetDataSet(string strSql) {…}
public void ListOfUsers() {…}
public void PrintDataSet(DataSet ds) {…}
public int InsertUpdateDalete (string strSql) { …}
public Object GetScalar(string strSql) ) { …}
```

נוסיף לדף את הפעולה הבודקת אם שם המשתמש תפוס. הפעם נשתמש בפעולה המחזירה עצם יחיד ממסד הנתונים:

```
public bool IsUserNameExists(string name)
```

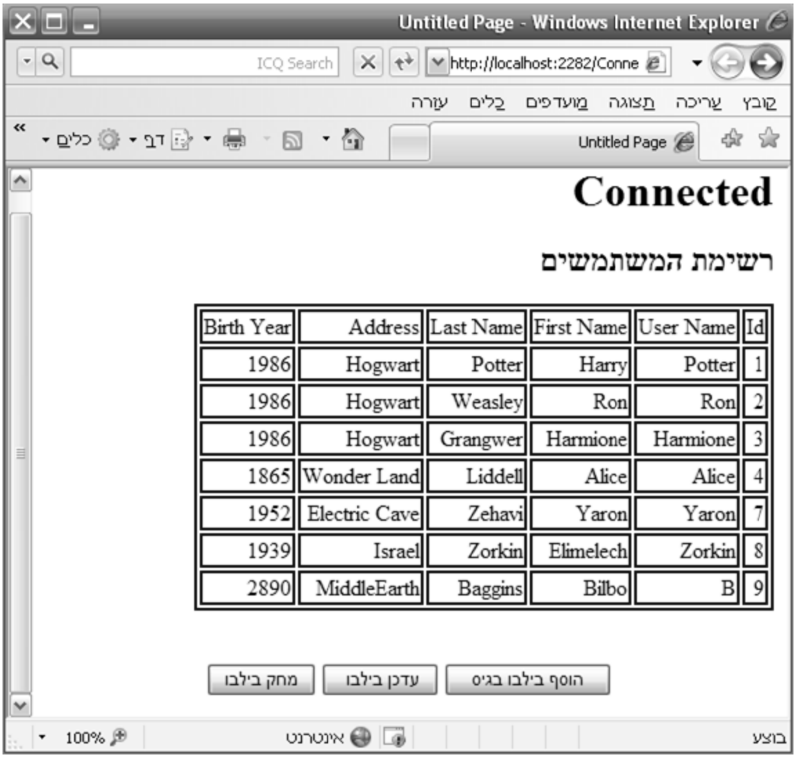
נשנה את פעולת ההכנסה כך שתקבל גם את כתובתו ושנת לידתו של המשתמש הנרשם.

```
public int Insert(string userName, string password, string firstName, string lastName,)
{
                                 // בדיקה אם קיים משתמש זהה
   if(IsExists(userName))
      return 0;                  // אם יש 0 שורות משתמש לא קיים
                                 // הגדרת השאילתה
   string sql = string.Format("INSERT INTO tblUsers
             (userName,password,firstName,  lastName, address,
             birthYear) VALUES ('{0}','{1}','{2}','{3}','{4}',{5})",
             userName, password, firstName, lastName, address, year);
                                 // זימון פעולה המעדכנת את מסד הנתונים
      return ExecuteNonQuery(sql);
   }
```

נזמן פעולה זו בעת לחיצה על כפתור 'הוסף בילבו בגיס':

```
protected void Insert()
{
            // הכנסת נתונים עבור בילבו בגיס
            // בדרך כלל הנתונים יתקבלו משדות טופס
   Insert("B","bbb","Bilbo","Baggis","Middle Earth",2890);
}
```

לחיצה על הכפתור תוסיף את בילבו בגיס לרשימת המשתמשים:



איור 4-13
תוצאות העדכון של הטבלה בשיטה המקושרת לאחר הוספת שורה

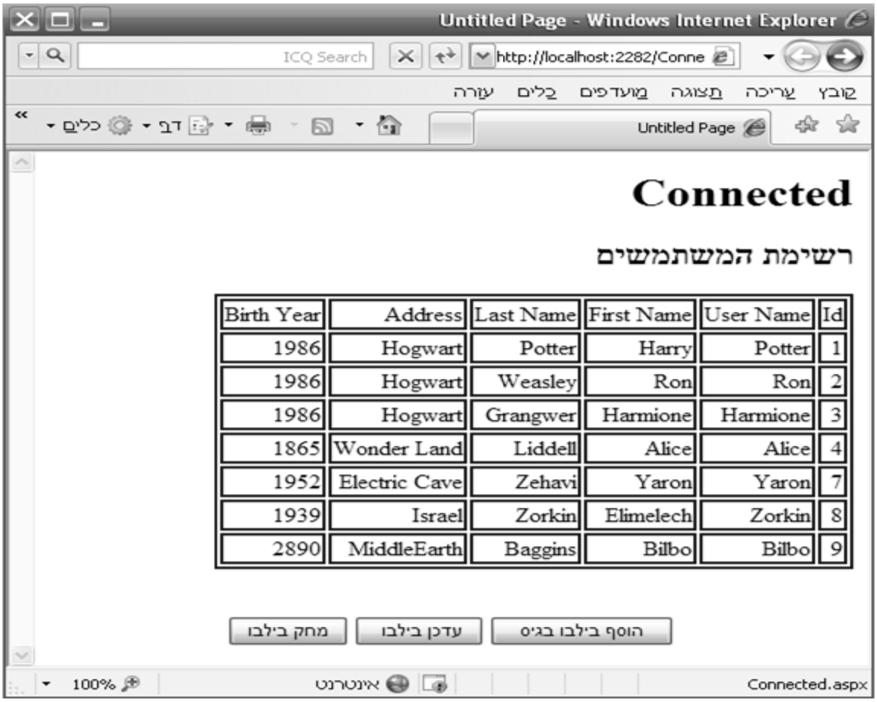
לחיצות נוספות על כפתור 'הוסף בילבו בגיס' לא יוסיפו שורות, שכן פעולת ההוספה בודקת אם המשתמש כבר קיים.

הוספנו את בילבו עם שם המשתמש 'B' ולא בשמו. נעדכן אפוא את שם המשתמש ל-Bilbo. לשם כך נוסיף פעולת עדכון. נשים לב רק כי עדכון שם משתמש יכול לגרום להופעת שם משתמש פעמיים, לפיכך, גם בפעולה זו נצטרך לבדוק ששם המשתמש אינו תפוס לפני ביצוע העדכון. פעולות עדכון של שדות שאינם בעלי ערך ייחודי אינן דורשות בדיקות מסוג זה.

```
public int Update()
{
```

```
        // נבדוק שאין משתמש בשם Bilbo
    if (IsExists("Bilbo"))
        return 0;
        // הגדרת השאילתה
    string sql = "UPDATE tblUsers SET userName='Bilbo' WHERE userName='B'";
        // קריאה לפעולת עדכון מסד הנתונים
    return ExecuteNonQuery(sql);
}
```

ולאחר לחיצה על הכפתור יציג הדפדפן את הדף הזה :



איור 4-14
תוצאות העדכון של הטבלה בשיטה המקושרת

עתה נותר עוד למחוק את בילבו. לשם כך נוסיף פעולה בשם Delete() אשר תחזיר את מספר השורות שהושפעו מפעולה זו.

```
public int Delete()
{
```

```
            // הגדרת השאילתה
    string sql = "DELETE From tblUsers WHERE userAddress='Middle Earth'";
            // קריאה לפעולת עדכון מסד הנתונים
    return ExecuteNonQuery(sql);
}
            // פעולה זו תזומן בעת לחיצה על הכפתור 'מחק בילבו'
protected void Delete()
{
            // הגדרת שאילתת המחיקה
    string sql = "DELETE From tblUsers WHERE userAddress='Middle Earth'";
            // קריאה לפעולת העדכון של מסד הנתונים
    ExecuteNonQuery(sql);
}
```

והדפדפן יציג שוב את הטבלה, ללא בילבו.

בשתי השיטות, המקושרת והלא-מקושרת, נוח לכתוב פעולות עזר המבצעות את החלקים המשותפים והמקבלות כפרמטרים את ערכי השדות הדרושים לבניית השאילתה. את הערכים נקבל מתוך טופס או מערכים השמורים ב-Session.

4.7 פעילות מסכמת – בניית אתר למשחק 'מכונת המזל'

נמשיך בבניית האתר לפרויקט שתיארנו בפרקים הקודמים.

שלב א

הוסיפו לפרויקט מסד נתונים אשר ישמור מידע על המשתמשים הרשומים באתר. בחרו את המידע שברצונכם לשמור על המשתמש, חוץ משם המשתמש, הסיסמה ומספר הנקודות שצבר במשחק מכונת המזל.

שלב ב

הוסיפו לפרויקט טופס רישום לאתר, אשר יבקש מהמשתמש את כל המידע שהחלטתם לשמור, פרט למספר הנקודות. בעת לחיצה על כפתור 'הירשם', שלחו את פרטי המשתמש לדף אשר יוסיף את המשתמש למסד הנתונים, על-פי הנתונים שסיפק בטופס, ובהנחה שמספר הנקודות של משתמש חדש הוא 1000.

בעת הרישום ודאו שלא קיים משתמש בשם זהה במסד הנתונים.

אם שם המשתמש קיים, שלחו שוב את המשתמש לטופס הרישום והודיעו על כך למשתמש. אם שם המשתמש אינו קיים, הוסיפו אותו למסד והפנו אותו לדף המשחק.

שלב ג

עדכנו את דף ההתחברות כך שהלחיצה על כפתור 'שלח' תגרום לבדיקת הנתונים הנמצאים במסד הנתונים. אם המשתמש קיים, יישמרו נתוניו הדרושים ב-Session, ואם המשתמש אינו קיים – יישלחו שוב נתוניו לדף ההתחברות בציון הודעה מתאימה.

שלב ד

עדכנו את האתר כך שמספר הנקודות שצבר משתמש ישמרו במסד הנתונים. שימו לב כי בעת התחברות יש לשמור ב-Session את מספר הנקודות שנרשמו למשתמש באתר, ובעת התנתקות, יש לעדכן את מסד הנתונים במספר הנקודות העדכני.

שלב ה

עדכנו את דף המנהל כך שיציג את רשימת המשתמשים בטבלה. לכל משתמש יוצגו נתוניו בשורה, ובעמודות האחרונות בשורה יופיעו קישורים לעדכון ולמחיקה של המשתמש.

שלב ו

הוסיפו לדף הניהול אפשרויות ניהול נוספות כגון:

חיפוש משתמשים – על-פי שם
על-פי טווח הניקוד
על-פי מין
על-פי עיר המגורים

סטטיסטיקה – הצגת כמות המשתמשים בעלי ניקוד חיובי וכמות המשתמשים בעלי ניקוד שלילי, הצגת מספר המשתמשים ומספר המשתמשות.

סקרים – תוצאות של סקר שביעות הרצון של המשתמשים באתר (לשם כך יש לבנות דף סקר המציג אפשרויות בחירה בכפתורי רדיו ומבקש מהמשתמשים לדרג את שביעות הרצון שלהם מן האתר). לכל

משתמש ניתן לשמור במסד הנתונים תכונה בוליאנית שערכה שקר כל עוד המשתמש לא ענה על הסקר, ואמת – מרגע שענה על הסקר.

משתמש יכול לענות על הסקר רק פעם אחת, כך שלאחר שענה עליו לא יוכל עוד לגשת לדף הסקר.

נספח ד':

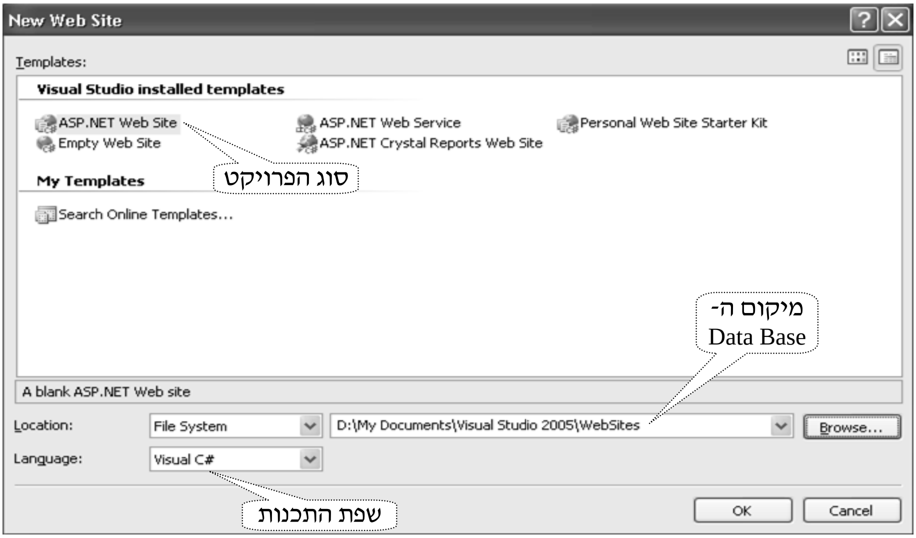
שימוש במסד הנתונים SQL Server

נספח זה כולל הוראות התקנה ושימוש בסביבה זו.

בניית מסד הנתונים

א. פתחו WebSite חדש מסוג ASP.NET Web Site :

שימו לב למיקום של התיקייה שנפתחה.

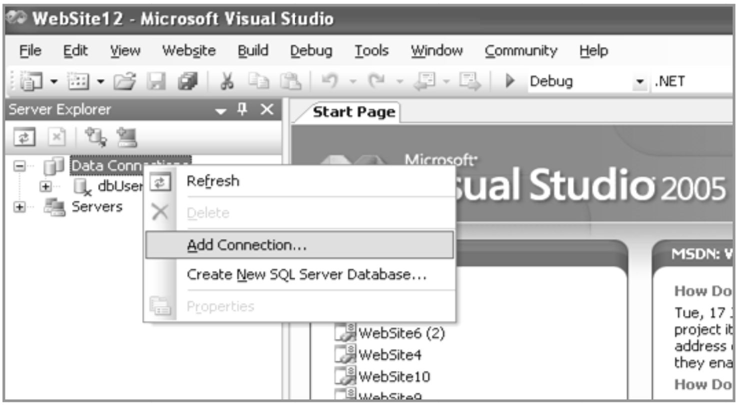


איור 4-15
בניית WebSite חדש

ב. יצירת מסד נתונים :

בחלון View ← Server Explorer[6], בעזרת הלחצן הימני בעכבר בחרו באפשרות Data Connection ולאחר מכן בחרו בפעולה Add Connection :

[6] שימו לב : במקצת הגרסאות של Visual Studio חלון זה נקרא database explorer.

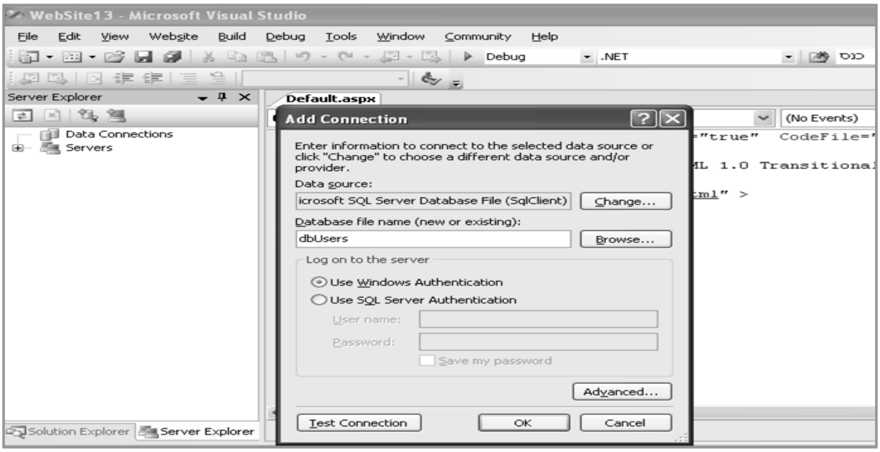


איור 4-16

קישור למסד הנתונים

ג. בחלון שנפתח רשמו את שמו של מסד הנתונים, ולסיום לחצו ok.

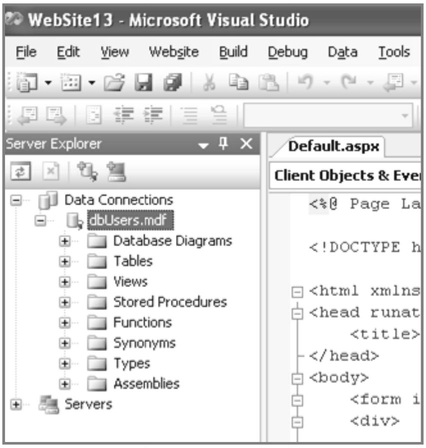
בדוגמה שלנו נרשום dbUsers (בכדי לזהות את סוג הטיפוס על-פי שמו, הוגדרה הרישא של שם בסיס נתונים כ-db והטבלה כ-tbl).



איור 4-17

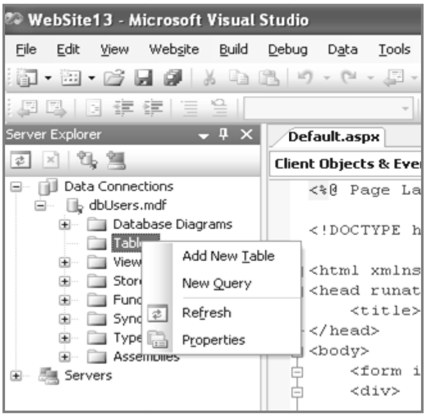
מתן שם למסד הנתונים

ד. בחלון ה-Server Explorer אפשר לראות את מסד הנתונים שהוספנו. הקלקה כפולה על השם של מסד הנתונים מציגה פירוט של כל התיקיות הקשורות למסד הנתונים.



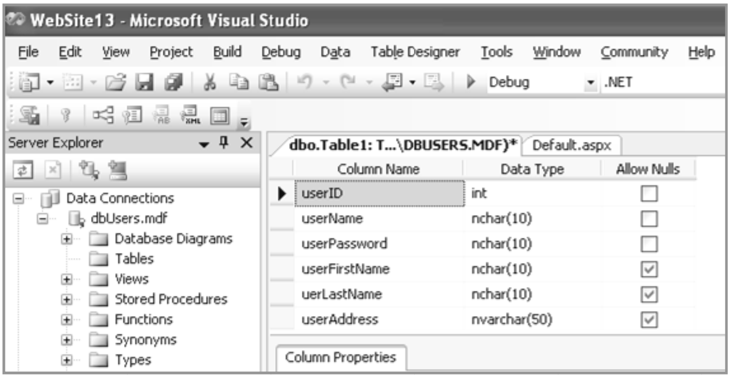
איור 4-18
פירוט התיקיות של מסד הנתונים dbUsers

ה. כדי להוסיף טבלה, לחצו עם הכפתור הימני של העכבר על התיקייה Tables ובחרו בפעולה Add New Table.



איור 4-19
הוספת טבלה חדשה למסד הנתונים

ו. בחלון שנפתח הגדירו את השדות ואת טיפוסי הנתונים המתאימים.



איור 4-20
הגדרת שדות הטבלה

להגדרת השדות יש להזין את הפרטים האלה: שם השדה, טיפוס הנתונים ואם שדה יכול להיות ללא ערך (שדה ריק).

SQL Sever מכיל טיפוסי נתונים שונים. כל טיפוס נתונים מגדיר תחום ערכים אפשרי ופעולות שאפשר לבצע על שדה מטיפוס זה. את טיפוסי הנתונים המוגדרים במסד נתונים SQL Server ניתן לחלק למספר קבוצות המוצגים בטבלה הבאה:

סוג נתונים		טיפוס	תיאור
מספרים	מספרים שלמים	int	ייצוג מספרים טבעיים
		Bit	ייצוג של 0 או 1 (מתאים לייצוג משתנה בוליאני)
	מספרים לא שלמים	float	ייצוג בשיטת נקודה צפה
		Real	ייצוג בשיטת נקודה צפה
		decimal	ייצוג עם דיוק קבוע (קביעת מספר ספרות אחרי הנקודה)
טקסט	תווים ומחרוזות	char	מחרוזת באורך קבוע
		varchar	מחרוזת באורך משתנה
		Text	משמש לייצוג טקסט ארוך
זמן		datetime	לייצוג תאריך ושעה

לבחירת טיפוס הנתונים לחצו על הלשונית בעמודה 'Data Type' ובחרו מתוך הרשימה הנתונה. השתמשו בטבלה הבאה כדי לקבוע את טיפוסי הנתונים של השדות השונים.
הכניסו את טבלת המשתמשים לפי ההגדרות שבטבלה שלפניכם:

שם השדה	**טיפוס הנתונים**	**המשמעות של ההגדרה**
userID	int	מספר שלם
username	nvarchar(10)	מחרוזת באורך משתנה של עד 10 תווים
userPassword	nvarchar(10)	מחרוזת באורך משתנה של עד 10 תווים
userFirstName	nvarchar(10)	מחרוזת באורך משתנה של עד 10 תווים
userLastName	nvarchar(10)	מחרוזת באורך משתנה של עד 10 תווים
userAddress	nvarchar(50)	מחרוזת משתנה של עד 50 תווים
userGender	bit	בוליאני: כן – עבור זכר ולא – עבור נקבה
userBirthYear	int	מספר שלם
userDegree	int	מספר שלם

לאחר הכנסת השדות, נקבע את השדה userID למפתח הראשי. מקמו את מצביע העכבר על השדה userID והקליקו על הכפתור הימני בעכבר, ברשימה שתיפתח בחרו באפשרות Set Primary Key. שדה המפתח מסומן בצלמית המפתח מימין.

שאלה למחשבה

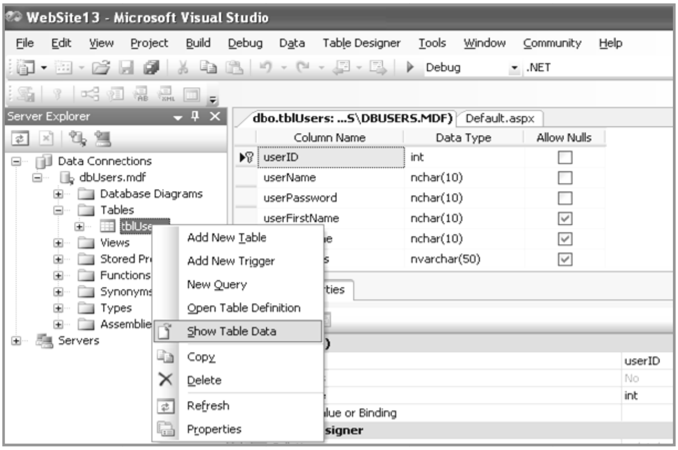
האם רצוי להוסיף שדה המציין את גיל המשתמש או לבחור בשדה זה במקום בשדה תאריך?

תשובה – ודאי שלא. גיל הוא ערך של משתנה. אם מעוניינים בשדה גיל, צריך להגדירו כשדה חישובי המשתנה על-פי התאריך הנוכחי.

ז. לאחר שתסיימו להגדיר את הטבלה, שמרו את הפרויקט ותנו לטבלה שם. רצוי לתת שם המתחיל ברישא tbl כך שנוכל בקלות לזהות את השם הזה עם טבלה במסד הנתונים.

בחרו בתפריט File → Save Table, ובחלונית שתיפתח רשמו את שם הטבלה tblUsers.

ח. לאחר שתסיימו להגדיר את הטבלה הכניסו לה ערכים. בחלון ה- Server Explorer לחצו על ה + שליד תיקיית Tables. מקמו את העכבר על שם הטבלה, הקליקו על הלחצן הימני בעכבר ובחרו בפעולה Show Table Data. נקבל את תוכן הטבלה ונוסיף רשומות כרצוננו.



איור 4-21
הכנסת ערכים לטבלה tblUsers

תרגיל

נסו להכניס לטבלה את הערכים האלה:

userID = 12
userName = ישראל
userBirthYear = 1.1.2008
userBirthYear = ישראל
userDegree = ישראל

מהן ההתרעות שאתם מקבלים אשר מונעות מכם להכניס ערכים מטיפוס שגוי?

זהו השלב שבו בולט אחד היתרונות של מסד נתונים לעומת טבלה במעבד תמלילים או בגיליון אלקטרוני. מסד הנתונים מונע הכנסת טיפוסים שגויים לטבלה וכן הכנסת שתי שורות בעלות ערך זהה בשדה המפתח.

הגדרת שאילתות

שאילתות במסד הנתונים ישולבו ביישום. עם זאת, בכדי לוודא שהשאילתות שאנו כותבים אמנם פועלות כנדרש, נריץ אותן תחילה בסביבת מסד הנתונים, ורק לאחר מכן נעתיקן למקומן ביישום.

נלמד כעת כיצד להוסיף שאילתות, להגדירן ולהריצן.
בחלון Server Explorer נמקם את העכבר על שם הטבלה ונקליק על הלחצן הימני בעכבר. נבחר בפעולה New Query. נתבקש לבחור טבלה ו/או טבלאות אשר מתוכן יילקחו הנתונים לשאילתה. לאחר בחירת הטבלאות, יש לסגור את החלון Add Table.

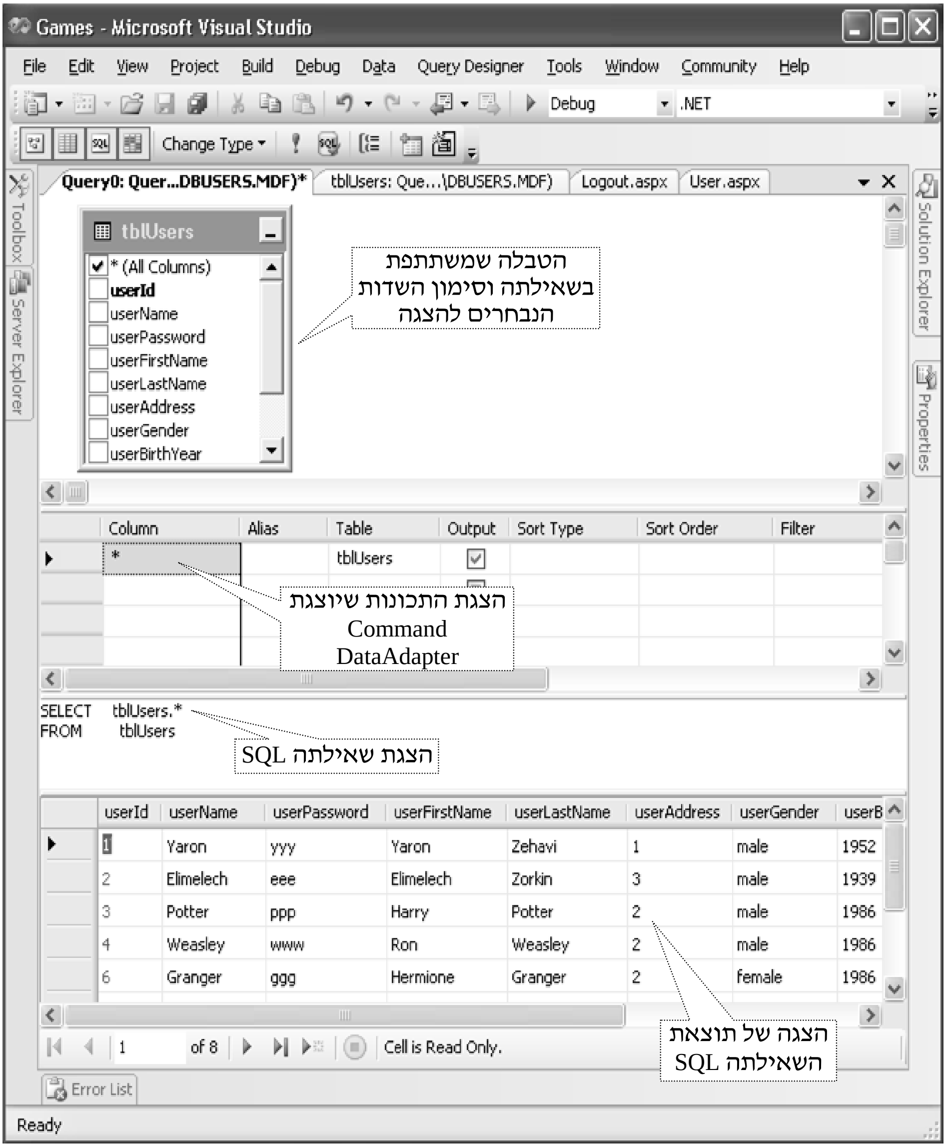
לפרויקט יתווסף קובץ נוסף להגדרת שאילתה. החלון שייפתח יורכב מכמה אזורים – אזור הטבלאות, אזור הגדרת התנאים על השדות, אזור השאילתה ב-SQL ואזור תוצאת השאילתה. בטבלה המוצגת יש לסמן את השדות המעורבים בשאילתה (אם מעוניינים בכל השדות, ניתן לסמן (All Columns)*).

נסמן √ במשבצת המיועדת לכל השדות ונקבל את השאילתה SELECT * FROM tblUsers. נריץ את השאילתה על-ידי לחיצה על הכפתור. תוצאת השאילתה תוצג בחלקו התחתון של החלון.

החלון באיור 4.22 התקבל כתוצאה מהרצת השאילתה SELECT * FROM tblUsers :

הרץ את כל השאילתות שהוגדרו בסעיף הקודם על בסיס הנתונים tblUsers.

בכדי להפעיל שאילתות המעדכנות את הנתונים במסד, יש לשנות את סוג השאילתה. נבחר בסוג השאילתה הרצוי מתוך האפשרויות שבכפתור Change Type.



איור 4-22
הגדרה והרצה של שאילתה

נתחיל בשאילתת ההוספה הבאה:

INSERT INTO tblUsers (userName, userPassword, userFirstName, userLastName, userAddress, userGender, userBirthYear, userDegree) VALUES ('Israel', 'myPass', 'Israel', 'Israeli', 'Herzelia', 'M', 1948, 3)

נבחר את השדות שאת ערכם נוסיף ואת הערכים עצמם ונריץ את השאילתה.

לאחר הרצת השאילתה נקבל הודעה המאשרת שהשאילתה שינתה רשומה אחת.

כיצד נבדוק שהרשומה אכן התווספה למסד הנתונים? נריץ שאילתת בחירה להצגת הנתונים כולם:

```
SELECT * FROM tblUsers
```

בצורה דומה נגדיר שאילתת עדכון. נבחר בסוג השאילתה Update, נסמן את השדות לעדכון בעמודה Set, נגדיר את הערכים החדשים בעמודה New Value ונוסיף תנאים לרשומות שאותן נרצה לעדכן בעמודה Filter.

```
UPDATE    tblUsers
SET userFirstName = 'Israela', userGender = 'female'
WHERE  (userName = 'Israel')
```

גם בדיקת השאילתה הזאת תיעשה על-ידי הרצת שאילתה להצגת הנתונים כולם.

עלינו עוד לבדוק שאילתת מחיקה. נמחק את הרשומה שצירפנו זה עתה.

```
DELETE FROM tblUsers WHERE username='Israela'
```

כבדיקה, נריץ שוב את השאילתה SELECT * FROM tblUsers ולא נראה עוד את הרשומה המתארת את המשתמש Israela.

יצירת שאילתות עם תנאים:

לדוגמה: כל המשתמשים שנולדו לאחר 1990.

עלינו לנסח את השאילתה הבאה:

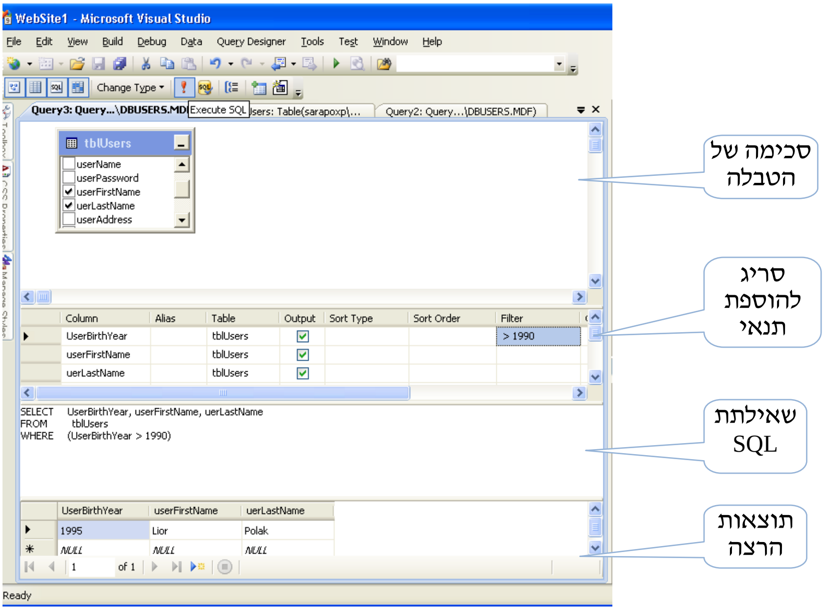
```
SELECT     userBirthYear, userFirstName, uerLastName
FROM        tblUsers
WHERE     (UserBirthYear > 1990)
```

צרו שאילתה חדשה והוסיפו את טבלת המשתמשים tblUsers לחלון השאילתות.

סמנו את השדות: userBirthYear, userFirstName, uerLastName.

כדי ליצור את התנאי userBirthYear > 1990, הוסיפו את התנאי '> 1990' במשבצת Filter.

לסיום הפעילו את השאילתה. להלן תוצאות ההרצה של שאילתה זו.



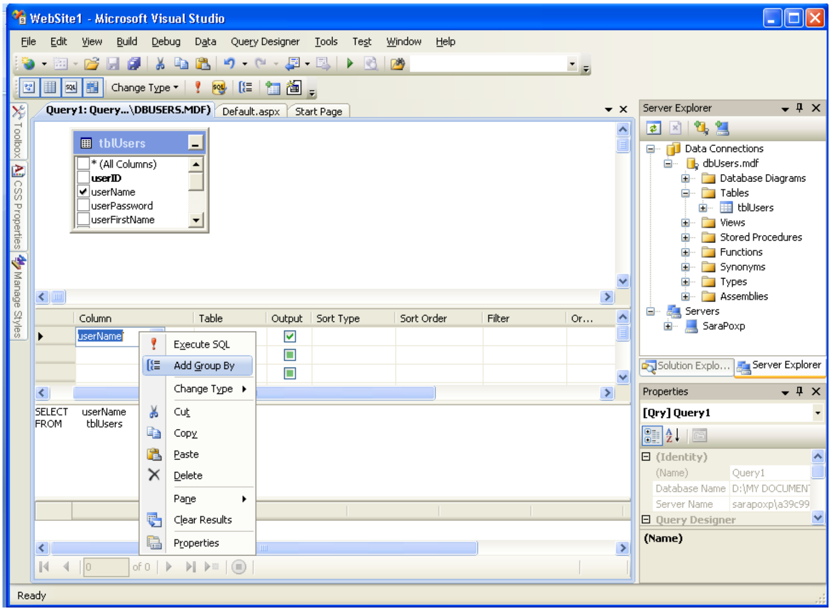
איור 4-23

הגדרה והרצת שאילתה בעזרת סרגל

בדרך דומה ניתן ליצור שאילתות עם תנאים מורכבים ושימוש בשאילתת הקיבוץ Group By.

לדוגמה, כדי ליצור שאילתה שמונה את מספר הרשומות, בנו שאילתה חדש ובחרו בטבלה tblUsers ובשדה username.

הקליקו על הלחצן ימני בעכבר על שם השדה, ובחלון שנפתח בחרו Add Group By.

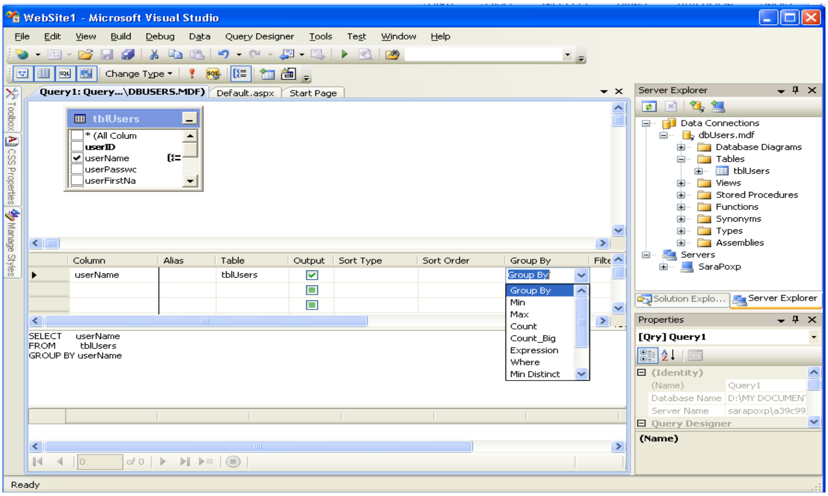


איור 4-24

הגדרה של שאילתת Group By

בחלון הסריג להוספת תנאים נוספה העמודה Group By.

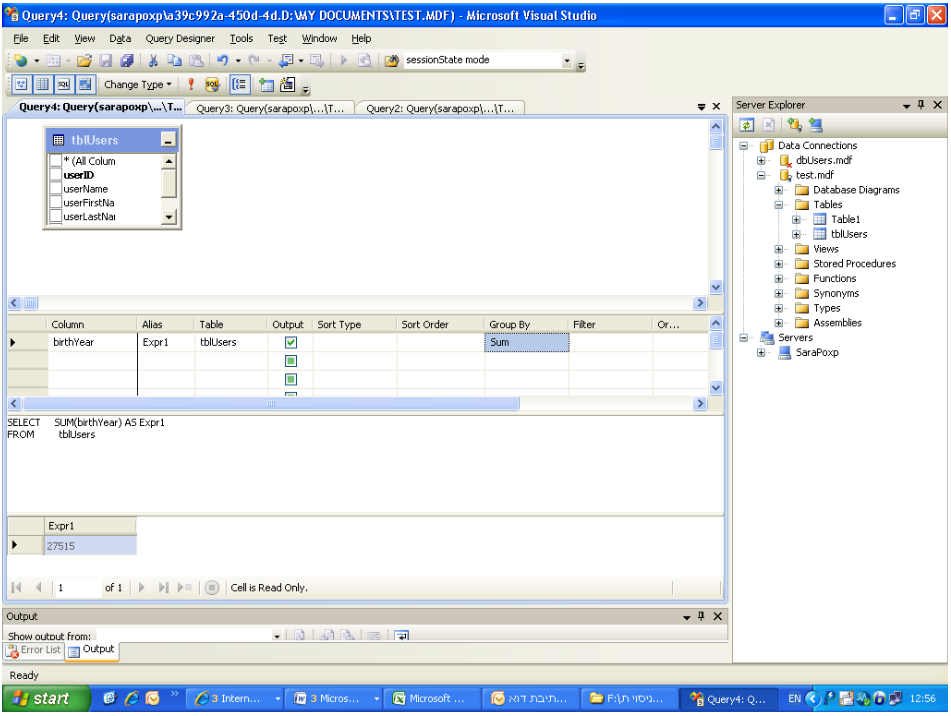
- לחצו על הלשונית ובחלון שנפתח בחרו count
- לסיום, הפעילו את השאילתה



איור 4-25

הגדרה של סוג שאילתת Group By

דוגמה נוספת:



איור 4-26

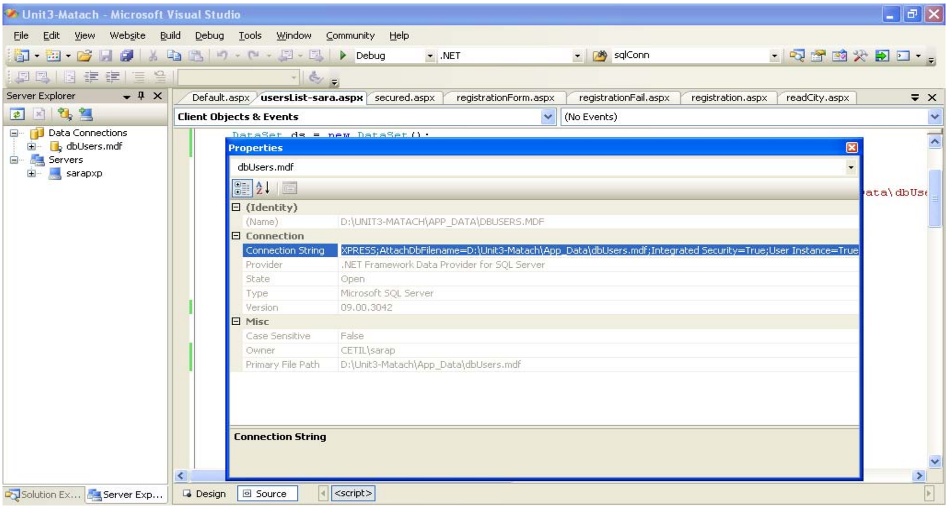
חלון שאילתות עם פונקציות המתייחסות לשדה נומרי וחלון group by שאילתת SQL מוצגת מלמטה

יצירת מחרוזת ההתחברות

בחלון Server Explorer נצביע עם העכבר על מסד הנתונים, ונסתכל על מאפייניו המפורטים בחלון Properties. אחד המאפיינים הנו מחרוזת ההתחברות.

נסמן את מחרוזת ההתחברות ונקבל:

Data Source=.\SQLEXPRESS;AttachDbFilename="D:\My Documents\Visual Studio 2005\WebSites\Introduction2Stateless\Games\App_Data\DBUsers.mdf";Integrated Security=True;User Instance=True



איור 4-27

יצירת מחרוזת ההתחברות

נקיף את המחרוזת כולה במרכאות (" "), נמחק את סימני המרכאות שבאמצע המחרוזת, ונוסיף בתחילתה את התו @ , המציין התעלמות מתווים מיוחדים. המחרוזת שנקבל היא :

@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\My Documents\Visual Studio 2005\WebSites\Introduction2Stateless\Games\App_Data\DBUsers.mdf;Integrated Security=True;User Instance=True"

נספח ד' – שימוש במסד הנתונים SQL Server

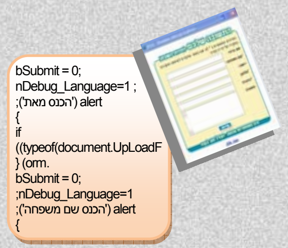
וזוהי מחרוזת ההתחברות.

אם מסד הנתונים נמצא בתיקיית הפרויקט, בתוך תיקיית App_Data, ניתן להחליף את המסלול אל מסד הנתונים בהגדרה DataDirectory, כלומר בהצהרה כי המסד נמצא בתיקיית המידע של הפרויקט.

במקרה כזה, העתקת הפרויקט למיקום אחר באותו מחשב או במחשב שונה אינה מצריכה עדכון של מחרוזת ההתחברות שכן המסלול אל מסד הנתונים הוא מסלול יחסי למיקום היישום.

במקרה כזה מחרוזת ההתחברות תיראה כך :

@"Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|\DBUsers.mdf;Integrated Security=True;User Instance=True"



פרק 5

תכנות של צד הלקוח

5.1 שילוב פעולות בצד לקוח

כפי שראינו בפרקים הקודמים, הלקוח יכול לשלוח לשרת בקשה אשר תכיל נתונים. השרת בודק את הנתונים שנשלחו, ומחזיר ללקוח תגובה בהתאם לתוצאותיה. השרת עורך תחילה בדיקות תקינות. למשל, בעת שליחה של טופס רישום לאתר, יש לוודא תחילה שכל הנתונים הדרושים מולאו ושהנתונים תקינים, לדוגמה:

- השם מורכב מאותיות בלבד.
- מספר הטלפון מורכב מספרות בלבד ותבניתו היא כשל מספר טלפון תקני.
- כתובת הדוא"ל מכילה את התווים '@' ו- '.' (נקודה) ובהם אות אחת לפחות וכדומה.

בחלק מהמקרים לאחר שנבדקה תקינותם של הנתונים, יש לבצע בדיקות נוספות. לדוגמה, יש לוודא ששֵם המשתמש המבוקש אינו קיים כבר, שכן לא ייתכנו באתר שני משתמשים בעלי שם משתמש זהה.

היכן נבצע בדיקות תקינות? את בדיקות התקינות נבצע בצד הלקוח. בדיקות תקינות בצד הלקוח מורידות מהעומס שעל השרת, שכן הנתונים המגיעים אליו תקינים ועל כן אין צורך בשליחת בקשה חוזרת לנתונים תקינים. כדי לאכוף כללי תקינות ואחידות ועל בדיקות שנעשות בצד לקוח, רצוי שהתסריטים לבדיקת תקינות יכתבו כדפי שרת. דפים אלה נשלחים אל הלקוח כאשר הוא מבקש את הדף והם מתבצעים בצד הלקוח. אולם, קיימות בדיקות תקינות שדורשות שימוש במאגר של נתונים ולכן מבצעים אותם בצד שרת. לדוגמה, את הבדיקה האם שם משתמש שהזין לקוח חדש כבר קיים במסד הנתונים, ניתן לבצע כמובן בצד השרת, שכן לשרת יש גישה לבסיס הנתונים.

בכדי לבדוק תקינות נתונים ובמידת הצורך לטפל בקליטה חוזרת של הנתונים, עלינו לשלב בדף ה- HTML, אותו מציגה תוכנת הלקוח, קטעי קוד לביצוע הכתובים בשפת תסריט (script language).

שפות התסריט בצד הלקוח נועדו במקורן בדיקת לבדוק את תקינותו של הקלט, אבל בכוחן לעשות פעולות רבות מעבר לכך. באתרים רבים אנו רואים תמונות המשנות את גודלן או אף תמונה המשתנה בעת הצבת הסמן באמצעות תפריט הנפתח בלחיצה על כפתור, ו/או בטקסט רץ אשר ניתן לעצור את ריצתו או לשנות את כיוון גלילתו. פעולות אלה ואחרות מתבצעות בשפת התסריט והן יוצרות אינטראקציה עם המשתמש שלא ניתן לבצעה באמצעות דפי HTML.

הקוד לביצוע שמשולב בדף HTML נכתב בשפת התסריט המוכרת על-ידי רוב הדפדפנים. תוכנת הלקוח (הדפדפן) יודעת לקרוא קובצי HTML, שהם קובצי טקסט, ולהריץ קטעי קוד (הנקראים תסריטים) שנכתבים בשפת התסריט. שפת התסריט דומה לשפת התכנות, אך היא פשוטה יותר והיא אינה כוללת את האפשרויות ששפת התכנות מאפשרת. שפת התסריט אינה עוברת את שלב ההידור, כי אם **פירוש** (**interpretation**). המפרש קורא את הטקסט הכתוב ומפרש אותו מיידית, כלומר מתרגם אותו כך שתוכנת הלקוח תוכל להציג את התוצאה.

בפרק הזה נציג את שפת התסריט JavaScript[1]. שפת JavaScript, וכמוה שפות תסריט אחרות, היא שפה מונחית עצמים ומונחית-אירועים. שפת JavaScript מכילה מספר מחלקות מוגדרות מראש, לדוגמה המחלקה Window (חלון) והמחלקה- Document (מסמך). מחלקות אלה מגדירות תכונות ופעולות שבהן ניתן להשתמש כחלק מהתסריט. בזמן שהדפדפן מפרש ומבצע את התסריט, נוצרים באופן אוטומטי מופעים של המחלקות השונות בהן התסריט משתמש.

שפת JavaScript גם מאפשרת להגיב לאירועים שהמשתמש יוצר. סביבה חלונאית, כגון הדפדפן, היא סביבה מונחית-אירועים. לחיצה על העכבר, פתיחת חלון, סגירת חלון, בחירת

---

[1] שפת Java ושפת JavaScript אינן קשורות זו לזו. שפת Java היא שפת תכנות שפיתחה חברת Sun, ואילו את שפת JavaScript פיתחה חברת Netscape כשפה שהדפדפנים יכולים להריץ.

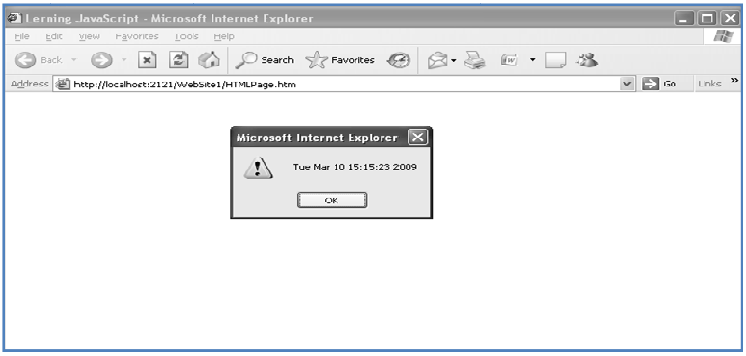
קטע טקסט, לחיצה על כפתור, הם כולם אירועים שהמשתמש גורם להם. גם דפי HTML מכילים לחצנים, תיבות טקסט, תיבות בחירה, כפתורי רדיו ועוד. משתמש הלוחץ על הכפתור send, ממלא טקסט או בוחר תיבות בחירה, גורם להתרחשות אירועים המשפיעים על תוכנת הלקוח, הדפדפן, אשר מצִדה מגיבה לאירועים אלה. נוסף על כך, קיימים גם אירועים המתרחשים על-ידי המערכת, כגון אירועי שעון, המתרחשים אחת לזמן מסוים (כפי שקבע מראש המתכנת). בסעיפים הבאים נכיר את העצמים הדרושים לניהול חלון ומסמך, נכיר אירועים שונים שבעזרתם ניתן לעצב באופן דינמי את העיצוב של הדף ונלמד לבדוק את תקינות הקלט לפני שליחתו לשרת.

דוגמה לדף HTML שמשולב בו תסריט

לפניכם דוגמה ראשונה לקובץ HTML המכיל תסריט בשפת JavaScript :

```
<!--Alert.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head >
    <title>Learning JavaScript</title>
    <script type="text/javascript">
        window.alert(Date());
    </script>
  </head>
  <body>
  </body>
</html>
```

הרצת הדף בדפדפן תגרום להופעת החלון הזה :



איור 5-1

חלון פלט המתקבל מהפעולה alert

הפעולה window.alert גורמה להופעת חלון הנושא את התאריך והשעה. הפעולה **alert** היא פעולה של המחלקה Window, מחלקה המובנה בשפת JavaScript. פעולה זו מטפלת בהצגת **פלט** ללקוח בחלון. הפלט יכול להיות מחרוזת או תוצאה של פעולת חישוב. בדוגמה שלנו הצגנו את התאריך והשעה שמחזיר עצם אחר, שמובנה ב- JavaScript, העצם Date. בניגוד לשפת C# או Java, בשפת JavaScript ניתן לרשום את הפעולה alert גם ללא ציון שם העצם window. כלומר, נוכל לרשום את התסריט גם כך:

```
<script type="text/javascript">
    alert(Date());
 </script>
```

קטעי קוד אלה ייכתבו בשפת התסריט JavaScript בין התגים <script>… </script>, אשר ימוקמו בין תגי HTML התחומים על-ידי head או ה- body של הדף:

```
<!--Template.htm-->
<html>
<head>
 <title>My Site</title>
 <script type="text/javascript">
                                        כאן ימוקמו התסריטים:
 </script>
</head>
<body>
 <title>My Site</title>
 <script type="text/javascript">
        אפשר למקם את התסריט גם כאן, אך דבר זה פחות מומלץ.
 </script>
</body>
</html>
```

בדרך ממקמים את התגים בקטע התחום על ידי התג HEAD. בקטע זה נרשום תסריטים שמטפלים באירועים ופונקציות (אליהם נתייחס בהמשך הפרק). תסריטים הממוקמים בקטע התחום על ידי התג BODY בדרך כלל מטפלים בתוכן של הדף וקשה יותר לשלוט על זמן ביצועם.

שאלה 5.1

א. מה יהיה הפלט של ההוראות הבאות. בדוק!:

1. alert("כל הכבוד!")
2. alert(7)
3. alert(3 + "*" + 6 + "= " + 3*6)
4. alert(3 + "+" + 6 + "= " + 3+6)

ב. מהו ההבדל בין הפלט המתקבל בסעיף 3 לבין הפלט המתקבל בסעיף 4?

5.2 התחביר של שפת JavaScript

בתרגיל 5.1 ראיתם כי ההוראה alert(3 + "+" + 6 + "= " + 3+6) מדפיסה את המחרוזת "3+6=36", ואילו ההוראה alert(3 + "*" + 6 + "= " + 3*6) מדפיסה 18, כלומר את תוצאת הפעולה החשבונית 3*6. חשוב להבין כי בשפת JavaScript, כמו בשפות אחרות, האופרטור + משמש הן לחיבור מספרים והן לשרשור מחרוזות. כאשר המפרש של JavaScript נתקל בפעולה חשבונית שאינה הפעולה '+' (חיבור), הוא ממיר את הערכים למספר, אם אפשר, ומבצע את הפעולה החשבונית. לעומת זאת, כאשר המפרש של JavaScript נתקל באופרטור '+' , הוא מתייחס אליו כאל שרשור מחרוזות. לפיכך, אם ברצוננו להציג את ערכו המספרי של הביטוי 3+6, עלינו להשתמש בפעולה המקבלת ביטוי חשבוני ומחזירה את תוצאתו. פעולה כזו קיימת ב-JavaScript ונקראת eval(). ההוראה alert(eval(3+6)) תציג אפוא את תוצאת החישוב של פעולת החיבור המתאימה.

השפה JavaScript מכילה פעולות נוספות הממירות למספרים ערכים מהטיפוס מחרוזת:

- הפעולה parseInt מקבלת מחרוזת ומחזירה אותה כמספר שלם. אם לא ניתן להמיר את המחרוזת למספר שלם, הפעולה מחזירה את הערך NaN המציין כי הנתון אינו מספר (NaN משמעו Not a Number)
- הפעולה parseFloat מקבלת מחרוזת ומחזירה אותה כמספר ממשי. אם לא ניתן להמיר את המחרוזת למספר ממשי, הפעולה מחזירה את הערך NaN.

לפני שנעמיק בשפת JavaScript, נציג בקצרה את כללי התחביר של השפה.

1. השפה רגישה לגודל האותיות: ההוראה ("שלום")**a**lert (האות הראשונה קטנה) היא חוקית, אך ההוראה ("שלום")**A**lert (שבה האות הראשונה גדולה), אינה חוקית.
2. הגדרת מחרוזת יכול להיעשות באמצעות מרכאות (״) או גרש (׳), אך יש להקפיד שהתיחום ייעשה באותו הסימן. המחרוזות ״שלום״, ׳שלום׳, ׳ביה״ס׳ ו- ״ביה״ס״ הן מחרוזות חוקיות. לעומת זאת, המחרוזות ״שלום׳ ו- ׳שלום״ , אינן תקינות.
3. שרשור טקסטים נעשה על-ידי הסימן ׳+׳.
4. הערות הנמשכות לאורך כמה שורות נרשמות בין התווים /*...*/, והערות הנמשכות עד לסוף השורה נרשמות לאחר התווים //.

הגדרת משתנים והוראות השמה

כמו כל שפת תכנות, גם JavaScript מאפשרת להגדיר **משתנים**. אבל שלא כמו בשפות תכנות אחרות, המשתנים של JavaScript אינם מטיפוס מוגדר. משתנה יכול להכיל כל ערך שהוא, בין שהוא ערך מספרי ובין שהוא ערך בוליאני, מחרוזת או ערך לא מוגדר; וטיפוס המשתנה נקבע לפי הערך שהושם בו.

כדי להצהיר על המשתנים בשפת JavaScript משתמשים במילה var (קיצור של variable) או מציבים ערך למשתנה:

```
var x, n;
x = 5;
```

אנו נקפיד להשתמש ב-var להצהרה על משתנים גם אם יינתן להם ערך בעת ההצהרה:

```
var  isValid = true;
```

שמות המשתנים חייבים להתחיל באות לועזית ויכולים לכלול אותיות לועזיות, מספרים, קו תחתי והתו $. רצוי ומומלץ לתת למשתנים שמות בעלי משמעות שייצגו את תפקידם. שם של משתנה אינו יכול להיות מילה שמורה בשפה, ויש להקפיד על הבחנה בין אותיות קטנות ובין אותיות גדולות. ב-JavaScript המשתנים name, Name, NAME, הם משתנים שונים.

השמה של ערכים למשתנה נעשית על-ידי מתן ערך, לדוגמה:

```
var  num1 = 3 , num2 = 5;
```

או באמצעות ביטוי חשבוני או לוגי, לדוגמה:

```
var sum = num1 + num2;
```

בעת הוראת ההשמה מחושב תחילה ערכו של האגף הימני בפעולת ההשמה, והערך המתקבל מושם במשתנה הרשום באגף השמאלי של הוראת ההשמה. לצורך עריכה של חישובים עם משתנים, ניתן להשתמש באופרטורים חשבוניים '+' (חיבור), '-' (חיסור), '*' (כפל), '/' (חילוק) ו- '%' (שארית), לדוגמה:

```
var n1 = 4, n2 = 5;
sub = n1 - n2;
n1 = n1 * 8;
```

ניתן לרשום פעולות חשבון גם בצורה מקוצרת, לדוגמה:

n1 += n2	// n1 = n1 + n2	הוספת ערכו של משתנה אחד למשתנה אחר
n1 –= n2	// n1 = n1 – n2	הפחתת ערכו של משתנה אחד ממשתנה אחר
n1 *= n2	// n1 = n1 * n2	הכפלת ערכו של משתנה אחד בערכו של משתנה אחר
n1 %= n2	// n1 = n1 % n2	חישוב השארית כתוצאה מחלוקתו של משתנה אחד במשתנה אחר

בשפת JavaScript, כמו בשפות רבות אחרות, נהוגים קיצורים נוספים, כגון שימוש באופרטור '++' להוספת 1 לערכו של משתנה ו '-- ' להפחתת 1 מערכו של משתנה.

קליטת נתונים

עתה, משהכרנו את המשתנים בשפה, ניתן ללמוד כיצד קולטים נתונים מהמשתמש ולשמרם במשתנים. קיימות כמה שיטות לקליטת נתונים מהמשתמש. השיטה העיקרית היא באמצעות טפסים ועליה נרחיב בהמשך. אפשר לקלוט נתונים מהמשתמש גם באמצעות הפעולה prompt, שהיא פעולה של העצם window. הפעולה prompt פותחת חלון שבו המשתמש צריך להקליד את הנתונים. הפעולה מחזירה מחרוזת שיש לשמור אותה במשתנה.

לפניכם דוגמה לדף HTML המכיל את תסריט JavaScript. התסריט כולל פעולות לקליטת שם וגיל של המשתמש ופעולה להצגת הודעה ובה ברכה ליום הולדתו.

```
<!--BirhtDay.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
```

```
    <title>Learning JavaScript</title>
    <script type="text/javascript">
        var name =window.prompt("הקלד את שמך");
        var age = window.prompt("הקלד את גילך", 16);
        window.alert(name+" , מזל טוב ליום הולדתך ה "+age);
    </script>
</head>
<body>
</html>
```

ניתן להציג בתיבת הטקסט שמציגה הפעולה window.prompt מחרוזת שתוצג כערך התחלתי. לדוגמה, בהוראה השנייה בה הצגנו א ההודעה היא "מהו גילך?", הוספנו את הערך 16 שיוצג בתיבת הטקסט. נציג את הדף BirthDay.htm בדפדפן ונקבל את סדרת חלונות.

החלון הראשון מציג את ההודעה 'שמך?' ותיבת טקסט ובתוכה 'הערך הראשוני':

איור 5-2
תיבת טקסט לקליטת שם ללא ערך תחילי

נקליד את הערך 'ישראלי':

איור 5-3
תיבת טקסט עם ערך שהקליד המשתמש

ונקבל חלון קלט חדש המבקש מן המשתמש לציין את גילו. הפעם תיבת הטקסט מכילה את הערך 16. נשאיר את הערך הראשוני, נלחץ על הכפתור OK ונקבל את הברכה הבאה ליום הולדתו:

איור 5-4
תיבת טקסט לקליטת גיל עם ערך התחלתי 16

שימו לב, רישום של שני הערכים בעברית הופך את סדר הופעתם בסביבת העבודה. השאירו את הסדר כפי שנרשם בסביבת העבודה. הערכים יוצגו בחלונות בצורה נכונה.

איור 5-5
הודעה ברכה למשתמש הכוללת את שמו וגילו

שאלה 5.2

א. כתוב תסריט המבקש מהמשתמש שני ערכים מספרים, מבלי לתת ערך ראשוני בעת הבקשה, ומציג כהודעה את תרגיל הכפל בין שני המספרים ואת תוצאתו. לדוגמה, עבור הערכים 2, 4 תוצג ההודעה 2*4=8

איור 5-6
הודעה המציגה תוצאת כפל בין שני מספרים

ב. כתוב תסריט המבקש מהמשתמש שני ערכים מספריים, ומציג אותם בשורות נפרדות. השתמשו במחרוזת "\n" עבור ירידת שורה.

עד כה הצגנו את המידע למשתמש בעזרת הודעה קופצת והשתמשנו בפעולה alert. אפשרות אחרת היא להציג את המידע בדף ה-HTML. לשם כך, נשתמש בפעולה document.write. הפעולה write היא פעולה של עצם מטיפוס Document והיא מקבלת כפרמטר מחרוזת או קוד HTML המוצגת למשתמש. לדוגמה, הקוד הבא יחולל דף HTML המכיל כותרת מרמה 1 בעלת הטקסט "האתר שלי" הרשום בצבע אדום ובגופן נרקיסים:

```
<script type ="text/javascript">
   document.write("<h1 style='color:Red; font-family:Narkisim' > האתר שלי</h1>");
</script>
```

שימו לב לשילוב בין המרכאות (") , המגדירות את המחרוזת המועברת כפרמטר, לבין הגרש (') , המגדיר את המחרוזת של המאפיין style.

<u>הערה</u>: בספר, לעיתים בגלל מגבלות של מקום, אנו פורסים מחרוזות על-פני מספר שורות, אך אין לעשות זאת ב-JavaScript. על המחרוזת להיכתב כולה באותה השורה.

ניתן כמובן לשרשר מחרוזות כפי שכבר עשינו בפעולה alert. לדוגמה, הקוד הבא יבקש מהמשתמש להקליד את שמו ויציג בדף כותרת ובה ברכת שלום למשתמש:

```
<script type ="text/javascript">
       var name =prompt("הקלד את שמך");
        document.write("<h1>שלום"+name+"</h1>");
</script>
```

שאלה 5.3

צרו מסמך HTML הקולט מהמשתמש את שמו, את הצבע האהוב עליו ושני תחומי התעניינות שלו (כגון סוגי ספורט או אמנות). המסמך יציג את שמו של המשתמש

בכותרת מרמה 2, הכתובה בצבע האהוב עליו, ומתחתיה רשימת תבליטים המציינת את תחומי ההתעניינות שלו.

5.3 מבני בקרה

כמו בכל שפת תכנות, פרט למשתנים ניתן להגדיר ב-JavaScript מבני בקרה, כלומר הוראות תנאי ולולאות.

ביצוע מותנה if... else

תפקידה של הוראה לביצוע-בתנאי- if היא להתנות ביצוע של קטע קוד בהתקיימותו של תנאי או תנאים מסוימים. אם ערכו הלוגי של התנאי הוא 'אמת' (true), אזי מתבצעת סדרת ההוראות שבאה אחרי התנאי. סדרת ההוראות לביצוע נכתבת בבלוק המוגדר בין סוגרים מסולסלים { }. אם ערכו הלוגי של התנאי הוא 'שקר' (false), אזי מתבצעת סדרת ההוראות שנכתבת בבלוק דומה לאחר המילה השמורה else. אם סדרת ההוראות היא של הוראה אחת בלבד, ניתן לוותר על כתיבת הסוגרים המסולסלים. ניתן גם לרשום הוראת if ללא else. במקרה כזה, אם ערכו של התנאי הוא שקר, לא תתבצע שום הוראה.

לפניכם דוגמה לתסריט המציג מגוון הודעות ברכה לימי הולדת, בהתאם לגיל החוגג:

```
<script type="text/javascript">
 var age = prompt("הקלד את גילך");
 if(age < 15)
 {
   document.write("כמה גדלת!");
 }
 else
 {
   document.write("יום הולדת שמח <br /> מזל טוב!");
 }
</script>
```

ההוראה if בודקת אם גיל המשתמש כפי שהוקלד קטן מ-15. אם כן, היא מדפיסה את ההודעה 'כמה גדלת!'; אם לא, היא מדפיסה הודעה בשתי שורות: בשורה הראשונה – 'מזל טוב!' ובשורה אחריה - 'יום הולדת שמח'.

התנאי הנבדק בהוראת התנאי הוא ביטוי לוגי. את הביטוי הלוגי יש לכתוב בתוך סוגריים עגולים ( ). ערכו של ביטוי לוגי יכול להיות 'אמת' (true) או 'שקר' (false) בלבד. ביטויים לוגיים מורכבים מפעולות ההשוואה הבאות: '==' (שווה), '=!' (שונה), '>' (קטן), '<' (גדול), '=>' (קטן או שווה), '=<' (גדול או שווה). כמו כן, ניתן להרכיב מספר ביטויים לוגים יחד בעזרת האופרטורים וגם ,'&&' (וגם), ' || ' (או), '!' (לא). ערכו של ביטוי לוגי המורכב ממספר ביטויים ובהם האופרטור '&&', הוא 'אמת', אך ורק כאשר ערכם של כל הביטויים המרכיבים אותו הוא 'אמת'. לעומת זאת, ערכו של ביטוי לוגי המורכב מכמה ביטויים וביניהם האופרטור '||' הוא 'אמת' כאשר ערכו לפחות אחד מן הביטויים המרכיבים אותו הוא 'אמת'. האופרטור '!' הופך את ערך הביטוי. ביטוי שערכו 'אמת' יהפוך ל 'שקר', ולהפך.

לפניכם דוגמה לקוד הקולט מן המשתמש שלושה מספרים המייצגים את המקדמים במשוואה ריבועית, ומייצר דף המציג את המשוואה ואת מספר פתרונותיה, על-פי ערכו של הדיסקרימיננט. הקפידו על כך שהמשוואה תציג גם מקדמים שערכם אפס ומקדמים שערכם שלילי יוצגו על-ידי – +. ניתן כמובן להוסיף קוד היוצר את מחרוזת המשוואה בהתאם לנתונים שהתקבלו.

```
<!—squareEqu.htm-->
<script type="text/javascript">
   var a =prompt("הקלד מספר");
   var b = prompt("הקלד מספר");
   var c =prompt("הקלד מספר");
   var tar = a+"x^2+"+b+"x+"+c+"=0";
   var dis = b*b-4*a*c;
    document.write("הריבועית למשוואה " + "<br />"+ tar+ "<br />");
    if(dis > 0)
    {
      document.write("יש שני פתרונות");
```

```
    }
    else
    {
      if(dis == 0)
      {
        document.write("יש פתרון יחיד");
      }
      else
      {
        document.write("אין פתרונות");
      }
  }
</script>
```

שאלה 5.4

צרו מסמך HTML הקולט מהמשתמש שני מספרים. אם שני המספרים זוגיים, יוצג סכומם – אחרת תוצג מכפלתם.

שאלה 5.5

צרו מסמך HTML הקולט מהמשתמש שני מספרים, ומציג את שני המספרים והודעה המציגה את המספר הגדול מביניהם.

שאלה 5.6

צרו מסמך HTML הקולט מהמשתמש שני מספרים ופעולה חשבונית (+, -, *,/) ומציג את הביטוי החשבוני שמתקבל משני המספרים והפעולה וכן את תוצאת הביטוי.

שאלה 5.7

צרו מסמך HTML הקולט מהמשתמש את שמו ומינו (זכר / נקבה) והמציג בכותרת מרמה אחת את שמו של המשתמש. אם המשתמש הוא ממין נקבה, יוצג שמה בצבע אדום, אחרת – יוצג השם בצבע ירוק.

שאלה 5.8

צרו מסמך HTML הקולט מהמשתמש את הזמן המינימלי לריצת 100 מ'

הנדרש כדי להתקבל לנבחרת הריצה של בית-הספר ואת זמן הריצה של המשתמש, והמציג הודעה אם המשתמש יכול להשתתף בנבחרת הריצה או לא.

שאלה 5.9

צרו מסמך HTML הקולט מהמשתמש שנה מסוימת; אם השנה מעוברת, הוא מציג בחלון (alert) הודעה מתאימה. שנה מעוברת היא שנה שמספרה מתחלק ב-4 אך לא ב-100 וכל השנים שמתחלקות ב- 400.

שאלה 5.10

צרו מסמך HTML הקולט מהמשתמש את מספר החודש בלוח השנה הלועזי והמציג את מספר הימים בחודש הזה.

שאלה 5.11

צרו מסמך HTML הקולט מהמשתמש את מספר היום בשבוע (בין 1 ל-7), והמציג את שמו של היום (ראשון – שבת).

הוראת בחירה – Switch

פתרונה של שאלה 5.11 הצריכה תנאים רבים, שכן ערכו של היום בשבוע יכול להיות אחד מבין שבעה ערכים אפשריים. השפה JavaScript מכילה הוראת בחירה switch בדיוק למקרים כאלה. הוראה זו בוררת בין מספר ערכים אפשריים של משתנה מבצעת פעולות ומציגה הודעות שונות לכל אפשרות. נדגים את השימוש בהוראת הבחירה switch בקוד הפותר את שאלה 5.11 באמצעות הוראה זו.

```
<!—SwitchDay.htm-->
<script type="text/javascript">
   var day =prompt("הקלד מספר יום בשבוע");
   switch (parseInt(day))
   {
      case 1:
         document.write("ראשון");
         break;
      case 2:
         document.write("שני");
         break;
```

```
      case 3:
        document.write("שלישי");
        break;
      case 4:
        document.write("רביעי");
        break;
      case 5:
        document.write("חמישי");
        break;
      case 6:
        document.write("שישי");
        break;
      case 7:
        document.write("שבת");
        break;
      default:
        document.write("אין יום כזה");
    }
</script>
```

נסביר כמה נקודות בתסריט הזה:

1. הפעולה prompt מחזירה מחרוזת, ובכדי להתייחס לערך החוזר ממנה כמספר עלינו להמירו למספר. לשם כך, נשתמש בפעולה parseInt. אפשר היה שלא להמיר את המחרוזת למספר, אך אז צריך היה לסמן את הערכים בגרשיים (case "1":).
2. יש לפרט כל ערך אפשרי בהוראת case נפרדת.
3. רצוי, אך לא חובה, להוסיף אפשרות אחרונה default המייצגת את כל האפשרויות שלא פורטו.
4. ההוראה break מסיימת את ביצוע ההוראות. ללא הוראה זו יתבצעו ההוראות הבאות גם אם הן שייכות להוראת case אחרת.

כיצד ניתן להשתמש בהוראה break? לפניכם קוד הקולט מספר בין 1 ל-6 ומציג כפלט הודעה המציינת אם המספר זוגי או אי-זוגי. אם המספר אינו בין 1 ל-6, תיכתב ההודעה

'הקלדת מספר שגוי'. קטע הקוד משתמש בעובדה כי ההוראות מתבצעות עד להוראת ה-break הראשונה.

```
<!—EvenOdd.htm-->
<script type="text/javascript">
   var num =prompt("הקלד מספר");
   switch (parseInt(num))
   {
      case 1:
      case 3:
      case 5:
         document.write("מספר אי-זוגי");
        break;
      case 2:
      case 4:
      case 6:
         document.write("מספר זוגי");
         break;
      default:
         document.write("הקלדת מספר שגוי");
         break;
   }
</script>
```

שאלה 5.12

א. פתרו את שאלה 5.6 באמצעות הוראת התנאי switch.

ב. פתרו את שאלה 5.10 באמצעות הוראת התנאי switch.

הוראות לביצוע-חוזר (לולאות)

כל שפת תכנות מכילה מבני בקרה המאפשרים ביצוע-חוזר של קטע קוד, מספר פעמים בזו אחר זו. קטע הקוד הזה נקרא **לולאה (loop)**. קיימים מספר מבני חזרה אך אנו נתייחס

ללולאה במבנה 'בצע-מספר-פעמים', היא לולאת ה-for, ולולאה ל'ביצוע-חוזר-בתנאי' – היא לולאת ה-while.

1. **הלולאה for** משמשת לביצוע-חוזר במספר ידוע מראש של פעמים. התחביר של לולאה זו מגדיר את הערך הראשוני של משתנה הלולאה, את ערכו האחרון של משתנה הלולאה ואת הערך המקדם את משתנה הלולאה. להלן דף HTML הנוצר באמצעות הלולאה for הכותבת את המילה 'שלום' ככותרת מרמה משתנה בין 1 ל-6:

```
<!--forLoop.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>Learning JavaScript</title>
   <script type="text/javascript">
      var i;
      for(i=1; i<7; i++)
      {
         document.write("<h"+i+">שלום</h"+i+">");
      }
   </script>
</head>
<body dir="rtl">
</body>
</html>
```

נסתכל על ההוראה for(i=1; i<7; i++). הוראה זו מאתחלת את משתנה הלולאה i ל-7, מקדמת אותו ב-1 בכל ביצוע-חוזר ובודקת שערכו קטן מ-7.

הדף שיתקבל מהרצת קוד זה הוא:

איור 7-5
פלט של ביצוע-חוזר מספר פעמים ידוע

הלולאה while משמשת לביצוע-חוזר מספר פעמים לאו דווקא ידוע מראש. התחביר של לולאה זו משתמש בערך ההתחלתי של משתנה הלולאה, שניתן למשתנה לפני הלולאה, ובתנאי המתייחס לערכו האחרון של משתנה הלולאה. את קידומו של הערך הזה יש לבצע בתוך הלולאה. בכדי לבצע את המשימה הקודמת בלולאה while, יש לכתוב את הקוד הבא:

```
<!—whileLopp.htm-->
<script type="text/javascript">
   var i=1;
   while(i<7)
   {
      document.write("<h"+i+">שלום</h"+i+">");
      i++;
   }
</script>
```

במקרה זה, משתנה הלולאה i אותחל לפני הלולאה, והמשתנה קודם בתוך הלולאה.

כפי שכבר ציינו, ייחודה של הלולאה while בכך שניתן לבצע מספר לא ידוע מראש של חזרות. הכיצד?

לדוגמה, נכתוב תכנית הקולטת מספרים חיוביים מהמשתמש עד לקליטת המספר השלילי הראשון. התכנית תמנה את כמות המספרים הזוגיים ואת כמות המספרים האי-זוגיים

שנקלטו. ברור כי אין לבצע משימה זו בעזרת הלולאה for שכן מספר החזרות תלוי בקלט ואינו ידוע מראש.

התכנית תגדיר שני משתנים שישמשו כמונים של מספר הזוגיים ומספר האי-זוגיים שיקלטו. מונים אלו יאותחלו לאפס. כמו כן תגדיר התכנית משתנה לולאה בשם num אשר יקלוט את הערכים מהמשתמש. הקליטה תתבצע בתוך הלולאה, אך כיוון שהכניסה ללולאה מותנית בערך חיובי של num, יש לקלוט את המספר הראשון <u>לפני</u> הלולאה. בתוך הלולאה ייבדק ערכו של num, ובהתאם לערכו יעודכן המונה המתאים. בסיום הלולאה תודפס הודעה מתאימה.

להלן התכנית:

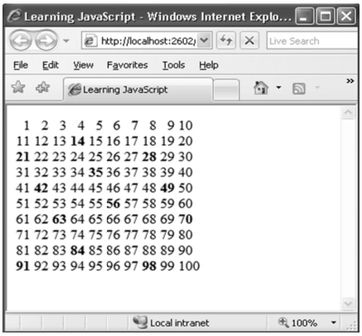
```
<!--WhileCountEvenOdd.htm-->
<script type="text/javascript">
   var countOdd = 0;
   var countEven = 0;
   var num =prompt("הכנס מספר חיובי. הכנס מספר שלילי לסיום");
   while(num > 0)
   {
      if(num%2==0)
         countEven++;
      else
         countOdd++;
      num =prompt("הכנס מספר חיובי. הכנס מספר שלילי לסיום");
   }
   document.write("even: "+countEven+"  odd: "+countOdd);
</script>
```

ניתן לקנן לולאות, כלומר לכתוב לולאה בתוך לולאה. נדגים זאת בתכנית אשר תדפיס את לוח המאה, ושבה יודגש כל מספר המתחלק ב-7. התכנית תגדיר לולאת for מקוננת. המשתנה i ייצג את מספר השורה והמשתנה j – את מספר העמודה. ערכו של המספר בשורה 3 עמודה 5 הוא 25 (בדוק!) על כן ניתן ל-i ערכים בין 0 ל-9 ול-j ערכים בין 1 ל-10 ונחשב את הערך בשורה i עמודה j על-ידי הנוסחה i*10+j. בכדי להציג את לוח המאה כך

שכל המספרים יעמדו זה מעל זה בעמודות המתאימות, נוסיף רווח נוסף לפני המספרים 1..9.

```
<!-- NestFor.htm -->
<script type="text/javascript">
  var i , j;
  for(i=0; i<=9; i++)
  {
    for(j=1; j<=10; j++)
    {
      if(i*10+j < 10)
        document.write("  ");
      if((i*10+j)%7==0)
      {
        document.write("<b>");
        document.write(i*10+j+" ");
        document.write("</b>");
      }
      else
        document.write(i*10+j+" ");
      if(j==10)
        document.write("<br />");
    }
  }
</script>
```

להלן לוח המאה, כפי שמודפס על-ידי קוד התכנית שכתבנו:



איור 5-8

לוח המאה בו המספרים המתחלקים ב- 7 מודגשים

שאלה 5.13

כתבו תכנית המדפיסה את המספרים בין 1 ל-100 בעשר שורות – עשרה מספרים בשורה – ושבה המספרים הזוגיים נכתבים בצבע אדום והאי-זוגיים – בכחול.

שאלה 5.14

כתבו תכנית הקולטת חמישה ציונים והמדפיסה את ממוצע הציונים הללו.

שאלה 5.15

כתבו תכנית המדפיסה את לוח הכפל. התכנית צריכה להשתמש בשתי לולאות for מקוננות ולהדפיס 11 שורות של 11 מספרים. בתכנית, השורה העליונה והעמודה השמאלית מייצגות את המספרים הנכפלים והן נצבעות בצבע שונה.

שאלה 5.16

כתבו תכנית הקולטת ציונים (מספרים שלמים חיוביים) עד אשר נקלט מספר שלילי, והמדפיסה את ממוצע הציונים הללו.

שאלה 5.17

כתבו תכנית הקולטת מספרים עד שסכומם של המספרים שנקלט גדול מ-1,000. התכנית מדפיסה את כמות המספרים שקלטה.

שאלה 5.18

כתבו תכנית הקולטת מספרים חיוביים עד לקליטת 0 והמחשבת את סכומם של המספרים הגדולים מ-20.

5.4 פונקציות

פונקציה היא קבוצת הוראות בתכנית מחשב, לביצוע משימה מסוימת. כתיבת פונקציה משפרת את קריאוּת התכנית ומקלה את תיקון השגיאות בה (מבניות) וכן מייעלת את הכתיבה שכן היא מונעת שכפול קוד (יעילות). פונקציה יכולה לקבל ארגומנטים ולהחזיר ערך. כדי להשתמש בפונקציה, צריך תחילה להגדירה. **ניתן להגדיר פונקציות אך ורק באזור התחום על יד התג HEAD.**

נסתכל על כמה דוגמאות המרוכזות בקובץ Functions.htm :

```
<!--Functions.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
   <title>Learning JavaScript</title>
   <script type ="text/javascript">
      function printHeader()
      {
         document.write("<b>תרגיל</b>");
      }
```

פונקציה שאין לה פרמטרים ואינה מחזירה ערך. הפונקציה מדפיסה כותרת

```
      function printEx(x, y)
      {
         var sum = x+y;
         document.write("<p dir='ltr'>");
         document.write(x+"+"+y+"="+sum);
         document.write("</p>");
      }
```

פונקציה שיש לה שני פרמטרים שהם מספר אך היא אינה מחזירה ערך.
הפונקציה מחשבת ומציגה את תרגיל החיבור בין שני המספרים

```
      function getNumber()
      {
         var num =prompt("מספר הכנס");
         return parseInt(num);
      }
```

פונקציה שאין לה פרמטרים אך מחזירה ערך. הפונקציה קולטת ערך מהמשתמש ומחזירה את ערכו המספרי. החזרת ערך מתבצעת במשפט return.

```
      function getInput()
      {
```

פונקציה שאין לה פרמטרים אך מחזירה ערך. הפונקציה קולטת מהמשתמש ערך ומחזירה ערך מהטיפוס string

```
        var val =prompt("ערך הכנס");
        return val;
    }

    function calculateEx(x, y, op)
    {
        var st = "<p dir='ltr'>"+x+op+y+"=";
        switch(op)
        {
            case "+":
              st += (x+y);
              break;
            case "-":
              st += x-y;
              break;
            case "*":
              st += x*y;
              break;
            case "/":
              st += x/y;
              break;
            case "%":
              st += x%y;
              break;
            default:
              st += "???";
              break;
        }
        st+="</p>";
        return st;
    }
  </script>
</head>
<body dir="rtl">
```

פונקציה שיש לה שלושה פרמטרים: שני מספרים וסימן פעולה ביניהם ומחזירה מחרוזת ובה התרגיל המתאים ופתרונו. אם סימן הפעולה שמתקבל אינו מייצג אחת מבין הפעולות %,/,*,-,+, יירשם ערך הביטוי כסימני שאלה (???)

```
  <h1>תרגילים דף</h1>
  <script type="text/javascript">
    for(i=1; i<3; i++)
    {
      printHeader();
      var n1 = getNumber();
      var n2 = getNumber();
      printEx(n1, n2);
    }

    for(i=1; i<3; i++)
    {
      printHeader();
      var n1 = getNumber();
      var n2 = getNumber();
      var op = getInput();
      document.write(calculateEx(parseInt(n1), parseInt(n2), op)+"<br />");
    }
  </script>
</body>
</html>
```

זימון הפונקציות מתבצע בין תגים <script>...</script> באזור התחום על ידי התג BODY

זימון פונקציה ללא פרמטרים ושאינה מחזירה ערך

זימון פונקציה המקבלת פרמטרים

זימון פונקציה ללא פרמטרים המחזירה ערך. הערך המוחזר נשמר במשתנה

זימון פונקציה עם פרמטרים המחזירה ערך. הערך המוחזר מועבר לפעולה write

כפי שראינו, זימון הפונקציות מתבצע בין התגים <script>...</script> באזור התחום על ידי התג BODY. ואולם אפשר לזמן פונקציה גם באמצעות תג הקישור <a> והמאפיין href, אשר ערכו יהיה מחרוזת המורכבת מהמילה :javascript ואחריה שם הפונקציה לזימון.

```
<!--FunctionByHref.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Learning JavaScript</title>
  <script type ="text/javascript">
    function changeBgColor(color)
    {
      document.bgColor = color;
    }
  </script>
```

```
</head>
<body dir="rtl">
  <h1>קישור באמצעות פונקציה זימון</h1>
  <br />
  <a href="javascript:changeBgColor('lime')">שנה צבע רקע לירוק בהיר </a><br />
  <a href="javascript:changeBgColor('red')">שנה צבע רקע לאדום </a><br />
  <a href="javascript:changeBgColor('yellow')">שנה צבע רקע לצהוב </a><br />
  <a href="javascript:changeBgColor('gray')">שנה צבע רקע לאפור</a><br />
  <a href="javascript:changeBgColor('white')">שנה צבע רקע ללבן</a><br />
</body>
</html>
```

דף זה יציג כותרת ואחריו חמישה קישורים לשינוי צבע. כל השינויים נעשים על-ידי זימון הפעולה changeBgColor, אך בכל פעם הערך יהיה שונה.

שאלה 5.19

כתוב פונקציה המקבלת שני מספרים ומחזירה את המספר הגדול מביניהם.

שאלה 5.20

כתוב פונקציה הקולטת מספרים חיוביים מהמשתמש, עד לקליטת הערך אפס, ומחזירה את הערך המקסימלי מבין המספרים שנקלטו.

5.5 שימוש בעצמים מוגדרים מראש

העצמים window ו-document

כאמור, JavaScript היא שפה מונחית עצמים המכילה עצמים מובנים מראש בהם ניתן להשתמש. לדוגמה, אנו השתמשנו כבר בפעולה alert של העצם window ובפעולה write של העצם document.

העצם window מייצג חלון פתוח בדפדפן. העצם הזה נוצר בעת פתיחת חלון והוא מכיל בין היתר את המאפיינים location (כתובת URL הנוכחית) ו-history (היסטוריה של אתרים

בהם גלשנו) ועצמים נוספים שמוכלים בתוכו, כגון document המטפל בדף HTRL. עצם זה מספק פעולות לפתיחת חלון, סגירת החלון, מזעור החלון וכדומה. העצם הזה הוא העצם הראשי ואפשר שלא לציינו בעת הקריאה לפעולותיו, על כן ניתן לכתוב alert() אך ניתן גם לכתוב window.alert(). כאמור, כל הפעולות בדפדפן מתבצעות בחלון. עצם החלון הוא העצם העיקרי בשפת javascript והוא מאפשר לקלוט נתונים מהמשתמש, להציג פלט למשתמש ועוד.

עצם המסמך (document) מתייחס למסמך HTML, וליתר דיוק להגדרות הנכללות בין התגים <body>...</body>. העצם הזה מכיל מידע על המסמך שנטען ונמצא בתוך חלון, כלומר בתוך מופע של העצם window. העצם document מכיל אף הוא עצמים כגון form, img ועוד. נוסף על כך, לעצם document ישנם מאפיינים ופעולות. לדוגמה, המאפיין bgColor של העצם הזה מתייחס לצבע הרקע של הדף. ניתן לקבוע את צבע הרקע על-ידי השמה למאפיין זה, או להציגו בהוראה alert. פעולות שניתן לבצע עם עצם זה הן הפעולה write לכתיבת קוד HTML או תסריט, הפעולה open והפעולה close לפתיחת מסמך וסגירתו ועוד.

לדוגמה, התסריט הבא מבקש מהמשתמש להקליד שם של צבע (באנגלית כמובן) ומחליף את צבע הרקע של המסמך לצבע שבחר המשתמש. הפעם נשתמש בשם המלא של הפעולה, כלומר נתייחס גם לשם העצם שאליו שייכת הפעולה:

```
<script type ="text/javascript">
   var color = window.prompt("צבע הכנס","red");
   window.document.bgColor = color;
</script>
```

קיימים כמובן עצמים נוספים שהם מובנים בשפת JavaScript כגון העצם Form שמטפל בטופס HTML, Body שמטפל בגוף של מסמך HTML ועוד. במסגרת ספר זה נציג עקרונות של שימוש בעצמים ולכן לא נתאר את כולם.

העצם מחרוזת (String)

העצם מחרוזת מייצג רצף של תווים המורכבת מאותיות האלף-בית בכל שפה, ספרות ותווים אחרים. ניתן להגדיר משתנה מטיפוס מחרוזת ולשים בה ערך, לדוגמה:

```
str="abc";
```

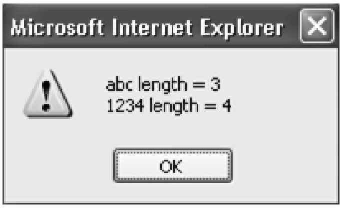
המרכאות מציינות כי הערך הוא מהטיפוס מחרוזת.

ראשית, נסקור את המאפיינים והפעולות של העצם מחרוזת:

המאפיינים:

התכונה	**משמעותה**
length	מכילה את אורך המחרוזת, כלומר את מספר התווים שבה.

שימו לב כי התכונה אינה מוגדרת בעבור מספרים, אלא בעבור מחרוזות בלבד. נדגים זאת באמצעות שתי הדוגמאות הבאות:

ההודעה הקופצת:	**הקוד:**
	<script type="text/javascript"> var st="abc"; var stNum="1234"; alert(st+ " length = " + st.length+"\n"); alert(stNum+ " length = " + st.length+"\n"); </script>
	<script type="text/javascript"> var num=1234; alert(num + " length = " + num.length+"\n"); </script>

כאשר הערך 1234 הוגדר כמחרוזת (על-ידי סימונו במרכאות), ניתן היה להשתמש בתכונה length. אך כאשר ערך זה הוגדר כמספר התכונה, length מחזירה "undefined" שכן התכונה אינה מוגדרת עבור מספר.

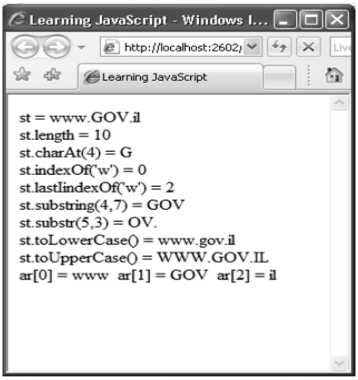
הפעולות:

הפעולה והפרמטרים	משמעותה
charAt(k)	הפעולה מקבלת מספר שלם k המייצג את המיקום במחרוזת, ומחזירה את התו הנמצא במיקום הזה. נדגיש כי האות הראשונה במחרוזת נמצאת במקום 0.
indexOf(ch)	הפעולה מקבלת את התו ch ומחזירה את המיקום הראשון של התו במחרוזת. אם התו לא נמצא במחרוזת, היא מחזירה 1–.
lastIndexOf(ch)	הפעולה מקבלת תו ch ומחזירה את המיקום האחרון של התו במחרוזת. אם התו לא נמצא במחרוזת, היא מחזירה 1–.
substring(from, to)	הפעולה מקבלת שני מיקומים במחרוזת (מספרים שלמים) ומחזירה תת-מחרוזת של המחרוזת, המורכבת מהתווים שבתאים שבין from ל-to,לא כולל התו שבמיקום to.
substr(from, len)	הפעולה מקבלת שני שלמים: הראשון מציין את המיקום במחרוזת from והשני את אורך תת-המחרוזת. הפעולה מחזירה תת-מחרוזת, המורכבת מהתווים, החל בתו שבמיקום from ובאורך len.
toUpperCase(string st)	הפעולה מקבלת מחרוזת אותיות, ומחזירה את המחרוזת באותיות גדולות.
toLowerCase(string st)	הפעולה מקבלת מחרוזת, ומחזירה את המחרוזת באותיות קטנות.
split(ch)	הפעולה מקבלת תו ch המשמש כתו הפרדה, ומחזירה מערך של תתי-מחרוזת של המחרוזת המקורית המופרדים בתו ההפרדה ch.

להלן דוגמה לשימוש במאפיינים ובפעולות של העצם String. שימו לב כי התווים במחרוזת ממוקמים החל מהמיקום 0, לפיכך, הפעולה st.charAt(1) תחזיר את התו השני במחרוזת.

```
<!--StringSample.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Learning JavaScript</title>
  <script type="text/javascript">
   var st = "www.GOV.il";
   document.write("st = "+st+"<br />");
   document.write("st.length = "+st.length+"<br />");
   document.write("st.charAt(4) = "+st.charAt(4)+"<br />");
   document.write("st.indexOf('w') = "+st.indexOf("w")+"<br />");
   document.write("st.laslindexOf('w') = "+st.lastIndexOf("w")+"<br />");
   document.write("st.substring(4,7) = "+st.substring(4,7)+"<br />");
   document.write("st.substr(5,3) = "+st.substr(5,3)+"<br />");
   document.write("st.toLowerCase() = "+st.toLowerCase()+"<br />");
   document.write("st.toUpperCase() = "+st.toUpperCase()+"<br />");
   var ar = st.split(".");
   for(i=0; i<ar.length; i++)
   {
      document.write("ar["+i+"] = "+ar[i]+"   ");
   }
   document.write("<br />");
   </script>
</head>
<body>
</body>
</html>
```

הדף המתקבל הוא:



איור 9-5
פלט שנוצר מעיבוד מחרוזות

פונקציה שימושית נוספת (שהיא אינה פעולה של העצם String אך היא מוכרת על-ידי כל העצמים) היא הפונקציה **isNaN(*x*)**[2]. פונקציה זו מקבלת ערך מחרוזתי או מספרי, ומחזירה את הערך 'אמתי' (true) אם הערך מחרוזתי ולא מספרי, ואת הערך 'שקרי' (false) בכל מקרה אחר (**N**ot **a** **N**umber). לדוגמה, הפלט של התסריט הזה:

```
  <script type ="text/javascript">
            document.write(isNaN(123)+ "<br />");
            document.write(isNaN("Hello")+ "<br />");
            document.write(isNaN("2005/12/12")+ "<br />");
</script>
```

הוא:

```
false
true
true
```

[2] הפונקציה isNaN היא פונקציה גלובלית וניתן להשתמש בה עם כל אחד מהעצמים המובנים בשפת JavaScript.

5.6 טיפול באירועים

שפת JavaScript היא שפה מונחית-אירועים, כלומר היא יודעת להגיב על אירועים המתרחשים כתגובה לפעולות המשתמש, כגון לחיצה על כפתור, סגירת חלון, הקלקה על העכבר, וכן לאירועים המתרחשים כתגובה לפעולת שעון. תגובה לאירוע מפעילה את המתפעל אירועים (event-handler) אשר יכול לזמן פונקציה או לבצע תסריט שמטרתם לטפל באירוע שנוצר.
לפניכם רשימה חלקית של אירועים :

טבלה 5-1 רשימת אירועים

Event-handler	**העצם**	**האירוע**
Onblur	form	עצם מאבד מיקוד, למשל, בעת מילוי שדה טופס ומעבר לשדה אחר
Onchange	form	האם חל שינוי בערכו של עצם שאיבד את המיקוד שלו
Onclick	mouse	לחיצה על עצם לחיץ, למשל, כפתור
Onfocus	form	עצם נבחר ומקבל מיקוד
Onload	body	טעינת מסמך. ניתן לזמנו בתג body
Onmouseout	mouse	העברת עכבר מעל עצם
Onmouseover	mouse	יציאת עכבר מתחום העצם
Onreset	form	לחיצה על כפתור מהטיפוס reset
Onselect	form	בחירת ערך מרשימת ערכים (רשימה נגללת) בטופס
Onsubmit	form	לחיצה על כפתור מהטיפוס submit
Onunload	body	עזיבת הדף. ניתן לזמנו בתג body

לדוגמה, נסתכל על הדף הבא. הדף מכיל כותרת מרמה 1, שלוש פסקאות ותמונה מתחלפת. כאשר נעביר את העכבר מעל הפסקה הראשונה, צבע הרקע של הפסקה יהפוך לאדום ויישאר במצב הזה. כאשר נעביר את העכבר מעל הפסקה השנייה, צבע הרקע של הפסקה

יהפוך לאדום, אך כאשר נזיז את העכבר אל מחוץ לתחום הפסקה, יחזור הרקע לצבעו הקודם. כאשר נעביר את העכבר על התמונה, תתחלף התמונה. אבל כאשר נזיז את העכבר אל מחוץ לשטח התמונה, תחזור התמונה המקורית. נדגיש כי בכדי שאפשר יהיה לגשת לעצם התמונה ולשנות אותו יש לתת לעצם שם ולגשת אליו דרך שמו, לכן בדוגמה הבאה רשמנו את השם "name="whiteCat כחלק מהתג img. כמו כן, בתוכנית זו אנו מבצעים שינויים בתוכן של הדף, ולכן נרשום את התסריטים בתוך האזור התחום על ידי התג BODY.

```
<!--Events.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Events Samples</title>
</head>
<body dir="rtl" style="background-color:Green; color:Yellow;">
   <form id="form1" runat="server">
   <div>
       <h1 style="color:Red"> הדגמת אירועים</h1>
       <p style="font-size:medium; color:Blue"
            onmouseover="style.backgroundColor='red';">
            onmouseover<br /> פִּסְקָה זו תדגים את האירוע onmouseover
            בעת העברת העכבר על הפִּסְקָה , ישתנה צבע הרקע של הפִּסְקָה לאדום
       </p>
       <p style="font-size:medium; color:Yellow"
         onmouseover="style.backgroundColor='red';"
         onmouseout="style.backgroundColor='green';">
         <br /> העברת עכבר על הפסקה הקודמת שינתה את צבע הרקע של הפסקה,
         <br /> אך הצבע לא חזר לצבעו המקורי עם הזזת העכבר אל מחוץ לאזור הפסקה.
         <br /> פסקה זו מפעילה שני אירועים. האחד onmouseover
         <br /> והשני onmouseout אשר ידאג להחזרת הצבע המקורי.
       </p>
       <p>
         <br /> ניתן כמובן להחליף תמונה בעת העברת העכבר.
         לפניכם הדוגמה, רק זכרו להחזיר את התמונה המקורית לאחר הזזת העכבר
```

```
                        <br />
            <img name="whiteCat" src="Images/cat1.jpg" alt="cat changes"
                onmousemove="document.whiteCat.src='Images/cat2.jpg';"
                onmouseout="document.whiteCat.src='Images/cat1.jpg';" />
        </p>
    </div>
    </form>
</body>
</html>
```

אפשר כמובן לשלב אירועים ופונקציות כך שבעת התרחשותו של אירוע, תופעל הפונקציה שהוגדרה קודם לכן. למשל, נתייחס לטופס שבו נדרש המשתמש לציין את מינו ומופיעה ברכה מתאימה על כניסתו לאתר. נשתמש לשם כך בכפתורי הרדיו ונבדוק את בחירתו של המשתמש. על-פי ערך בחירתו תוצג אחת מן ההודעות האלה: "ברוך הבא לאתרנו", או "ברוכה הבאה לאתרנו". בדיקת הערך שבחר המשתמש ויצירת המחרוזת המתאימה להודעה ייעשו בפעולה בשם WelcomeByGender, אשר תוגדר תחת התג <script> באזור ה-<head>. הקריאה לפעולה תתבצע בעת התרחשותו של האירוע onclick בכפתור send.

```
<!--EventsFunctions.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>  Events Functions </title>
    <script type="text/javascript">
    function WelcomeByGender()
    {
        var gender;
        var welcome;
        for(i=0; i<form1.radioGender.length; i++)
        {
            if (form1.radioGender[i].checked)
                gender = form1.radioGender[i].value;
        }
        if(gender==0)
```

```
            welcome = " ברוך הבא";
        else
            welcome = "ברוכה הבאה";
        welcome += " לאתרנו";

        return welcome;
    }
    </script>
</head>
<body dir="rtl">
    <form id="form1" runat="server">
    <div>
        <h3>סמן את מינך</h3>
        <input type="radio" id="radioGenderF"
                name="radioGender" value="1" />נקבה<br />
        <input type="radio" id="radioGenderM"
                name="radioGender"  value="0" />זכר<br />
        <br />
        <input type="submit" id="btnSend" value="שלח"
                onclick="alert(WelcomeByGender())"/>
    </div>
    </form>
</body>
</html>
```

וכך ייראו ההודעות למשתמש ולמשתמשת:

איור 5-10
הצגת הודעות ברכה

ראינו כי באמצעות בתסריט JavaScript אפשר לגשת לשדות הטופס דרך שמם. למעשה, ניתן לגשת לכל תג HTML בתנאי שנגדיר לתג את המאפיין id. בעזרת הפעולה getElementById, השייכת לעצם document, נוכל לגשת לעצם שהתג מייצג. נוסף על כך, נשתמש במאפיין innerHTML של העצם בכדי להציג את תגי ה-HTML הפנימיים שלו או לשנותם.

נתחיל בדוגמה פשוטה לשימוש משולב בפעולה getElementById ובמאפיין innerHTML. ניצור דף HTML המכיל פסקה המקפיצה בהקלקת עכבר הודעה שמציגה את תוכנה, כולל תגי ה-HTML שהיא מכילה. לשם כך, נכתוב פעולה בשם getInnerHtml אשר תקבל את המאפיין id של עצם HTML, תשתמש בפעולה getElementById בכדי לשמור עותק של העצם ותציג את תוכנו תוך כדי שימוש בפעולה alert ובמאפיין innerHTML :

```
<!—GetElementByID1.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Learning JavaScript</title>
  <script type="text/javascript">
   function getInnerHtml(idStr)
   {
     var id = document.getElementById(idStr);
     alert(id.innerHTML);
    }
  </script>
</head>
<body dir="rtl">
<h1> שימוש ב - getElementById  </h1>
    <p id="clickP" onclick ="getInnerHtml('clickP')">
                          הקליקו על פסקה זו בעזרת <big>העכבר</big>
    </p>
</body>
</html>
```

הרצת הדף הזה תציג את הדף הזה :

איור 5-11
אירוע עכבר

והקלקה בעכבר על פסקה זו תגרום להופעת ההודעה :

איור 5-12
הודעה המופיעה כתוצאה מאירוע עכבר[3]

בדוגמה הזאת הצגנו את קוד ה-HTML הפנימי (התגים <big> ו- </big>). אבל אנו יכולים לשנות את הטקסט ואת תגי ה-HTML של העצם וכן את סגנונו. נוסיף לדוגמה פסקה. העברת עכבר מעליה תשנה את הטקסט בפסקה ואת סגנונה. לשם כך, נכתוב פעולה בשם changeInnerHtml, שתשתמש בפעולה getElementById כדי לשמור עותק של העצם ותשנה את תוכנו של המאפיין innerHTML. נוסף על כך, נגדיר לפסקה צבע אדום. לשם כך, נשתמש במאפיין style ובתכונה color. ההפניה למאפיין style תיעשה דרך ההפניה לעצם, והפניה לתכונה תיעשה על-ידי השימוש ב '.' לאחר הפניה ל-style.

```
<!—GetElementByID2.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
```

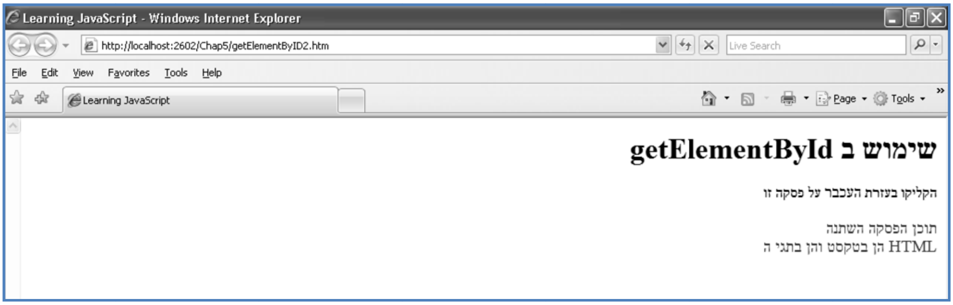
3 הכיתוב בתיבת ההודעה הוא משמאל לימין ובספר זה לא נטפל באפשרויות של הצגה מימין לשמאל

```
<head>
  <title>Learning JavaScript</title>
  <script type="text/javascript">

<script type="text/javascript">
    function getInnerHtml(idStr) {
        var id = document.getElementById(idStr);
        alert(id.innerHTML);
    }
    function changeInnerHtml() {
        var id = document.getElementById("overP");
        id.innerHTML = "<big> תוכן הפסקה השתנה <br>HTML הן בטקסט והן בתגי ה </big>";
        id.style.color = "Red";
    }
  </script>

</head>
<body dir="rtl">
    <h1> getElementById –שימוש ב </h1>
    <p id="clickP" onclick ="getInnerHtml('clickP')">
        הקליקו ב <big>העכבר</big> על פסקה זו
    </p>
    <p id="overP" onmousemove="changeInnerHtml()" >
        העבירו את העכבר מעל פסקה זו <br />
        האם הבחנתם בשינוי? <br />
    </p>
</body>
</html>
```

הרצת הדף תציג את הדף הזה:

איור 5-13
אירוע עכבר

העברת העכבר מעל הפסקה השנייה תגרום לשינוי הטקסט, והדף ייראה כך :

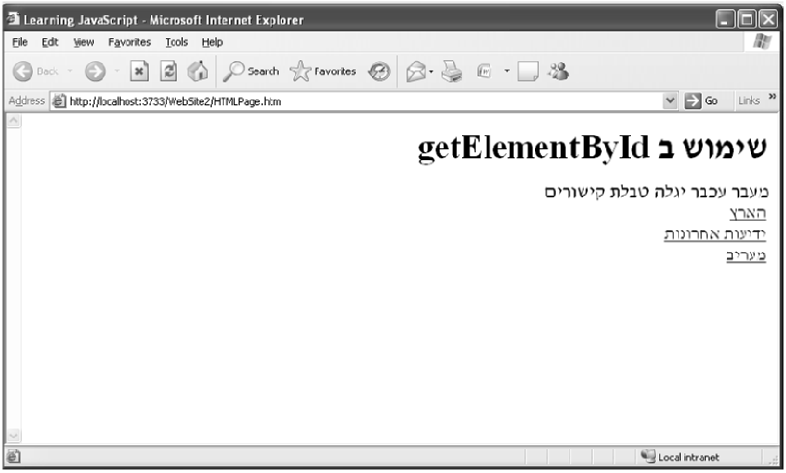


איור 5-14
תוצאת אירוע עכבר

אפשר להשתמש בפעולה getElementById גם כדי להציג או להסתיר מידע. נוסיף לדף פסקה המכילה כותרת וטבלה של קישורים לאתריהם של עיתונים ישראלים. ניתן ערך למאפיין id בטבלה ובעת טעינת הדף נסתיר את הטבלה על-ידי האירוע onload של התג <body>. העברת העכבר על שורת הכותרת של הפסקה תגרום להצגת הטבלה. מובן שהזזת העכבר אל מחוץ לאזור הפסקה תסתיר את הטבלה, כלומר את רשימת הקישורים. לשם כך נכתוב שתי פעולות : האחת showLinks והשנייה hideLinks, אשר יקבלו את המזהה (identifier) של המידע שמבקשים לחשוף או להסתיר. בתג <table> של הטבלה נגדיר אירועים של העברת העכבר והזזת עכבר אל מחוץ לתחום המוגדר ונקרא לפעולות בהתאמה.

```
<!—GetElementByID3.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>Learning JavaScript</title>
  <script type="text/javascript">
  function showLinks(idStr)
  {
    var id = document.getElementById(idStr);
    id.style.display="block";
  }
  function hideLinks(idStr)
  {
    document.getElementById(idStr).style.display="none";
  }
  </script>
</head>
<body dir="rtl" onload="hideLinks('links')">
    <h1>שימוש ב getElementById</h1>
      <p onmousemove="showLinks('links')"
      onmouseout="hideLinks('links')">
      <big>מעבר עכבר יגלה טבלת קישורים </big><br />
      <table id="links">
        <tr><td>
          <a href ="http://www.haaretz.co.il">הארץ</a>
        </td></tr>
        <tr><td>
          <a href ="http://www.ynet.co.il"> ידיעות אחרונות</a>
        </td></tr>
        <tr><td>
          <a href ="http://www.nrg.co.il">מעריב</a>
        </td></tr>
       </table>
    </p>
</body>
</html>
```

נעביר את העכבר מעל שורת הכותרת ונקבל את הדף הזה :



איור 5-15
תוצאת אירוע עכבר – הצגת קישורים

כאשר העכבר יהיה מחוץ לגבולות הכותרת, נקבל רק את הטקסט הזה :

שימוש ב getElementById מעבר עכבר יגלה טבלת קישורים

בצורה דומה ניתן להסתיר פסקאות, לכתוב תפריטים נגלים ונסתרים ועוד. יש לשים לב כי כיוון שאנו רוצים להשתמש בקישורים הגלויים לעין, עליהם להיות מוגדרים כחלק מהפסקה ולכן גם הטבלה צריכה להיות מוגדרת כחלק ממנה.

פעולה שימושית נוספת של העצם window היא הפעולה ()setInterval. פעולה זו מקבלת כפרמטר מרווח זמן במילישניות, והיא מזמנת פונקציה או ביטוי במרווחי הזמן שהוגדרו. לדוגמה, נציג דף שמדגים את השעון של המערכת כל 50 מילישניות.

```
<!-- Clock.htm-->
<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
```

```
<input type="text" id="clock" size="35" />
<script language=javascript>
var int=self.setInterval("clock()",50)
function clock()
  {
                 var t=new Date()
                document.getElementById("clock").value=t;
          }
</script>
</form>
</body>
</html>
```

שאלה למחשבה

מהו ההבדל בין דף זה, המציג את הזמן, לבין הדף להצגת זמן שהצגנו בתחילת הפרק?

בדף הראשון שהצגנו הזמן מוצג רק פעם אחת, ואילו כאן הדף מתעדכן בכל 50 מילישניות. פונקציה זו שימושית גם בעת הצגת אנימציה על-ידי רענון התצוגה במרווחי זמן קבועים.

שאלה 5.21

צרו מסמך המכיל שלושה כפתורי submit הגורמים לשינוי צבע הרקע של המסמך באמצעות לחיצה. השתמשו בפונקציה עם פרמטר וזַמנו אותה בעת אירוע onclick.

שאלה 5.22

צרו מסמך המכיל תיבת טקסט, הכפתור reset והכפתור submit. לחיצה על הכפתור reset תשנה את צבע הרקע של הדף. לחיצה על הכפתור submit תדפיס הודעה שבה הטקסט שהוקלד בתיבת הטקסט (אם הוקלד טקסט).

שאלה 5.23

צרו מסמך המכיל פסקה כזאת, שכאשר מעבירים עכבר מעליה ומחוצה לה יתחלף הגופן בפסקה.

5.7 כתיבת תסריט לבדיקת תקינותם של נתוני טופס

כאמור, מטרתה הראשונה של שפת התסריט היא בדיקת תקינותו של טופס. נזכיר כי טופס הוא עצם של העצם **חלון** (**window**), ותפקידו לאסוף נתונים מהמשתמש ולשלחם למנהל האתר או למסד הנתונים.

נבנה טופס רישום בסיסי המכיל את השדות האלה: שדה טקסט לשם המשתמש, שני שדות סיסמה – לסיסמה ולאימותה, שדה טקסט נוסף לכתובת הדוא״ל ושדה נוסף למספר טלפון. הטופס צריך לכלול גם כפתור לשליחת הנתונים. נשים לב כי תוכן הטופס צריך להיות כלול במקטע, ועל כן התג העוקב לתג הטופס <form> הוא התג <div>. שימוש בתג מקטע יאפשר בהמשך להגדיר לכל קטע במסמך עיצוב משלו.

את טופס ההרשמה או טופס הכניסה נבנה כדף שרת שישלח ללקוח. לאחר שהטופס ייבדק בצד הלקוח, הנתונים התקינים ישלחו לשרת. בהמשך נוכל להשתמש בנתונים אלו כדי לעדכן את מסד הנתונים או לאחזר ממנו מידע.

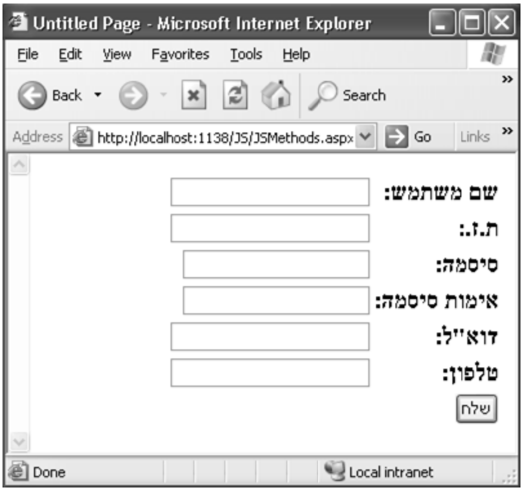
לפניכם הקובץ BaseReg.aspx המציג את הטופס הזה. שימו לב לשמות השדות. השם של תיבת טקסט מתחיל ב-txt, השם של תיבת הסיסמה מתחיל ב-pwd. ניתן כמובן לבחור גם בשמות אחרים, אך חשוב לתת לשדות שמות בעלי משמעות, המגדירים את סוג השדה ותוכנו המקלים את הגישה אליהם בעת כתיבת הקוד.

```
<!--BaseReg.aspx-->
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

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

```
<head runat="server">
  <title>Text and Password Form</title></head>
<body>
   <form id="formBaseReg" name=" formBaseReg" runat="server" >
   <div>
       <table>
           <tr>
               <th align="right"> :שם משתמש </th>
               <td><input type="text" id="txtUserName"/></td>
           </tr>
           <tr>
               <th align="right">:מספר זהות.</th>
               <td><input type="text" id="txtId"/></td>
           </tr>
           <tr>
               <th align="right">:סיסמה</th>
               <td><input type="password" id="pwd" /></td>
           </tr>
           <tr>
               <th align="right">:אימות סיסמה</th>
               <td><input type="password" id="pwdConfirm"/></td>
           </tr>
           <tr>
               <th align="right"> :דוא"ל</th>
               <td><input type="text" id="txtEmail"/></td>
           </tr>
           <tr>
               <th align="right">:טלפון</th>
               <td><input type="text" id="txtTel"/></td>
           </tr>
           <tr>
             <td colspan="2">
               <input type="submit" value="send" />
             </td>
```

```
      </tr>
        </table>
    </div>
    </form>
</body>
</html>
```

להלן הטופס כפי שהוא מוצג למשתמש:



איור 5-16
הצגת דף טופס הרשמה אשר ישמש בסיס להסברים ולתרגילים הבאים

לחיצה על כפתור מהטיפוס submit מפעילה אירוע בשם onsubmit, אשר מזמן פונקציה בוליאנית (valid) הבודקת את הטופס. רק אם הפונקציה תחזיר את הערך true, יישלח הטופס לשרת כדי להפעיל את הפונקציה valid, הבודקת את תקינות הטופס, נשבץ את האירוע onsubmit בתג הטופס:

```
<form id="formBaseReg" onsubmit="return valid()" >
```

הפונקציה עצמה תוגדר בראש הדף, בתוך התג <head> ובתוכו התג <script type="text/javascript">. נציג שוב את הדף עם השינויים הדרושים, אך ללא חזרה על הטבלה המכילה את תיבות הטקסט והכפתור:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
  <title>Text and Password Form</title>
```

```
  <script type="text/javascript">
   function Valid()
   {
     // עריכת בדיקות תקינות
     Return ...
   }
   </script>
</head>
<body dir="rtl">
   <form id="formBaseReg" onsubmit="return valid()" >
   <div>
      <table>
            ...
      </table>
   </div>
   </form>

</body>
</html>
```

קטעי הקוד הנכתבים בין התגים <script></script> מתבצעים מיד כשה-browser מתרגם את הקוד (בעזרת תכנית פירוש).

טופס ב-HTML נוצר על-ידי התג <form> והתג <input /> לסוגיו. התג <form> יוצר עצם מסוג **טופס (form)** בתוך העצם חלון המוכל בעצמו בעצם מהטיפוס **מסמך (document).** כל שדה המוגדר על-ידי התג <input /> יוצר עצם בתוך העצם טופס. נוכל אפוא לגשת לעצם כלשהו דרך ההגדרה ההיררכית שלו. התייחסות לכל עצם והגישה אליו היא באמצעות המזהה הייחודי שלו, המאפיין id. להלן כמה דוגמאות:

- כדי להתייחס לעצם טופס שנוצר באמצעות התג <form> נרשום:

```
window.document.formBaseReg
```

- כדי להתייחס לשדה שם המשתמש, אשר נוצר באמצעות התג <input />, נשתמש בערך ה- id שלו ונרשום:

window.document.formBaseReg.txtUserName

- כדי להתייחס לערך השדה txtUserName , כלומר לשם המשתמש אשר הוקלד בטופס, נרשום:

window.document.formBaseReg.txtUserName.value

בצורה הזאת נתייחס לשדות הטופס כאל משתנים בדף שבו הם מוגדרים.

- הערכים בשדות הטופס הם עצמים מהטיפוס מחרוזות וניתן להשתמש במאפיינים ובפעולות של העצם string כדי לקיים בדיקות על ערכים אלו.

בפעולה valid, המתבצעת בעת לחיצה על הכפתור 'send', נקרא לפעולות השונות בכדי לבדוק אותן. נכתוב כעת כמה פונקציות שעורכות בדיקות תקינות נפוצות בטופס המכיל את פרטי המשתמש.

בדיקות תקינות של הנתונים

בסעיף נפרט את סוגי בדיקות התקינות והפעולות שיש לבצע על נתונים שונים כדי להבטיח את תקינותם.

א. בדיקת התקינות של מספר הזהות

נכתוב פעולה הבודקת אם מספר הזהות שהוקלד בשדה txtId יכול להיות מספר זהות, כלומר אם הוא מורכב מתשע ספרות. בדיקה זו מחולקת לשתי בדיקות: האחת – אם המספר מורכב מתשעה תווים, והשנייה – אם כולם ספרות.

פתרון:

שלב 1 – נשמור את ערך השדה במשתנה בשם stId.

שלב 2 – נוודא כי אורך המחרוזת הוא בדיוק 9. אם לא, נכתוב הודעה למשתמש ונחזיר את הערך false.

שלב 3– נוודא כי המחרוזת בנויה מספרות בלבד, כלומר ()isNaN תחזיר את הערך false. אם לא, נכתוב הודעה למשתמש ונחזיר את הערך false.

שלב 4 – אם הגענו לשלב הזה, אזי כל הדרישות התקיימו ונחזיר את הערך true.

```
function isValidId()
{
   var stId = document.getElementById("txtId");
   if(stId.value.length != 9)
   {
      document.getElementById("idError").innerHTML = "The ID must be 9 digits";
      return false;
   }
   if(isNaN(stId.value))
   {
      document.getElementById("idError").innerHTML = "The ID must contain only digits";
      return false;
   }
   return true;
}
```

ב. בדיקת התקינות של כתובת דוא״ל

נכתוב פעולה הבודקת אם כתובת הדוא״ל שנכתבה בתיבת הטקסט txtEmail מכילה את התווים '@' ו- '.' (נקודה).

פתרון:

שלב 1 – נשמור את ערך השדה במשתנה בשם stEmail.

שלב 2 – נוודא כי התו '@' מופיע במחרוזת. אם לא, נכתוב הודעה למשתמש ונחזיר את הערך false.

שלב 3 – נוודא כי התו '.' (נקודה) מופיע במחרוזת. אם לא, נכתוב הודעה למשתמש ונחזיר את הערך false.

שלב 4 – אם הגענו לשלב הזה, אזי כל הדרישות התקיימו ויוחזר הערך true.

```
function isValidEmail()
{
    var stEmail=document.getElementById("txtEmail").value;
    if(stEmail.indexOf("@")<0)
    {
```

```
        document.getElementById("emailError").innerHTML = "Email must contain a @";
        return false;
    }
    if(stEmail.indexOf(".")<0)
    {
        document.getElementById("emailError").innerHTML = "Email must contain a .";
        return false;
 }
    return true;
}
```

משתמש אשר הקליד כתובת דוא״ל שאין בה התו '@' ואף לא התו '.' (נקודה), יקבל התרעה אחת בלבד, רק את זו המזכירה לו שנדרש התו '@'. רק לאחר שיקליד את התו '@', אם עדיין יהיה חסר התו '.' (נקודה) יקבל על כך התרעה. אבל אנו נעדיף כמובן להתריע על כל השגיאות בהתרעה אחת. כיצד ניתן לעשות זאת? על-ידי הגדרת משתנה מהטיפוס מחרוזת שלתוכה נשרשר את כל השגיאות ויציגן בהתרעה שתופיע בסיום הפעולה. להלן הפונקציה המתוקנת:

```
function isValidEmailOneAlert()
{
    var stEmail=document.getElementById("txtEmail").value;
    var stAlert="";
    if(stEmail.indexOf("@")<0)
        stAlert+="Email must contain a @<br />";
    if(stEmail.indexOf(".")<0)
        stAlert+="Email must contain a '.' <br />";
    if(stAlert=="")
        return true;
    else
    {
        document.getElementById("emailError").innerHTML = stAlert;
        return false;
    }
}
```

המשתנה stEmail מוגדר כמחרוזת ריקה. אם התו '@' אינו נמצא במחרוזת הקלט, נשרשר למחרוזת הקיימת כבר ב-stEmail את המחרוזת "Email must contain a @", ואם התו '.' אינו נמצא במחרוזת הקלט, נשרשר למחרוזת הקיימת ב- stEmail את המחרוזת "Email must contain a '.' ". בסיום כל הערה נוסיף את התג
 שכן את הודעת השגיאה אנו מכניסים למאפיין innerHTML. לבסוף, נבדוק את ערכו של stEmail. אם המחרוזת ריקה, אזי לא נמצאו שגיאות ועל כן נחזיר את הערך true. ואולם אם המחרוזת אינה ריקה, אזי נמצאו שגיאות. נכתוב התרעה למשתמש ובה הודעות על השגיאות שנמצאו ונחזיר את הערך false.

ג. בדיקת תקינותה של סיסמת משתמש

נכתוב פעולה הבודקת אם הסיסמה שבחר המשתמש מכילה ספרות ואותיות.

שאלה למחשבה

האם אנו יכולים לאמת את הסיסמה שהקליד המשתמש?

פתרון:

שלב 1 – נשמור את ערך השדה במשתנה בשם stPwd.

שלב 2 – ניצור משתני עזר: st – מחרוזת ריקה שתכיל את ההודעה למשתמש, digits, chars – אשר יכילו את מספר הספרות ומספר האותיות בערך השדה, בהתאמה. את המשתנים המספריים נאתחל ב-0.

שלב 3 – נעבור על מחרוזת הקלט בלולאה for מתחילתה ועד סופה. את אורך המחרוזת נקבל מהמאפיין length. כדי לבדוק כל תו ולוודא שאינו ספרה ניעזר בפעולה ()isNaN. אל התווים עצמם נגיע באמצעות הפעולה ()charAt.

שלב 4 – בסיום הלולאה נבדוק אם מספר הספרות או מספר האותיות הוא 0. אם כן, נעדכן את מחרוזת ההודעה למשתמש.

שלב 5 – נבדוק אם מחרוזת ההודעה למשתמש ריקה. אם כן, אזי לא נמצאו שגיאות ועל כן נחזיר את הערך true, ואולם אם המחרוזת אינה ריקה, אזי נמצאו שגיאות. נכתוב התרעה למשתמש המכילה הודעות על השגיאות שמצאנו, ונחזיר את הערך false .

```
function isDigitsAndLetters()
{
   var stPwd=document.getElementById("pwd").value;
```

```
  var st="";
  var digits = 0, chars = 0;
  for (i=0;i<stPwd.length;i++)
  {
    if (isNaN(stPwd.charAt(i)))
      chars++;
    else
      digits++;
  }
  if (chars==0||digits==0)
    st+="Strong password must contain at least 1 digit and 1 letter\n";
  if(st=="")
    return true;
  else
  {
    document.getElementById("pwdError").innerHTML = st;
    return false;
  }
}
```

בדיקת תקינותה של סיסמת המשתמש – דרך נוספת

נכתוב פעולה הבודקת אם הסיסמה שבחר המשתמש מורכבת מאותיות גדולות וקטנות. כדי לדעת אם הסיסמה מורכבת מאותיות קטנות בלבד, נעתיק את הסיסמה למשתנה נוסף, נהפוך את כל אותיותיו לאותיות קטנות ונשווה בין שתי המחרוזות. אם המחרוזות זהות, אזי הסיסמה מורכבת מאותיות קטנות בלבד. אם לא, נבדוק באותה צורה אם הסיסמה מורכבת מאותיות גדולות בלבד.

פתרון:

שלב 1 – נשמור את ערך השדה במשתנה בשם stPwd.

שלב 2 – ניצור משתנה עזר stOneCase שאליו נעתיק את הסיסמה ונהפוך את אותיותיה לאותיות קטנות בלבד.

שלב 3 – נבדוק אם מחרוזת המקור ומחרוזת האותיות הקטנות זהות. אם כן, נודיע למשתמש ונחזיר את הערך false.

שלב 4 – אם לא, נעדכן את מחרוזת ההעתק ונהפוך את אותיותיה לאותיות גדולות בלבד. נשווה בינה לבין מחרוזת המקור. אם המחרוזות זהות, נודיע למשתמש ונחזיר את הערך false.

שלב 5 – אם הגענו לשלב הזה, אזי הסיסמה מורכבת מאותיות גדולות וקטנות ועל-כן נחזיר את הערך true.

שימו לב כי הפעם אין צורך לצבור הודעות למשתמש. מחרוזת יכולה להיות מורכבת כולה מאותיות קטנות או מאותיות גדולות, אך לא יכולה משתיהן גם יחד.

```
function isUpperAndLowerCase()
{
   var stPwd=document.getElementById("pwd").value;
   var stOneCase = stPwd.toLowerCase();
   if(stPwd==stOneCase)
   {
      document.getElementById("pwdError").innerHTML = "All LowerCase";
      return false;
   }
   stOneCase = stPwd.toUpperCase();
   if(stPwd==stOneCase)
   {
      document.getElementById("pwdError").innerHTML = "All UpperCase";
      return false;
   }
   return true;
}
```

ד. בדיקת תקינותו של מספר טלפון

נכתוב פעולה הבודקת אם מספר הטלפון שהוקלד תקין.

מספר טלפון הוא תקין אם :

1. הוא מורכב מספרות בלבד, פרט לסימן - המפריד בין הקידומת למספר.

2. הקידומת בעלת שתי ספרות, המספר עצמו בן שבע ספרות.
3. הקידומת בעלת שלוש ספרות, המספר עצמו בן שמונה ספרות.

בפתרון, ראשית נחפש את הסימן – כשנמצא אותו, נפריד את המספר לשתי תת- מחרוזות: תת-המחרוזת שלפני הסימן ותת-המחרוזת שאחריו. נבדוק שכל אחת מתת המחרוזת היא מספר. בפתרון זה נשתמש בפונקציה isNum, הבודקת אם המחרוזת מכילה ספרות בלבד, ואותה נבדוק בהמשך.

פתרון:

שלב 1 – נשמור את ערך השדה במשתנה בשם strTel.

שלב 2 – ניצור משתנה עזר st לשמירת מחרוזת ההודעה למשתמש.

שלב 3 – נמצא את מיקומו של הסימן '-' (מינוס) במחרוזת הטלפון ,ונשמור את הערך הזה במשתנה position.

שלב 4 – נשמור במשתנה beforeMinus את תת-המחרוזת המציינת את אזור החיוג, ובמשתנה afterMinus את תת-המחרוזת המציינת את המספר עצמו. שימו לב כי אם הסימן '-' נמצא במקום 2, אזי הקידומת היא בעלת שתי ספרות ואורך המספר המצופה הוא שבע ספרות. אם הסמן '-' נמצא במקום 3, אזי הקידומת היא בעלת שלוש ספרות ואורך המספר המצופה הוא שמונה ספרות. שימוש בנתון הזה מאפשר שימוש בפעולה substr במקום בפעולה substring, אף שאין העדפה של פעולה אחת על השנייה.

שלב 5 – עתה נבדוק בעזרת פונקציית העזר inNum אם כל אחת מתת-המחרוזות מורכבת מספרות בלבד, ואם לא, נעדכן את מחרוזת ההודעה למשתמש.

שלב 6 – נבדוק אם מחרוזת ההודעה למשתמש ריקה. אם כן, אזי לא נמצאו שגיאות ועל כן נחזיר את הערך הבוליאני true. אבל אם המחרוזת אינה ריקה, אזי נמצאו שגיאות. נכתוב התרעה למשתמש הכוללת הודעות על השגיאות שנמצאו ונחזיר את הערך הבוליאני false.

```
function isTel()
{
   var strTel=document.getElementById("txtTel").value;
   var st="";
   var pos=strTel.indexOf("-");
```

```
    var beforeMinus=strTel.substring(0,pos);
    var len;
    if(pos==2)
       len=7;
    else
       len=8;
    var afterMinus=strTel.substr(pos+1,len);
    if(!isNum(beforeMinus))
      st+=" Before  minus must be only numbers -<br />";
    if(!isNum(afterMinus))
      st+=" After minus must be only numbers -<br />";
    if(st=="")
       return true;
    else
    {
       document.getElementById("phoneError").innerHTML = st;
       return false;
    }
}
```

הפונקציה isNum מקבלת פרמטר מהטיפוס מחרוזת ומחזירה ערך בוליאני true אם הערך שקיבלה הוא מספר, ואת הערך הבוליאני false בכל מקרה אחר:

```
function isNum(str)
{
    if(!isNaN(str))
       return true;
    else
       return false;
}
```

אפשר גם לכתוב זאת כך:

```
function isNum(str)
{
    return (!isNaN(str));
}
```

לפניכם קובץ aspx שבו מוצג טופס לקליטת הנתונים: שם משתמש, מספר תעודת זהות, טלפון, אימייל וסיסמה. כדי לבדוק את הנתונים הוספנו את אירוע המפעיל את בדיקות התקינות עם לחיצה על כפתור השליחה.

```
<input type="submit" value="שלח"  onclick="return isValid()/>
```

הפונקציות לבדיקת תקינות הנתונים מוצגות בחתימתן בלבד, ללא הקוד לביצוע.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
   <title>Validate Form</title>
   <script type="text/javascript">
   function isValidId()
   {
   }
   function isValidEmail()
   {
   }
   function isValidEmailOneAlert()
   {
   }
   function isDigitsAndLetters()
   {
   }
   function isUpperAndLowerCase()
   {
   }
   function isNum(str)
   {
   }
   function isTel()
```

```
    {
    }
    function isValid()
    {
      return (isValidId()&&
              isValidEmail()&&
              isValidEmailOneAlert()&&
              isDigitsAndLetters() &&
              isUpperAndLowerCase() &&
              isTel()                    );
    }
    </script>
</head>
<body dir="rtl">
    <form id="form1" runat="server">
    <div>
        <table>
            <tr>
                <th align="right"> :שם משתמש </th>
                <td><input type="text" id="txtUserName"/></td>
            </tr>
            <tr>
                <th align="right"> :.מספר זהות </th>
                <td><input type="text" id="txtId"/></td>
            </tr>
            <tr>
                <th align="right">:סיסמה </th>
                <td><input type="password" id="pwd" /></td>
            </tr>
            <tr>
                <th align="right">:אימות סיסמה</th>
                <td><input type="password" id="pwdConfirm"/></td>
            </tr>
            <tr>
```

```
            <th align="right">:דוא"ל</th>
            <td><input type="text" id="txtEmail"/></td>
        </tr>
        <tr>
            <th align="right">:טלפון</th>
            <td><input type="text" id="txtTel"/></td>
        </tr>
        <tr>
            <td colspan="2"><input type="submit" value="שלח"  onclick="return isValid()/> </td>
            </tr>
</table>
  </div>
   </form>
</body>
</html>
```

שאלה 5.24

כתוב פעולה הבודקת אם אורכה של הסיסמה המבוקשת לפחות שישה תווים ואם הערכים שהוקלדו בשדה 'סיסמה' ובשדה 'אימות סיסמה' הם זהים.

שאלה 5.25

כתוב פעולה המודיעה למשתמש מה חוזק סיסמתו.

חוזק הסיסמה נקבע כך :

רמה 1 – הסיסמה היא אחת מאוסף ידוע של סיסמאות, אשר לרוב מורכב מרצף ספרות או מרצף אותיות עוקבות במקלדת (כגון : 123456, password, qwer, asdfg ועוד), או זהה לשם המשתמש.

בדקו לפחות שש סדרות מוכרות.

רמה 2 – הסיסמה איננה אחת מאוסף הסיסמאות הידוע ואורכה גדול משש.

רמה 3 – נוסף על כך, הסיסמה מכילה ספרות ואותיות.

שאלה 5.26

הרחיבו את הפעולה הבודקת אם מספר הטלפון תקין.

מספר טלפון הוא תקין אם:

1. התו הראשון הוא 0.
2. התו השני אינו 1.
3. הוא מורכב מספרות בלבד, פרט לסימן '–' המפריד בין הקידומת למספר.
4. אם הספרה השנייה היא 5, אזי הקידומת בעלת שלוש ספרות והמספר עצמו בן שמונה ספרות, אחרת - הקידומת בעלת שתי ספרות והמספר עצמו בן שבע ספרות.
5. המספר עצמו (ללא הקידומת) אינו מתחיל ב-0.

שאלה 5.27

הרחיבו את הפעולה הבודקת אם כתובת הדוא"ל תקינה.

כתובת דוא"ל תקינה אם:

1. היא מכילה את התו '@'.
2. התו @ אינו הראשון במחרוזת.
3. היא אינה מכילה יותר מאשר מופע אחד של התו '@'.
4. היא מכילה את התו '.' (נקודה) לפחות פעם אחת, אחרי התו '@'. (אין מניעה שהתו '.' יופיע גם לפני התו @).
5. התו '.' (נקודה) אינו האחרון במחרוזת.

שאלה 5.28

כתוב פעולה הבודקת אם שֵם המשתמש מורכב מאותיות בלבד ואינו מכיל את התווים:

';' (נקודה-פסיק ;), '>' (גדול מ-), '<' (קטן מ-).

שאלה 5.29

כתוב פעולה הבודקת אם שם המשתמש מורכב מאותיות עבריות או אנגליות בלבד.

הנחיות לביצוע: פעולה זו תשתמש בשתי פונקציות:

- פונקציה אחת לבדיקה ששם המשתמש מורכב רק מאותיות עבריות. **רמז:** ניתן להשוות גדלים של אותיות, שכן האותיות בעברית מסודרות בסדר האל"ף-בי"ת, וכך האות 'ג' קטנה מהאות 'כ'.

- הפונקציה השנייה תבדוק ששם המשתמש מורכב מאותיות אנגליות בלבד. **רמז:** ניתן להמיר את המחרוזת ל-LowerCase בלבד ולבדוק שכל התווים הם בין 'a' ל- 'z'.

5.8 ייבוא קבצי JavaScript

כתיבת תסריטי JavaScript באזור תגי ה- head גורם, לעיתים קרובות, להגדלת גודל דף. כדי להקטין את גודל הדף, ובכך, להקטין את זמן טעינתו, להקטין את מספר השגיאות ולפתח דף שבו אנו מפרידים בין התוכן המוצג (באמצעות תגי HTML) לבין הפעולות שיש לבצע (תסריט JavaScript), ניתן לכתוב דף HTML המייבא קובץ javascript חיצוני (באופן דומה לדרך שבה קשרנו קובץ CSS להגדרת התצוגה). כדי לייבא קובץ המכיל תסריטי Javascript לקוד HTML נבצע את הפעולות הבאות:

א. ניצור קובץ המכיל תסריט JavaScript עם הסיומת js. לדוגמה נשמור את הקובץ הבא בקובץ בשם myJs.js

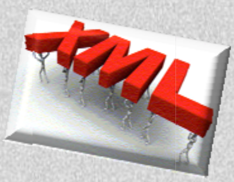
```
function popup()
{
	alert("Hello World")
}
```

ב. כעת בדף ה HTML נוסיף תסריט המשתמש במאפיין "scr" המציין את קובץ המקור (הכולל תסריט JavaScript) בו נשתמש בדף זה.

```
<html>
<head>
	<script  language="JavaScript" src="myjs.js"> </script>
</head>
<body>
	<input type="button" onclick="popup()" value="Click Me!">
</body>
</html>
```

5.9 תרגיל משימה נלווה

הוסף בדיקות תקינות לטופס הרישום לאתר המשחקים.



פרק 6

עקרונות שפת XML

6.1 מבוא

הוספת סימנים למסמכים שונים מטרתה לספק מידע נוסף על התוכן. לדוגמה, הוספת סימני פיסוק במסמך משפר את הקריאות. באופן דומה מורים בבית-הספר נוהגים לסמן את העבודות של התלמידים: הם מסמנים משפטים שגויים, מוסיפים פירוט או סימני פיסוק, מדגישים פסקאות וכדומה. בתחום המחשוב, 'סימון' של טקסט מתבצע על ידי שימוש בקודים הקרויים תגים (או לעתים תוויות) שמאפשרים להגדיר את מבנה הנתונים, את מראם ולעתים גם את משמעותם. לדוגמה, שפת HTML, שהכרתם בפרק 2, היא שפה של סימנים המתייחסים בעיקר למבנה החזותי של המידע שהמסמך מכיל. לעומת זאת, שפת הסימנים XML, שאותה נציג בפרק הזה, מטרתה להגדיר את מבני הנתונים.

מסמכים בשפות HTML ו-XML דומים זה לזה בכך שהם מכילים נתונים המסומנים בתגים. אבל בזאת מסתיים הדמיון ביניהם. ב-HTML, התגים מגדירים את המראה של הנתונים – הם מגדירים את מיקומן של הכותרות ושל הפסקאות וכדומה. בשפת XML התגים מגדירים את המבנה ואת המשמעות של הנתונים. כאשר אתם מתארים את המבנה ואת המשמעות של הנתונים שלכם, אתם מאפשרים שימוש חוזר באותם הנתונים בדרכים ולמטרות שונות. לדוגמה, את נתוני המכירות שנשמרים במסד נתונים של מחלקת השיווק ניתן להעביר להנהלת החשבונות כדי לטפל בתשלומים הקשורים במכירות. במילים אחרות, באפשרותכם להשתמש במערכת אחת כדי ליצור את הנתונים שלכם ולסמנם אותם בתגי XML, ולאחר-מכן לעבדם במגוון מערכות אחרות, ללא קשר לחומרה, למסד הנתונים או למערכת ההפעלה. האפשרות להעבירם ממסד נתונים אחד לאחר היא אחת הסיבות לכך ששפת XML הפכה לאחת השיטות הפופולאריות ביותר להעברת נתונים.

כדי להבין טוב יותר את שפת ה-XML ולעמוד על יכולותיה, הבה נבחן את התרחיש שלפניכם:

בחברה להוצאה לאור רוצים לשווק את הספרים שלהם בכל העולם. השיווק נעשה בעיקר דרך האינטרנט. כל בעל אתר יכול לפרסם את רשימת הספרים באתרו, והוא יקבל אחוז מסוים ממחיר הספר אם הקנייה נערכה דרך האתר שלו. בהתחלה חשבו בחברה לשלוח העתק מהמסד שמכיל את נתוני הספרים לאתרי האינטרנט שמעוניינים לשווק את הספרים. ואולם, עד מהרה התברר שהרעיון הזה אינו מוצלח, מפני שאתרים רבים משתמשים במסדי נתונים שונים ובמערכות הפעלה שונות, ולכן הם אינם יכולים לעבד נתונים שמעוצבים בתבנית של מסד הנתונים של החברה. כפתרון חלופי, הוצע לשלוח קובץ טקסט שמכיל את רשימת הספרים. ואולם, גם הפתרון הזה לא היה מוצלח משום ששימוש בקובץ הטקסט מונע מבעלי האתרים את האפשרויות המתקדמות של מסדי הנתונים, כמו עדכון, חיפוש, מיון וכו'.

כפי שניחשתם, אחד הפתרונות לבעיות שהצגנו הוא השימוש בשפת XML. שליחת הנתונים בתבנית XML יוצרת הפרדה בין התבנית שבה נשמר המידע בטבלאות שבבסיסי הנתונים שבשרת לבין התבנית שבה מוצג המידע אצל הלקוח. בשיטה הזאת, הלקוח משתמש בקובץ XML כדי להמיר את הנתונים למבנה המתאים לו, וכך הוא יכול לעבד את הנתונים הללו מבלי לשנות את היישום שלו (מלבד כמובן בניית המודול שמטפל בהמרת נתונים מקובץ XML). כיום, נהוג להשתמש בתקן XML גם להעברת נתונים בין תוכנות שונות אשר לאו יועדו מראש לעבוד בתיאום.

אז מהי בעצם שפת XML ?

XML (**Extensible Markup Language**), או בעברית **'שפת סימון הניתנת להרחבה'**, היא שפה לתיאור ולהגדרה של נתונים, המבוססת על תקן שנקבע על-ידי הארגון הבינלאומי W3C (**World Wide Web Consortium**) העוסק בתקינה באינטרנט. התגים ב-XML אינם קבועים או מוגדרים מראש; אנו מגדירים ומרחיבים אותם לפי צרכינו (מכאן שמה – Extensible), ופעולות אלה מאפשרות לנו גמישות בטיפול בנתונים. כדי להבין זאת, נתבונן בדף XML הזה:

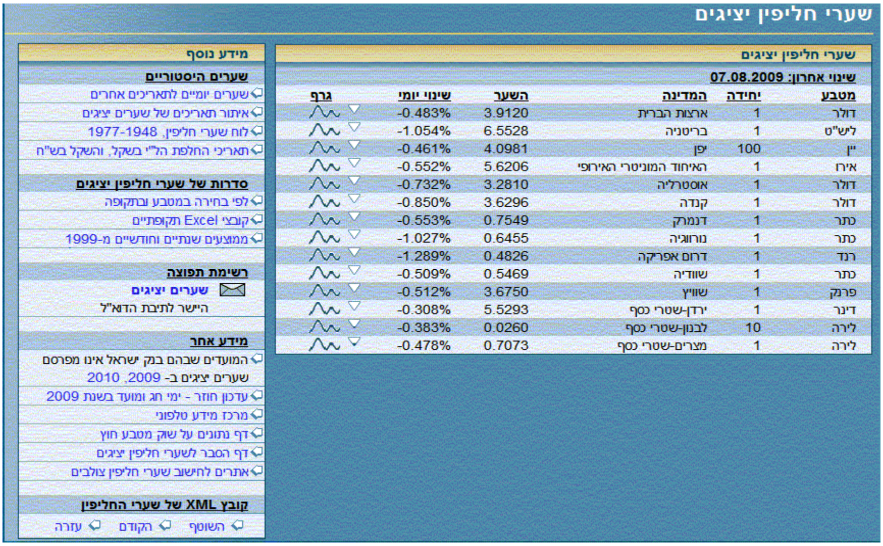
```
<student>
   <id> 02233465</id >
   <firstName> Amir </firstName>
   <lastName>  Nagar </lastName>
</student>
```

בדף הזה אנחנו מאחסנים מידע על תלמיד ששמו 'Amir Nagar' ושמספר הזהות שלו הוא '02233465'. שימו לב לכך שהקובץ קריא, ואף ללא ידע מוקדם ב-XML קל להבחין כי הקובץ מכיל תגים שמסומנים ב- < > הדומים לתגים בשפת HTML. כמו כן, אפשר לראות כי לכל תג פותח יש תג סוגר. הנתונים עצמם נמצאים בין התגים הפותחים לתגים הסוגרים. (על התחביר ועל חוקי שפת XML נרחיב בסעיף 6.2). את הנתונים הללו ניתן להעביר לכל יישום שיכלול מרכיב תוכנה אשר יידע לפענח אותם ולהשתמש בהם. טכניקה זו דומה ליצירת דף HTML שתוכנת הדפדפן יודעת לקרוא ולהציג.

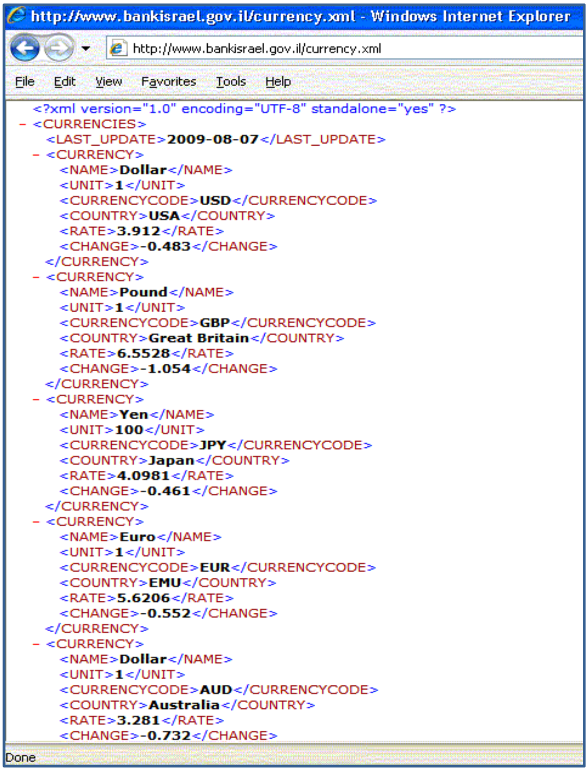
אתרים רבים משתמשים בשפת XML להצגת הנתונים. לדוגמה, נתבונן בשערים היציגים של מטבע החוץ שבנק ישראל מפרסם יום יום באתר שלו. גְלשו לכתובת הזאת:

http://www.bankisrael.gov.il/heb.shearim/

האתר מציג את השערים היציגים היומיים והוא מאפשר גם להציג את המידע בשפת XML. בתחתית הדף, בצד שמאל, לחצו על הקישור 'קובץ XML של שערי החליפין' ובחרו באפשרות 'השוטף' (איור 6-1).



א



ב

איור 6-1

א. שערי החליפין היציגים באתר של בנק ישראל
ב. קובץ XML של שערי החליפין היציגים

שאלה 6.1

הציגו שני תרחישים להדגמת השימושים האפשריים של קובץ ה-XML של שערי החליפין היציגים.

בפרק הזה מטרתנו העיקרית היא להבין כיצד אפשר להציג את הנתונים השמורים בתבנית XML. ייצוג XML משמש כייצוג מתווך בין המבנה של הנתונים שבו משתמשת תוכנת הלקוח לבין המבנה של הנתונים שבו משתמשת תוכנת השרת. לדוגמה, תוכנת השרת משתמשת במסד הנתונים SQLServer, ואילו תוכנת השרת משתמשת במסד הנתונים MySQL.

אפשר גם להשתמש בנתונים שמיוצגים בתקן XML בשפות תכנות שונות, כגון C#, JAVA, Delphi, Visual Basic, או בשפות התסריט, לדוגמה JavaScript וכדומה. למשל, ניתן לקרוא קובץ XML בכל אחת מהשפות ללא קשר לתוכנו של הקובץ. כמו כן, אפשר להשתמש בטכנולוגיות שונות, כמו ASP, ASP.NET, JSP וכדומה. שפת XML היא גם חלק בלתי נפרד מתכניות הגרפיקה, כגון FLASH ו-PHOTOSHOP, ומתכניות האופיס, כגון WORD, ACCESS, EXCEL. לדוגמה, אם נכתוב מסמך באחת מתכניות האופיס, נוכל לשמור אותו כמסמך XML ולהעבירו מיישום ליישום. כמו כן, היא עובדת עם רוב מערכות ההפעלה כמו WINDOWS, LINUX, UNIX וכן בטלפונים הניידים.

יתרונות העבודה עם XML:

1. קובצי XML מאפשרים החלפת מידע בין מערכות: במקרים רבים מערכות ההפעלה ובסיסי הנתונים אינם תואמים זה את זה. קובץ XML הוא קובץ קריא בכל מערכות ההפעלה המודרניות, והוא מאפשר תקשורת פשוטה עם כל היישומים לניהול מסדי נתונים הנפוצים בשוק.
2. קובצי XML מאפשרים גמישות בגישה למידע: כיוון שייצוג בשפת XML אינו תלוי בחומרה או בתוכנה, המידע המאוחסן באמצעותה אינו מוגבל לסוגים מסוימים של דפדפנים ברשת האינטרנט.
3. הבחנה בין המידע העיצוב לבין העיצוב: המידע מופרד מממשק המשתמש, ולכן שינויים בתצוגה אינם מצריכים שינוי במבנה הנתונים. תכונה זו מבדילה שפה זו משפת HTML שבה המידע והעיצוב נמצאים באותו המסמך.

4. המידע קל לעיבוד: המבנה של קובצי XML מאפשר ליישומים שונים, כגון מנועי החיפוש, לפענח את הנתונים בצורה יעילה ופשוטה.
5. הקוד ב-XML הוא ברור וקריא.
6. תוכנות 'חומת האש' (firewall) יכולות לקרוא ולוודא שהנתונים ששלח המשתמש אינם מכילים וירוסים או רכיבים אחרים אשר עלולים להזיק למחשב שמקבל את הנתונים.
7. השפה חופשית לשימוש. אין צורך בהרשאה לשימוש והיא אינה תלויה בספק התוכנה.

שפת XML לעומת שפת HTML

ההבדל העיקרי בין שפת XML ושפת HTML הוא שב-HTML הדגש הוא על אופן הצגת המידע וב-XML הדגש הוא על אופן הייצוג של המידע ומבנהו. חשוב להבין ששפת XML לא נועדה כדי להחליף את שפת ה-HTML, וכל אחת מהן פותחה לצרכים שונים.

נציג תחילה פעילות שתדגים את ההבדל בין דף HTML לדף XML.

1. פתחו את פנקס הרשימות.
2. כתבו את הקוד הזה:

```
<body>
  <h3>first line</h3>
  <b>bold</b>
  <u>underline</u>
</body>
```

3. שִׁמרו את הקובץ בשם a.html (זכרו איפה שמרתם אותו).
4. הקליקו הקלקה כפולה על הקובץ שנוצר, ורשמו מהו הפלט שהתקבל.
5. כעת, בצעו את שלבים 1 – 4 פעם נוספת, אבל בשלב 3 שמרו את הקובץ בשם a.xml.
6. הקליקו הקלקה כפולה על הקובץ שנוצר ורִשמו מהו הפלט שהתקבל.
7. רשמו את ההבדלים בין הפלטים של שני הקבצים.

להלן הפלטים של כל אחד מן הקבצים:

הפלט של הקובץ a.html:

first line

bold <u>underline</u>

הפלט של הקובץ a. xml:

```
<body>
      <h3>first line</h3>
      <b>bold</b>
      <u>underline</u>
 </body>
```

שאלה למחשבה

כיצד יודע הדפדפן איך להציג את הנתונים של קובץ ה-HTML בצורה הזאת?

התגים של HTML מוגדרים בדפדפן מראש, והוא יודע להציג את הנתונים לפי הגדרותיו. בדוגמה שלנו, הדפדפן הציג את המידע שבקובץ ה-HTML בהתאם להגדרות של תגי ה-HTML, כלומר: עם כותרת, עם הדגשה ועם קו תחתון. לעומת זאת, הדפדפן בקובץ ה-XML לא ידע לפענח את התגים, ולכן הוא התעלם מאופן הצגת הנתונים והציג את הקובץ כפי שהוא.

מחבר קובץ ה-HTML יכול להשתמש רק בתגים שמוגדרים מראש, למשל טבלאות, תמונות, קישורים וכדומה (מספר התגים ב-HTML קבוע), ורק במבנה שהוגדר מראש. לעומת זאת, מחבר קובץ ה-XML יכול לייצר את התגים שהוא צריך בלי הגבלה (מספר התגים אינו קבוע ואינו מוגדר מראש). באופן הזה, בקובץ XML ניתנת לו האפשרות לתאר את כל סוגי הנתונים שלא היה אפשר לתארם בשפת HTML, כגון נוסחאות מתמטיות.

6.2 המבנה של מסמך ה-XML

התחביר והחוקים של שפת XML

בדומה לשפות סימון אחרות למסמך XML יש מבנה ותחביר מוגדר מראש. מסמך XML מורכב מ**אלמנטים** (elements) מ**מאפיינים** (attributes) ו**מנתונים** (data). מסמך XML מכיל לפחות אלמנט אחד – שנקרא **שורש** (root) – אשר בתוכו יקננו אלמנטים אחרים. את האלמנט מייצגים תג התחלה ותג סיום וביניהם מצוין התוכן. התוכן יכול להכיל נתונים, אלמנטים מקוננים או שניהם גם יחד. נוסף על כך, אלמנט יכול להכיל גם מאפיינים שמספקים מידע נוסף. בניגוד לאלמנטים, מאפיינים אינם יכולים להכיל בתוכם אלמנטים או מאפיינים נוספים. לדוגמה, נתבונן בהגדרות XML שלהלן:

```
<customer name="Gabriel Okara">
  <occupation>Poet</occupation>
</customer>
```

ההגדרה מכילה שני אלמנטים: האלמנט הראשון מתאר לקוח, והוא תחום על-ידי התגים <customer> ו- </customer>. לכל תג פותח חייב להיות תג סוגר שמתחיל בתו '/'. האלמנט הזה מכיל מאפיין המתאר את שם הלקוח name="Gabriel Okara" וכן אלמנט נוסף שמקונן בתוכו. שימו לב, האלמנט המקונן נמצא בין התג הפותח לתג הסוגר של האלמנט המתאר את הלקוח. האלמנט השני מתאר את משלוח היד של הלקוח, והוא תחום על-ידי התגים <occupation> ו- </occupation>.

מאפיין מוגדר בתוך התג הפותח של אלמנט ודומה במבנהו למאפיין המוגדר במסמך HTML:

ערך = שם מאפיין

בדוגמה שלנו המאפיין name="Gabriel Okara" נמצא בתוך התג הפותח < customer>. שימו לב כי יש לסמן במרכאות את הערך של המאפיין.

דוגמה נוספת, כדי לתאר שם של ספר הכתוב באנגלית, נגדיר את האלמנט title המייצג שם של סרט ואת המאפיין language שמתאר את השפה אנגלית "en":

<title language="en">Gone with the wind</title>

ולמעשה לא קיימים כללים חד-משמעיים מתי צריך להשתמש באלמנט ומתי צריך להשתמש במאפיין. ההחלטה במה להשתמש תתקבל בעת עיצוב הנתונים. באופן כללי, רצוי להגדיר מידע חיוני שיהיה דרוש בהמשך לעיבוד ולתיאור הנתונים כאלמנט.

כאמור, מסמך XML צריך להכיל אלמנט אחד שהוא השורש המשמש ההורה של כל האלמנטים האחרים. למעשה, ניתן לתאר את הקשרים בין האלמנטים במבנה של עץ. בדוגמה המתארת פרטים של לקוח, השורש הוא האלמנט <customer> שבתוכו מוכל האלמנט <occupation> שהוא בנו.

חוקי XML הם פשוטים וקלים לקריאה ולשימוש. עם זאת, החוקים האלה נוקשים מאוד, וטעות קטנה אחת תמנע את הצגתו של המסמך.

להלן הכללים הבסיסיים לכתיבה של מסמך XML תקין:

א. כל תג במסמך צריך להיסגר; אסור להוסיף תג שאין לו תג סוגר.

ב. התגים של XML צריכים להיות מקוננים היטב, כלומר – מה שנפתח ראשון נסגר אחרון ומה שנפתח אחרון נסגר ראשון. להלן דוגמה לקינון כהלכה:

<u> Well-formed </u>

ודוגמה לקינון שלא כהלכה:

<u> not Well-formed </u>

ג. מסמךXML חייב להכיל לפחות אלמנט אחד (כולל השורש). במסמך חייב להיות אלמנט שנפתח בהתחלה ונסגר בסוף בלי שיופיע יותר מפעם אחת (שורש – root). השורש הוא אלמנט החובה היחיד.

ד. שפת הסימנים XML רגישה לגודל האות (case sensitive). את תגי הפתיחה והסגירה צריכים לכתוב בדיוק באותן אותיות – קטנות או גדולות. אסור למשל לסגור את התג <student> בצורה הבאה </STUDENT> או </Student>

ה. את ערכי המאפיינים (attribute) במסמך יש לסמן במרכאות. אם נכתוב ערכים ללא מרכאות, הם לא יוצגו במסמך ה-XML.

ו. הרווחים בתוכן של מסמך ה-XML נשמרים במחרוזת. לדוגמה, אם נרשום שתי מילים וביניהן שלושה רווחים, מספר הרווחים בין שתי המילים יישמר. לעומת זאת, שפת HTML מוחקת את הרווחים ומשאירה רק רווח אחד. לדוגמה, המשפט

```
space     example
```

יוצג בשני אופנים:

בקובץ XML כך:

```
space     example
```

ובקובץ HTML כך:

```
space example
```

ז. לא ניתן ליצור רווחים בתוך התגים. שימו לב, בניגוד לסעיף הקודם, כאן אנו מתייחסים לרווחים שבתוך התגים ולא לרווחים שהם התוכן בין התגים. אסור למשל להוסיף לאלמנט <name> רווחים כמו <name   > אף שהצבנו אותו מספר של רווחים בתג הסוגר.

ח. ניתן לכתוב הערות באופן דומה לזה של כתיבת הערות בדף ה-HTML:

```
<!-- comment -->
```

דוגמה למסמך XML

נתאר לדוגמה מסמך XML פשוט שבו נציג את הנתונים של טבלה בשם students, המכילה את פרטי הסטודנטים שלומדים קורס באוניברסיטה. לכל סטודנט יש מספר מזהה, שם פרטי, שם משפחה, ציון סופי בקורס ושם הקורס. להלן קטע מהטבלה המכילה פרטים של שלושה סטודנטים:

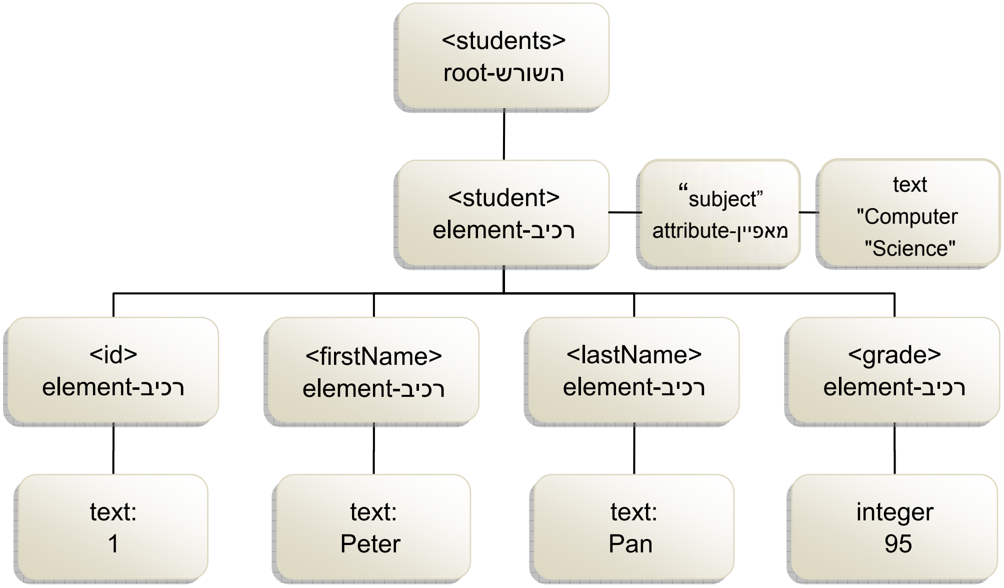
id	firstName	lastName	grade	Subject
1	Peter	Pan	95	Computer Science
2	Ali	Baba	90	Economy
3	Mickey	Mouse	85	Physics

את הטבלה ניתן לתאר בצורת עץ. אלמנט השורש מסומן בתג students (שהוא שם הטבלה) והוא מציין שאלה הנתונים של הסטודנטים. שאר האלמנטים הם מקוננים בתוך השורש,

והם מסומנים בתג student. כל אלמנט כזה מתאר פרטים של סטודנט אחד, והוא מכיל את תת-האלמנטים האלה: id, firstName, lastName, grade, subject. בדוגמה זו, את העמודה subject שבטבלה אנו מסמנים כמאפיין של האלמנט student.

כפי שציינו, האלמנט student יכול להופיע כמה פעמים מתחת לשורש; השורש עצמו מופיע רק פעם אחת במסמך.

בדוגמה זו נציג את כל הנתונים של סטודנט אחד בעץ המתאר את נתוני הסטודנטים, ולאחר-מכן נציג כיצד העץ ניתן לייצוג בקובץ XML.



איור 6-2
עץ המתאר את פרטי הסטודנטים

להלן קובץ XML המתאר את נתוני הסטודנטים. מְספרנו את מקצת השורות כהערות רק לצורך ההסבר.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  <!--1 שורה-->
<students>
<student subject ="computer science">          <!--3 שורה-->
        <id>1 </id>
        <firstName> Peter </firstName>
```

```
      <lastName> Pan </lastName>
      < average > 95  </ average >
  </student>                              <!--שורה 8-->
<student subject ="Economy">              <!--שורה 9-->
  <id>2 </id>
  <firstName> Ali </firstName>
  <lastName> Baba </lastName>
  <average> 90 </ average >
  </student>                              <!-- שורה 14-->
<student subject =" Physics">             <!--שורה 15-->
  <id>3 </id>
  <firstName> Mickey </firstName>
  <lastName> Mouse </lastName>
  < average > 85 </ average >
  </student>                              <!--שורה 20-->
</students>
```

נסביר בקצרה כיצד בנוי קובץ ה-XML:

- השורה הראשונה:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

מסמך XML מתחיל בהצהרה על תחילתו של המסמך (XML Declaration). ההצהרה מסומנת בסימני שאלה ויכולה להכיל כמה מאפיינים. המאפיין version מתאר את מספר הגרסה, והמאפיין encoding מתאר את הקידוד. בדוגמה שלנו, הצבנו למאפיין encoding את הערך ISO-8859-1 שמגדיר את האותיות הלטיניות. אף-על-פי שאין חובה להצהיר על מסמך XML, כדאי להקפיד לעשות זאת כדי להימנע מבעיות מיותרות בעתיד כאשר נבצע פעולות מתקדמות יותר.

- השורה השנייה:

```
<students>
```

לאחר שהגדרנו את השורה הראשונה, אנחנו צריכים להגדיר את אלמנט השורש. אלמנט זה מתאר בדרך-כלל את תוכנו של המסמך ועשוי לכלול בתוכו אלמנטים נוספים.

- השורות 3,9,15 :

כדי להציג מידע, נצטרך לבנות אלמנטים נוספים שמכילים את המידע. כל אלמנט כזה נקרא 'הבן של השורש'. בדוגמה שלנו לשורש יש שלושה בנים, וכל בן מקבל את השם student ומכיל בתוכו ארבעה אלמנטים (id, firstName, lastName, average) שמכילים את הנתונים של כל student. נוסף על כך, הוספנו מאפיין שמתאר את מקצוע הלימוד – subject.

- השורות 8,14, 20 :

החוקים הנוקשים של XML מחייבים אותנו לסגור כל תג שאנחנו פותחים. וכאן אנחנו סוגרים את תגי ה-student שהם למעשה הבנים של השורש.

- השורה האחרונה :

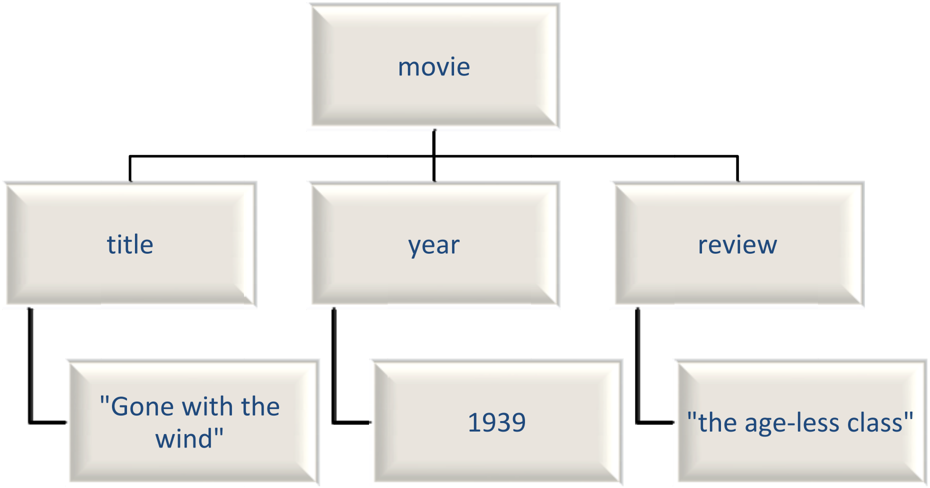
בשורה האחרונה אנו סוגרים את תג השורש ומסיימים את המסמך.

שאלה 6.2

א. הוסיפו לעץ באיור 6-3 ענף המתאר את הסטודנט השני (Ali Baba).

ב. שנו את ההגדרות כך שהמאפיין "subject" יוגדר כאלמנט של האלמנט students.

ג. כתבו קובץ XML לתיאור העץ הזה :



איור 6-3

עץ המתאר פרטי סרט

שאלה 6.3

בנו מסמך XML שמכיל נתונים על ארבעה ספרים. לכל ספר יש לכלול את הנתונים האלה: שם הספר, מספר הדפים והמחבר. על המסמך להיות בנוי לפי תחביר וחוקי XML המקובלים.

שאלה 6.4

האם קובץ ה-XML שלפניכם הוא תקין מבחינה תחבירית? נמקו את תשובתכם.

```
<student>
     <id>1 </id>
    <Name> אבי </Name>
    <avg> 90  </avg>
 </student>
<student>
   <id>2 </id>
     <Name>מוטי </Name>
```

```
        <avg> 80 </avg>
  </student>
```

מרחב השמות (XML Namespaces)

הגדרות XML יכולות להיכתב בכמה קובצי XML, וכל קובץ יכול לתאר למשל טבלה אחת. שימוש בכמה קובצי XML יכול לגרום התנגשות בין שמות. התנגשות בין שמות מתרחשת כאשר משתמשים באותם התגים בכמה מסמכי XML שונים, אבל בכל מסמך התגים מתארים אלמנטים שונים. לדוגמה, נניח שהוגדרו שני מסמכי XML: מסמך ה-XML הראשון מכיל את פרטי טבלת ה-HTML:

```
<table>
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
</table>
```

ומסמך ה-XML השני מכיל מידע על שולחן (table):

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

אם נשתמש בשני מסמכי ה-XML הללו באותו יישום, ייתכן בלבול משום ששניהם מכילים אלמנט בשם table (כל אחד מאפיין ישות שונה). כדי להימנע מהתנגשויות מסוג זה, ניתן להוסיף תחילית המציינת את סוג האלמנט, אחריו את התו ":" ובסיום את שם האלמנט. לדוגמה:

```
<h:table>
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
```

```
</h:table>
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

בדוגמה זו התג <h:table> נועד כדי לתאר אלמנט שהוא טבלת HTML (האות h מסמלת HTML) והתג <f:table> נועד כדי לסמן את האלמנט שולחן (האות f היא קיצור של המילה furniture – רהיט).

שאלה 6.5

חברה המוכרת ספרים ותקליטורים באמצעות האינטרנט מציגה את פרטי המוצרים גם בקובצי XML. להלן קטעים מתוך שני קובצי XML שניתן להוריד מקובץ האתר. קובץ cd.xml מתאר פרטי תקליטורים והקובץ השני book.xml מתאר פרטי ספרים.

```
<!--cd.xml -->
<CD-CATALOG>
<CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <COUNTRY>USA</COUNTRY>
   <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
 </CD>
<CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <COUNTRY>UK</COUNTRY>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
  </CD>
</CD-CATALOG>
```

```
<!--book.xml -->
<BOOK-CATALOG>
<BOOK>
 <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <PRICE>10.90</PRICE>
  <YEAR>1985</YEAR>
  </ BOOK >

  < BOOK >
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <PRICE>9.90</PRICE>
  <YEAR>1988</YEAR>
  </ BOOK >
</BOOK-CATALOG>
```

א. ציירו את עץ התקליטורים.
ב. הציגו את נתוני הספרים בטבלה.
ג. תארו את הבעיה שבה ייתקלו ספקים אשר רוצים להשתמש בשני הקבצים הללו. הציעו פתרון לבעיה זו.

6.3 סכמת XML (XSD)

מסמך XML אשר עונה על כללי התחביר שציינו, כגון התאמת תג סיום לכל תג פתיחה, קינון כהלכה ומתן שמות נכונים לאלמנטים, הוא מסמך שמעוצב כהלכה[1]. אבל מסמך שמעוצב כהלכה אינו תנאי מספיק לייצוא וייבוא של נתונים בין מערכות לבין יישומים שונים. כפי שציינו קודם, כאשר עובדים עם מערכות שונות, אחת הבעיות העיקריות היא חוסר התאימות של הנתונים. חוסר התאימות עלול לגרום לאי-התאמה בין ההגדרות של מערכת אחת לאחרת.

נמחיש את הבעיה בעזרת הדוגמה הזאת: מארגני האולימפיאדה רוצים לבנות מסד נתונים של ספורטאים. כל מדינה שמשתתפת באולימפיאדה מתבקשת לשלוח את פרטי הספורטאים שלה על-פי התבנית של קובץ XML שהגדירו מארגני האולימפיאדה. למשל, בתבנית שלפניכם מוגדרים השורש של המסמך ופרטי המדינה והאתלטים המייצגים מדינה זו:

```
<games>
 <country id=" ... ">
  <athlete>
   <id>  ...  </id>
   <name> ...   </name>
   <sport>  ... </sport>
   <age>   ... </age>
   <gender> ...  </gender>
  </athlete>
```

[1] קישור לכלי לבדיקת מסמך שמעוצב כהלכה:
http://www.w3schools.com/xml/xml_validator.asp

```
  </country>
</games>
```

לאחר שמארגני האולימפיאדה קיבלו את קובצי ה-XML מהמדינות השונות, הם ניסו להכניס את הנתונים שהתקבלו לטבלה במסד נתונים שהם בנו לצורך העניין. ואולם הם קיבלו הודעות שגיאה רבות. חִשבו, מה גרם להודעות שגיאה אלה?

להלן שתי דוגמאות של קבצים שהתקבלו משתי מדינות שונות.

קובץ א׳

```
<games>
  <country id="Italy">
    <athlete>
      <id>123-4</id>
      <name> Toni</name>
      <sport>swimming</sport>
      <age>20</age>
      <gender>male</gender>
    </athlete>
</games>
```

קובץ ב׳:

```
 <games>
 <country id="Israel ">
    <athlete>
      <id>43</id>
      <name> Israeli Ben</name>
      <sport>football</sport>
      <age>17</age>
      <gender> 0 </gender>
    </athlete>
  </country>
</games>
```

שאלה 6.6

א. יצגו את המידע ששלחו המדינות בשתי טבלאות.

ב. כמה שחקנים שלחה כל מדינה?

לאחר שהמארגנים בדקו את מסמכי ה-XML שהתקבלו, התברר, לדוגמה, שהערכים שהתקבלו עבור האלמנט 'מין' של הספורטאי לא התאימו לטיפוס השדה 'מין' שבטבלת הספורטאים. חלק מהמדינות רשמו את הערכים "male" ו-"female" וחלק מהמדינות רשמו את הערכים true ו-false ואחרות ציינו 0 או 1. כמו כן, חלק מהמדינות רשמו ערכים טקסטואליים לאלמנט id ובמסד הנתונים הוא היה מוגדר כשדה מסוג מספר. מתברר כי כדי לייבא את הנתונים ממערכת אחרת, היה צורך לא רק לתאר את התגים של האלמנטים בקובץ XML, אלא גם להוסיף מידע על טיפוסי הנתונים, ערכים אפשריים ועוד.

כדי לפתור בעיה זו אפשר להשתמש בסכמת XML הנקראת **XSD** (**XML Schema Definition**) ובעברית "**הגדרת סכמת XML**". XSD פותח כדי לתאר את המבנה של קובץ XML.

למעשה, אפשר לראות בקובץ XSD קובץ שאפשר להצמידו למסמך XML כדי לוודא שהוא מעוצב היטב וגם תקף מבחינה תחבירית. השימוש ב-XSD דומה להגדרה של סכמה של טבלת נתונים במסד נתונים טבלאי. סכמת XML מגדירה באילו אלמנטים ומאפיינים אפשר להשתמש בקובץ XML, את מספר האלמנטים הבנים ואת סדר הופעתם, את הטיפוסים של האלמנטים ואת ערכי ברירת המחדל, ערכים קבועים וכדומה.

הגדרת סכמת XML

באופן כללי, סכמת XML מגדירה את האלמנטים והמאפיינים האלה:

- האלמנטים אשר עשויים להופיע במסמך XML
- המאפיינים אשר עשויים להופיע במסמך
- האלמנטים שהם אלמנטי הבנים (הצאצאים)
- סדר ההופעה של האלמנטים הבנים
- מספר האלמנטים הבנים
- אם אלמנט הוא ריק או אם הוא כולל טקסט

- טיפוסי הנתונים לאלמנט ולמאפיינים
- ערכי ברירת המחדל וערכים קבועים לאלמנטים ולמאפיינים

נתאר בקצרה את האלמנטים של מסמך XSD וכיצד להגדיר סכמה לקובץ XML שתיארנו קודם לכן בדוגמה של האולימפיאדה.

א. האלמנט <schema> הוא אלמנט השורש בכל סכמת XML :

```
<?xml version="1.0"?>
<xs:schema>
...
...
</xs:schema>
```

אלמנט סכמה עשוי לכלול מספר מאפיינים, לדוגמה :

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
        targetNamespace= "http://www.olympicGames"
        xmlns= "http://www.olympicGames"
        elementFormDefault="qualified">
```

נסביר להלן כל מאפיין.

המאפיין הראשון xmlns יכול להימצא בכל קובצי ה-XML, והוא מציין שהמרחב שמות (namespace) השייך אליו חוקי וזמין במסמך. כל קובצי ה-XML מוגדרים על-ידי מרחב השמות שמגדיר את האלמנטים שלהם.

כיוון שקובץ XSD הוא גם מסמך XML, גם בדוגמה שלנו המאפיין xmlns נמצא בקובץ ה-XSD. המאפיין הזה מציין שכל האלמנטים וטיפוסי הנתונים שבהם סכמה זו משתמשת מוגדרים במרחב השמות: http://www.w3.org/2001/XMLSchema. למשל, הטיפוסים מספר שלם (integer) ומחרוזת (string) מוגדרים במרחב השמות הזה.

נוסף על כך, המאפיין הזה מגדיר כי התחילית של האלמנטים ושל טיפוס הנתונים, אשר שייכים למרחב השם http://www.w3.org/2001/XMLSchema, מורכבת מהתווים 'xs:' .

כאשר מוגדר מרחב שמות לאלמנט מסוים, כל האלמנטים הבנים שיש להם אותה תחילית (prefix), יהיו שייכים לאותו מרחב שמות.

המאפיין השני: targetNamespace="http://www.olympicGames"

מציין שכל האלמנטים שמוגדרים באמצעות סכמה זו (בדוגמה שלנו: GAMES, GENDER, AGE, SPORT, NAME, ID, ATHLETE, ID, COUNTRY) הם ממרחב השמות הזה. אנו מניחים כי הכתובת של אתר האולימפיאדה היא: http://www.olympicGames מאפיין זה נמצא רק בסכמות והוא מגדיר את מרחב השמות (namespace) של האברים שמוגדרים בסכמה, כלומר הוא מגדיר איך נראים האלמנטים של מסמך ה-XML.

למשל, בדוגמה של המשלחת האולימפית, האלמנט <age> שמציין את גיל השחקן צריך להיות מסוג מספר שלם והגדרתו תיראה כך:

```
<xs:element name=" age " type="xs:integer" />
```

לפיכך, האלמנט "age" בקובץ ה-XML צריך להיות מסוג מספר שלם, כי כך הוא הוגדר בסכמה שלו.

המאפיין השלישי: xmlns="http://www.olympicGames"

סכמה מגדירה אילוצים על אלמנטים שנמצאים בקובצי XML. כדי להכיר את האילוצים, היא צריכה להכיל הגדרות של הקבצים האלה.

המאפיין האחרון: elementFormDefault="qualified"

כל האלמנטים של מסמך XML, שמשתמש בקובץ ה-XSD הזה, צריכים להתאים להגדרות של מרחב השמות שמוגדר בסכמה.

ב. הגדרת אלמנט XML פשוט

אלמנט XML הוא אלמנט פשוט. נזכיר כי אלמנט פשוט יכול להכיל טקסט בלבד ויכול להיות מטיפוסים שונים, למשל int, string, date, או טיפוסים שאנו מגדירים בעצמנו. נוסף על כך, ניתן להוסיף הגבלות (facets) על טיפוסי הנתונים על-מנת להגביל את הערכים שהם יכולים לקבל ולאלץ את הנתונים כך שיתאימו לתבנית מסוימת שאנו מגדירים. לדוגמה, עבור אלמנטי XML שלהלן:

```
<id>43</id>
<name> Israeli Ben</name>
<sport>football</sport>
<age>17</age>
```

```
<gender> true </gender>
```

תיראה הסכמה כך:

```
<xs:element name="id" type="xs:integer"/>
<xs:element name="name" type="xs:string"/>
<xs:element name="sport" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="gender" type="xs:boolean"/>
```

בסכמת XML קיימים טיפוסי נתונים רבים. להלן הטיפוסים הנפוצים ביותר:

```
xs:string
xs:decimal
xs:integer
xs:boolean
xs:date
xs:time
```

ניתן להגדיר ערכים של ברירת מחדל וערכים קבועים בעבור אלמנטים פשוטים. לדוגמה, כדי להגדיר ערך ברירת מחדל באלמנט ששמו flowercolor, נרשום:

```
<xs:element name="flowercolor" type="xs:string" default="rose"/>
```

הגדרה זו מציינת אלמנט פשוט בשם flowercolor מהטיפוס מחרוזת עם ערך בררת מחדל המקצה את הערך rose כטקסט האלמנט, אלא אם כן הגדרנו ערך אחר.

כדי להגדיר ערך קבוע שלא ניתן לשינוי, נרשום את ההגדרה הזאת:

```
<xs:element name="flowercolor" type="xs:string" fixed="rose"/>
```

בדוגמה זו הערך קבוע ולא ניתן יהיה לספק שם אחר.

ג. מאפייני XSD – XSD Attributes

כל המאפיינים מוגדרים כאלמנטים פשוטים. שימו לב שהאלמנט אשר מכיל את אלמנט המאפיין אינו אלמנט פשוט (אלמנטים פשוטים יכולים לקבל רק טקסט ואינם יכולים לקבל הרחבה של מאפיינים).

- הגדרת המאפיינים – attributes

```
<xs:attribute name="attributeName" type="attributeType"/>
```

לדוגמה, בעבור אלמנט XML הזה :

```
<country id="Israel ">
```

נוכל להגדיר את המאפיין id כך :

```
<xs:attribute name="id" type="xs:string"/>
```

- הגדרת ערכים קבועים וערכי ברירת מחדל למאפיינים :

```
<xs:attribute name="id" type="xs:string" default="ENGLISH"/>
```

או

```
<xs:attribute name="id" type="xs:string" fixed="FRENCH"/>
```

כל המאפיינים הם מאפיינים אופציונאליים, כלומר אינם חייבים להוסיף מאפיין לאלמנט. כדי להגדיר באופן מפורש אם מאפיין כלשהו הוא אופציונאלי או שחובה להגדיר אותו, נשתמש במאפיין use. להלן דוגמה להגדרה של מאפיין אופציונאלי:

```
<xs:attribute name="id" type="xs:string" use="optional"/>
```

האלמנט בקובץ ה-XML יכול להיראות כך :

```
<name id="123456789">avi</name >
```

או כך :

```
<name>avi</name>
```

הגדרה של מאפיין שהוא חובה :

```
<xs:attribute name="id" type="xs:string" use="required"/>
```

האלמנט בקובץ ה-XML יכול להיראות כך :

```
<name id="123456789">avi</name >
```

אבל **לא** כך :

```
<name>avi</name>
```

ד. הגבלות

כאשר אנו מגדירים טיפוס נתונים עבור אלמנט או מאפיין כלשהו, אנו למעשה מגבילים את תוכן האלמנט. לדוגמה, אם אלמנט XML הוא מטיפוס מספר שלם ונכניס ערך מחרוזת כתוכן האלמנט, המסמך יפר את חוקי האימות והוא יהיה לא תקף (כלומר invalid).

נוסף על ההגבלות על טיפוס הנתונים, ניתן להוסיף הגבלות אחרות לאלמנטים ומאפיינים. המגבלות האלה נקראות facets. נציג דוגמאות מספר להגבלות על ערכים :

- הגבלה של גיל לתחום הערכים :

```
<xs:element name="age">
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="120"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

הגדרה של אלמנט בשם "age" , הכולל הגבלה על הערך. הערך חייב להיות בין 0 ל-120.

שימו לב שהמאפיין type הושמט באלמנט עצמו אולם בא לידי ביטוי כמאפיין base בתיאור ההגבלה. נוסף על כך, כל אלמנט ההגבלה גלום בתוך האלמנט simpleType.

דוגמה נוספת : נתאר הגבלה של ערך כחלק מקבוצה של ערכים ; הערכים כעת יכולים להיות רק 39 או 40 או 41.

```
<xs:element name="age">
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:enumeration value="39"/>
    <xs:enumeration value="40"/>
    <xs:enumeration value="41"/>
  </xs:restriction>
</xs:simpleType>
</xs:element>
```

לסיכום, נתאר חלק ממסמך Players.xml המוכר לנו מדוגמאות קודמות :

```
<?xml version="1.0" ?>
<Players>
   <Player>
```

```
        <PlayerID>1</PlayerID>
        <PlayerName>the king of chess</PlayerName>
        <NumbeOfVisits>566</NumbeOfVisits>
        <BestScore>565446</BestScore>
        <LastVisit>2008-01-01</LastVisit>
        <GameWithBestScore>Chess</GameWithBestScore>
        <ScoreInAllGames>565446</ScoreInAllGames>
    </Player>
    <Player>
        <PlayerID>2</PlayerID>
        <PlayerName>Prince</PlayerName>
        <NumbeOfVisits>44</NumbeOfVisits>
        <BestScore>256666</BestScore>
        <LastVisit>2008-02-02</LastVisit>
        <GameWithBestScore>Mahjung</GameWithBestScore>
        <ScoreInAllGames>3456642</ScoreInAllGames>
    </Player>
</Players>
```

כפי שרואים, קובץ ה-XML מכיל שני שחקנים ולכל שחקן שמורים שבעה אלמנטים בנים שמכילים עליו מידע.

הערה: התאריך צריך להיות בפורמט הזה: "YYYY-MM-DD", Y מציין שנה (4 ספרות, למשל 2008), M מציין חודש (2 ספרות) ו-D מציין יום (2 ספרות), ובין השנה, החודש והיום מפריד התו "-".

ולהלן קובץ XSD, שבאמצעותו נבדוק את המבנה ואת התוכן של קובץ ה-XML הזה.

קובץ ה-XSD יכול להיראות כך:

Players.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Players">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Player">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="PlayerID" type="xs:integer" />
              <xs:element name="PlayerName" type="xs:string" />
              <xs:element name="NumbeOfVisits" type="xs:integer" />
              <xs:element name="BestScore" type="xs:integer" />
              <xs:element name="LastVisit" type="xs:date" />
              <xs:element name="GameWithBestScore" type="xs:string" />
              <xs:element name="ScoreInAllGames" type="xs:integer" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

הערות:

קובץ זה מכיל מספר מחוונים שמגדירים את המבנה והקינון של האלמנטים.

- המחוון <xs:complexType> מציין כי אלמנט מורכב יכול להכיל אלמנטים אחרים בתוכו.
- המחוון </xs:sequence> מציין כי האלמנטים שהם הבנים לאלמנט הזה חייבים להופיע ברצף ובאותו הסדר. כל אלמנט יכול להופיע בין 0 למספר לא מוגבל.
- המחוון maxOccurs בעל הערך "unbounded" מציין שאלמנט זה יכול להופיע אין-ספור פעמים.

שאלה 6.7

בנו קובץ XML פשוט שמכיל מידע על אודות המשתמשים באתר המשחקים שעליו למדתם בפרק 4. הנתונים מכילים את הפרטים האלה: המספר המזהה של המשתמש, שם

המשתמש, הסיסמה, השם הפרטי, שם המשפחה, הכתובת, המין, שנת הלידה והדירוג של המשתמש. כמו כן, בנו קובץ XSD ובדקו בעזרתו (באופן ידני) אם הנתונים שהוכנסו לקובץ ה-XML הם מתאימים.

שאלה 6.8

הסבירו כיצד מגדירים ערך קבוע למאפיין של XSD?

שאלה 6.9

פרטו את שלושת טיפוסי הנתונים הקיימים בסכמת XML .

שאלה 6.10

הגדירו אלמנט בשם "note" שפירושו : ציון, שהערך שלו חייב להיות בין 0 ל-100.

6.4 שמירה ואחזור של הנתונים בקובצי XML

בסעיף הזה נלמד איך לשמור נתונים בתבנית XML בעזרת טכנולוגיית ה-ADO.NET. כידוע, טכנולוגיית ה-ADO.NET היא הטכנולוגיה שבאמצעותה ניתן להתקשר עם מסדי הנתונים בטכנולוגיית מיקרוסופט (NET). כפי שציינו בפרק 4, ADO.NET תומכת בשתי שיטות חיבור, מקושרת (connected) ולא-מקושרת (disconnected). בסביבה הלא-מקושרת אנו משתמשים בעצם מהטיפוס DataSet, השומר את נתוני הטבלה שאוחזרה כאשר מנתקים את הקשר עם בסיס הנתונים. העצם DataSet מכיל טבלה (או קבוצה של טבלאות) שנשמרת בזיכרון של המחשב והוא המייצג של מסד הנתונים.

DataSet שומר את הנתונים שלו בצורת xml, ולכן תהליך שמירת הנתונים בקובץ XML יהיה קצר ופשוט מאוד. כל מה שצריך לעשות הוא להפעיל את הפעולה WriteXml של המחלקה DataSet :

```
DataSet ds.WriteXml("d:\\asp\\tblExample.xml", XmlWriteMode.WriteSchema);
```

הפעולה WriteXml מקבלת שני פרמטרים:

- הפרמטר הראשון מהטיפוס מחרוזת אשר מכיל את שמו ואת מסלולו של קובץ ה-XML, שנרצה לשמור בו את הנתונים.
- הפרמטר השני (XmlWriteMode.WriteSchema) מציין כי הקובץ XML יכיל גם סכמה מתאימה.

לדוגמה, נניח כי בנינו טבלה בשם tblExample המכילה את הנתונים האלה:

ID	userName	age	password	gender
1	Beny	15	Beny123	False
2	Meny	16	meny	False
3	Molly	15	molly	True
4	Polly	16	polly	True
5	Ofir	14	Ofir567	False

איור 4-6
תוכן טבלה tblExample

נכתוב תכנית writeXml.aspx שיוצרת קובץ XML בשם tblExample.xml המתאר את מבנה הנתונים של הטבלה tblExample. הקובץ הזה מכיל גם סכמה מתאימה. בתכנית ניצור תחילה עצם מהטיפוס DataSet שמכיל את תוצאות השאילתה שמאחזרת את נתוני הטבלה, ולאחר מכן נשתמש בפעולה WriteXml כדי ליצור קובץ XML בשם tblExample.xml.

```
<!-- writeXml.aspx -->
<%@ Page Language="C#" Debug="true"%>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Xml"  %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
```

```
    public DataSet ds = new DataSet();
    protected void Page_Load(object sender, EventArgs e)
    {
                                            // יצירת קשר למסד הנתונים
 string connStr = @"Data
Source=.\SQLEXPRESS;AttachDbFilename=D:\asp\WebSite4\App_Data\DataBases\ExampleDB.mdf;Integr
ated Security=True;User Instance=True";
         SqlConnection conn = new SqlConnection(connStr);

                                            // אחזור נתונים ממסד הנתונים
        SqlCommand cmd = new SqlCommand();
        cmd.CommandText = "Select* from tblExample";
        cmd.Connection = conn;
        conn.Open();
        SqlDataAdapter adapter = new SqlDataAdapter(cmd);
        adapter.Fill(ds, "tblExample");


                                            // יצירת קובץ XML המכיל סכמה
        ds.WriteXml("d:\\asp\\tblExample.xml", XmlWriteMode.WriteSchema);

                                            // סגירת הקשר למסד הנתונים
        conn.Close();
    }
   </script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Example</title>
</head>
<body>
    <form id="form1" runat="server">
     </form>
</body>
```

```
</html>
```

לאחר הרצת התכנית פִּתחו את הקובץ tblExample.xml שנוצר בספרייה d:\asp. שימו לב כי הקובץ מכיל את הסכמה ולאחר מכן את הגדרת הנתונים בשפת XML.

שאלה 6.11

פִּתחו את קובץ tblExample.xml ורִשמו את :

א. הגדרות הסכמה שנוצרה ;

ב. תיאור הנתונים בשפת XML של השורה הראשונה בטבלה.

לקריאת קובץ ה-XML נשתמש בפעולה ReadXml של האובייקט DataSet. פעולה זו מקבלת שני פרמטרים :

- הפרמטר הראשון הוא מהטיפוס מחרוזת והוא מציין את שם קובץ ה-XML ואת המסלול אליו.
- הפרמטר השני (XmlReadMode.ReadSchema) מציין כי יש להשתמש בסכמה מתאימה, לדוגמה :

```
DataSet ds.ReadXml("d:\\asp\\tblExample.xml", XmlReadMode.ReadSchema);
```

נכתוב קובץ aspx היוצר עצם מהטיפוס DataSet שמכיל את הנתונים שנשמרו בקובץ tblExample.xml שיצרנו קודם לכן. לאחר מכן הוא מזמן את הפעולה ListOfUsers() שמציגה את הנתונים שנשמרו בעצם מהטיפוס DataSet. שימו לב, כי בדף ASP הזה אנו לא משתמשים במסד הנתונים.

```
<!--readXml.aspx -->
<%@ Page Language="C#" Debug="true"%>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Xml"  %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">
   public DataSet ds = new DataSet();
   protected void Page_Load(object sender, EventArgs e)
   {
// DataSet   קריאת קובץ XML לתוך עצם מטיפוס
      ds.ReadXml("d:\\asp\\tblExample.xml", XmlReadMode.ReadSchema);
   }

   public void ListOfUsers()                                   // הצגת תוכן הטבלה
   {
       //work with data in computer memory
        Response.Write("<h1> My List</h1>");
        string allUsersString="<table>";

        allUsersString+="<tr><td>ID</td><td>userName</td><td>age</td><td>password</td><td>gender
                          </td></tr>";
        foreach (DataRow dr in ds.Tables["tblExample"].Rows)
        {
          allUsersString+="<tr><td>"+dr["ID"].ToString()+"</td><td>"+dr["userName"].ToString()+"</td>";
          allUsersString+="<td>"+dr["age"].ToString()+"</td>";
          allUsersString+="<td>"+dr["password"].ToString()+"</td>";
          allUsersString += "<td>";
          if ((bool)dr["gender"])
              allUsersString += "Female";
          else
              allUsersString += "Male";
          allUsersString+="</td></tr>";

        }
        allUsersString+="</table>";
        Response.Write(allUsersString);
```

```
      Response.Write("<br/>");
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
  <title>Example</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <%ListOfUsers(); %>
  </div>
  </form>
</body>
</html>
```

6.5 הצגת מסמכי XML כמסמכי HTML

כאשר מריצים מסמך XML, הדפדפן אינו מכיל מידע איך להציג אותו. הסיבה לכך היא שקובץ ה-XML מכיל הגדרות המתארות את הנתונים אבל אינו מכיל הגדרות המתארות כיצד להציגם. לפיכך, הדפדפן מציג את התוכן ואת התגים יחד, בדיוק כפי שהם שמורים בקובץ ה-XML. הדפדפן בודק אם הקובץ מעוצב כהלכה (Well-formed): אם כן – הוא מציג את האלמנטים בצבע כדי להקל את הבנת מבנה הקובץ; אם לא – הוא מציג הודעת שגיאה. ישנן שיטות אפשריות שונות להצגת את המידע של קובץ XML. לדוגמה, ניתן להשתמש בגיליון סגנונות מדורג – CSS (Cascade Style Sheets) או בשפת תכנות, כגון C#, או בשפת תסריט כלשהי, כגון JavaScript. בכל אחת מהשיטות מצורף לקובץ ה-XML קובץ נוסף שמגדיר כיצד ייראה המסמך בדפדפן לפי השיטה וההגדרות שנבחרו.

בדוגמה שלהלן נתאר את האופן שבו קוראים ומציגים קובץ XML בעזרת JavaScript. לשם כך, נשתמש בעצם מהטיפוס XML DOM. העצם XML DOM מכיל כמה פעולות ומאפיינים שבאמצעותם ניתן לעבד קובצי XML. פעולות אלה מאפשרות לאחזר אלמנטים

ומאפיינים של קובץ XML, לעבור בין האלמנטים, לשנותם, למחקם ולהוסיף אלמנטים ומאפיינים חדשים ועוד. עצם מהטיפוס XML DOM שומר את הנתונים שבקובץ XML במבנה עץ היררכי בדיוק כפי שהם מופיעים במסמך.

לדוגמה, נתבונן בקובץ Players.xml :

```
<?xml version="1.0" ?>
<Players>
   <Player>
        <PlayerID>1</PlayerID>
        <PlayerName>the king of chess</PlayerName>
        <NumbeOfVisits>566</NumbeOfVisits>
        <BestScore>565446</BestScore>
        <LastVisit>2008-01-01</LastVisit>
        <GameWithBestScore>Chess</GameWithBestScore>
        <ScoreInAllGames>565446</ScoreInAllGames>
   </Player>
   <Player>
        <PlayerID>2</PlayerID>
        <PlayerName>Prince</PlayerName>
        <NumbeOfVisits>44</NumbeOfVisits>
        <BestScore>256666</BestScore>
        <LastVisit>2008-02-02</LastVisit>
        <GameWithBestScore>Mahjung</GameWithBestScore>
        <ScoreInAllGames>3456642</ScoreInAllGames>
   </Player>
</Players>

   <Player>
        <PlayerID>3</PlayerID>
        <PlayerName>sendbab</PlayerName>
        <NumbeOfVisits>565</NumbeOfVisits>
        <BestScore>286866</BestScore>
        <LastVisit>2008-03-03</LastVisit>
        <GameWithBestScore>Ninja</GameWithBestScore>
```

```
        <ScoreInAllGames>4564446</ScoreInAllGames>
    </Player>
</Players>
```

וכעת, נכתוב תסריט היוצר קובץ HTML שיכיל את כל האלמנטים של מסמך XML, ונציג אותם בטבלה. נתבונן בקוד הזה:

```
<script type="text/javascript">
function loadXMLDoc()
{
   var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
   xmlDoc.load("Players.xml")
   var XMLroot = xmlDoc.documentElement;
   document.write("<table border='1' bgColor='#F5FFEE'>")
   for (i = 0; i <= XMLroot.childNodes.length-1; i++)
   {
       document.write("<tr>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(0).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(1).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(2).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(3).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(4).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(5).text+"</td>");
        document.write("<td>"+XMLroot.childNodes.item(i).childNodes.item(6).text+"</td>");
       document.write("</tr>");
   }
   document.write("</table>");
}
</script>
```

עתה נעבור על הקוד ונסביר אותו משפט משפט.

- בהתחלה יצרנו עצם חדש מסוג XML DOM:

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
```

עצם זה מכיל כמה מאפיינים ופעולות שבהם נשתמש כדי להציג את הנתונים שבקובץ XML כקובץ HTML.

- הפעולה הראשונה מאפשרת לטעון את קובץ ה-XML שאנו מבקשים להמיר.

```
xmlDoc.load("Players.xml")
```

לאחר מכן, יצרנו פנייה לשורש (root), שממנו נתחיל את המעבר על הבנים של השורש.

```
var XMLroot = xmlDoc.documentElement;
```

כדי לעבור על כל האלמנטים, עלינו לדעת את מספר האלמנטים שיש במסמך ה-XML. לשם כך, ניעזר במאפיין xmlDoc.documentElement.childNodes.length. כעת נוכל לכתוב מבנה בקרה שבו נעבור על כל אחד מהאלמנטים.

```
for (i = 0; i <= XMLroot.childNodes.length-1; i++)
```

הגישה לבנים של השורש מתבצעת על-ידי ההוראה XMLroot.childNodes.item(i), כאשר "i" הוא מספר האלמנט. אבל מאחר שהבנים של השורש מכילים גם הם אלמנטים, נשתמש שוב בפעולה childNodes.item(x) (כאשר x הוא מספר האלמנט). בסוף המשפט כותבים את המאפיין text כדי לקבל את הערך של האלמנט.

```
XMLroot.childNodes.item(i).childNodes.item(0).text
```

לאחר הרצת הקובץ בדפדפן נקבל את הפלט הזה:

ScoreIn AllGames	GameWith BestScore	Last Visit	Best Score	Numbe OfVisits	Player Name	PlayerID
565446	Chess	2008-01-01	565446	566	the king of chess	1
3456642	Mahjung	2008-02-02	256666	44	mr. Bean	2
4564446	Ninja	2008-03-03	286866	565	sendbab	3

איור 5-6
פלט המציג תוכן של טבלה שאוחסנה כקובץ XML

6.6 סִדרוּת

סִדרוּת (**Serialization**) הוא תהליך שבו ניתן לשמור את מצבו של העצם בתבנית בינארית ולשלחו בערוץ תקשורת או לשמרו לאורך זמן באמצעי אחסון, כגון קובץ. מצבו של עצם כלשהו מוגדר מערכי השדות שלו. לדוגמה, הגדרנו עצם מהטיפוס Point השומר מידע על נקודה (5,2). במחלקה זו מוגדרים שני שדות: x ו-y. בעצם שיצרנו אתחלנו את ערכו של x ל-5 ואת ערכו של y ל-2. ביצוע תהליך סדרות מאפשר לשמור מידע זה (ערכים 5 ו-2) בצורה בינארית, כלומר כאוסף של 0 ו-1. בהמשך נוכל לשחזר את מצב העצם מתוך התבנית הבינארית כדי להוסיף ולבצע פעולות שבהם משתמשים בעצם הזה ולהמשיך להתייחס לאותה נקודה. התבנית הבינארית לשמירת מצבו של העצם חוסכת מקום בזיכרון ומאפשרת העברה של העצם גם ברשת תקשורת. שיטה העברה זו היא מהירה והמאפשרת גם הצפנה ודחיסה.

למעשה, ניתן להפעיל תהליך של סדרות גם על קבצים שהם בתבנית XML. סדרות של מסמךXML מאפשר להמיר מסמך XML לתבנית שהיא פשוטה, וכל שפות התכנות יכולות להמיר בקלות את המסמך לתבנית משלהם. ניתן גם להמיר עצם שעבר תהליך סדרות ל-XML ולהיפך.

להפעלת תהליך סדרות על קבצים שהם בתבנית XML יש מספר יתרונות. נציין את היתרונות העיקריים:

1. ניתן לפתוח אותו בעזרת עורך טקסט פשוט (למשל Notepad), לקרוא ולבחון את תוכנו.
2. ניתן לשלוח אותו לכל יישום אינטרנט, גם אם הוא לא עובד בטכנולוגית ASP.

יחד עם זאת, יש לתהליך זה גם חסרונות:

1. ב Xml התהליך הוא פחות יעיל משום הקובץ שנוצר הוא גדול מאוד ולכן שליחת קובץ גדול ברשת תקשורת תארך זמן רב יותר.
2. כל אחד יכול לפתוח אותו עם כל עורך טקסט ולכן יהיה קשה לשמור על החשאיות של הנתונים.

תהליך הסדרות של עצם לקובץ XML הוא פשוט מאוד.

לדוגמה, נגדיר מחלקה בשם Person, המגדירה שתי שדות : FirstName, LastName.

להלן הגדרת המחלקה

```
public class Person
{
  public  string FirstName;
  public  string LastName;
}
```

כעת נגדיר מחלקה בשם TestXML המגדירה עצם מטיפוס Person, מציבה ערכים וממירה את העצם לקובץ XML המוצג על הצג.

```
class TestXML
{
   static void Main(string[] args)
   {
      Person p=new Person();                    //Person יצירת עצם מטיפוס
      p.FirstName = "Jeff";                     // השמת ערכים בשדות
      p.LastName = "Price";
                                                // יצירת מופע של מחלקה המטפלת בסדרתיות
                                                //Person המופע מקבל כפרמטר מופע של העצם
        System.Xml.Serialization.XmlSerializer x = new System.Xml.Serialization.XmlSerializer(p.GetType());
        x.Serialize(Console.Out, p);            // הפעלת פעולה המבצעת את הסדרתיות
        Console.WriteLine();                    // XML הצגת קובץ
         Console.ReadLine();                    // קריאת נתונים ממשתמש כדי לסיים את התצוגה
   }
}
```