- Late Bound Data Expressions
הכוח שמאחורי הרעיון

ג'סטין-יוסף אנג'ל

מסמך זה הורד מהאתר http://www.underwar.co.il.

מחבר המסמך איננו אחראי לכל נזק, ישיר או עקיף, שיגרם עקב השימוש במידע המופיע במסמך, וכן לנכונות התוכן של הנושאים המופיעים במסמך. עם זאת, המחבר עשה את מירב המאמצים כדי לספק את המידע המדויק והמלא ביותר.

הזכויות שמורות ל**ג'סטין יוסף אנג'ל**.

ג'סטין-יוסף אנג'ל

the_j_angel@hotmail.com

http://blogs.israelblog.co.il/justin_angel/

שלום לכולם,

נתחיל בהגדרה כוללנית, Late Bound Data Expressions מיועדים לקבלת מידע (משתנים) בתצוגת עיצוב ולפרמט אותו לתצורה נבחרת. נעשה פירוש רש"י. ב"קבלת מידע (משתנים)" אנו מתכוונים כל אובייקט המכיל מידע, הכל ממחרוזות, DataReaderים וכלה ב-DataSetים. ב"בתצוגת עיצוב" אנו מתכוונים שבזמן שתמיד ניתן לשנות את תוכן המידע כחלק מה-Code Behind, ה- Late Bound Data Expressions מאפשרים לנו לשנות את המידע בתצוגת ה-Design. ב"פרמט אותו לתצורה נבחרת" אנו אומרים למעשה שאנו נקבל את הנתונים האלו ונגדיר להם אלגוריתם שהם יעברו לפני תצוגה ללקוח.

ישנם שתי שיטות בנויות בדוט נט ל-Late Bound Data Expressions: הראשונה DataBinder.Eval, והשנייה String.Format. שתיהן למעשה מאפשרות לקחת מידע במהלך אירוע ה-OnDataBound ולהציגו או לשנות לו את אופי התצוגה. נסקור את שתי הפונקציות הללו במהלך אפשרויות א' ו-ב'. אם אתם כבר מכירים ומבינים את דרך פעולתם, אני ממליץ שתקפצו ישר לחלק ג'.

א. DataBind my Page! (או, תראו את הפשטות)

בשיטה הזאת אנחנו עושים דבר מאוד הגיוני - DataBind לדף עצמו. נגדיר פקדים\משתנים ברמת הדף ונוכל לעשות להם DataBind. בואו נביט על דוגמה. החלטנו שאנו רוצים אפשרות לשנות בצורה תכנותית את הכותרת (<title/>myTitle<title>) של הדף. כידוע, אין כזאת אפשרות בנויה כחלק מהפקד System.Web.UI.Page.

דבר ראשון נצהיר על משתנה מחרוזת חדש ברמת הדף. לאחר מכן, נכניס לתוכו ערך, נראה כיצד משלבים אותה בתצוגת Design ונעשה DataBind. הצהיר על המשתנה מחרוזת ברמת הדף ניתן לו ערך כלשהו:

```
...
public class testingWebForm : System.Web.UI.Page
{
  // Declare Page Title string object
  protected string myStr;

  private void Page_Load(object sender, System.EventArgs e)
  {
    // Place some value in Page Title
   myStr = "פורום תפוז דוט נט";
  /* פרסומת סמויה */
  }
}
```

כמו כל String אחר ברמת ה-Page הצהרנו עליו ונתנו לו ערך. עכשיו הגיע הזמן להתחיל לראות חלק מהקסם של Late Bound Data Expressions. בתצוגת Design HTML של הדף נשנה את הערך של <title> כך שהוא יהיה ביטוי שעושה DataBind.

```
<html>
  <head>
    <title><%# myStr %></title>
  </head>
...
</html>
```

ועכשיו נגיד שבאירוע Page_load שצריך לעשות Page.DataBind.

```
...
public class testingWebForm : System.Web.UI.Page
{
  // Declare Page Title string object
  protected string myStr;

  private void Page_Load(object sender, System.EventArgs e)
  {
    // Place some value in Page Title
   myStr = "פורום תפוז דוט נט";

   // DataBind the page itself!
    this.DataBind();
  }
}
```

ולהוכחה, בסופו של דבר קיבלנו:

```
<head>
   <title>נט פורום תפוז דוט</title>
</head>
```

עד כה הכל היה מאוד פשוט: עשינו DataBind לפקד ברמת הדף לתוך תצוגת ה-Design שלנו. בואו ניקח את זה צעד אחד קדימה וננסה לעשות DataBind לאובייקט יותר רציני מאשר String, למשל DropDownList. אנו רוצים שהטקסט של הפריט הנבחר ב-DropDownList יוצג כחלק מהדף לאחר PostBack.

```
// WebForm.aspx
  <!-- This DropDownList will contain two ListItems -->
  <asp:DropDownList Runat="server" ID="myDDL">
     <asp:ListItem>עם סלים לאוטובוס עלתה גברת</asp:ListItem>
     <asp:ListItem>שהגברת עם הסלים האוטובוס נסע לפני
עלתה</asp:ListItem>
  </asp:DropDownList>

  <!-- This Button Will case a PostBack and Call myBtn_Click -->
  <asp:Button Runat="server" Text="Cause PostBack"
OnClick="myBtn_Click" id="Button1" />

  <!-- This Late Data Bound expression will diplay what is the currently
selected item
  <%# myDDL.SelectedItem.Text %>
// WebForm.asp.cs
 public void myBtn_Click(object sender, System.EventArgs e)
 {
   // On button click - DataBind the Page
    this.DataBind();
 }
```

מאוד פשוט וקל. נבחר ערך ב-DDL, נלחץ על הכפתור וכחלק מתהליך ה-DataBinding של הדף עצמו יוצג הטקסט של הפריט שנבחר. ולהוכחה, כאשר נבחר את "לאוטובוס עלתה גברת עם סלים" ונלחץ על הכפתור יודפס "לאוטובוס עלתה גברת עם סלים".

עד כאן ניקח את האפשרות הזאת, אבל חשוב לציין שכמובן שגם אפשר Page DataBinding לכל פקד ברמת הדף, החל מערכים של מחרוזות או מספרים, השעה הנוכחית וכלה במספר השורות בטבלה מסויימת ב-DataSet.

ב. DataBinder.Eval (או: כמו בלונדינית - טיפש וחמוד)

כאן מתגלה חלק מהיופי של Late Bound Data Expressions. השיטה DataBinder.Eval נמצאת בשימוש רב בפקדים DataGrid, DataList ו-Reapter. בפקדים אלו יש לנו אפשרות לכתוב כמעט את כל הקוד שמעצב את המידע בתצוגת Design. בואו נביט על מה למעשה קורה למשל ב-DataBinding מ-DataView: כאשר ביקשנו כזו DataBinding, כל שורה נשלחת כ-Container.DataItem ומבצעת DataBinding לקוד שמעצב את המידע.

הנה טבלה לדוגמה שאיתה נעבוד עד סוף מאמר זה. הטבלה היא נשים זקנות עם סלים ומכילה שלושה עמודות: שם הגברת עם סלים, כמות הסלים ומחירם הסופי.

שם הגברת עם סלים	כמות הסלים	מחיר הסלים
strOldLadyWithBags_name	intNumberOfBags	intFinalCost
לאה	2	10
אסתר	8	2,000
סימה	4	50

בואו נביט לשנייה על דוגמה פשוטה. אנו רוצים להציג בפקד DataList רשימה של השמות של נשים זקנות עם סלים.

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
   לגברת עם הסלים קוראים
   <%# DataBinder.Eval(Container.DataItem,
"strOldLadyWithBags_name")%>
  </ItemTemplate>
</asp:DataList>
```

נניח ועשינו שאילתא שמחזירה שמות של של נשים זקנות עם סלים ול-DataView שלה עשינו DataBind ל-myDataList. כמו כן נניח שהרשימה מחזירה: לאה, אסתר וסימה. אם נריץ את הדף נקבל: "לגברת עם הסלים קוראים לאה. לגברת עם הסלים קוראים אסתר. לגברת עם הסלים קוראים סימה".

נמשיך עם הדוגמה ונבקש בנוסף את מספר הסלים שכל גברת עם סלים עלתה לאוטובוס וכמה היא שילמה בחנות על הסלים שלה.

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# DataBinder.Eval(Container.DataItem,
"strOldLadyWithBags_name")%>
    עם עלתה לאוטובוס
    <%# DataBinder.Eval(Container.DataItem,
"intNumberOfBags")%>
   סלים
   שעלו סה"כ
   <%# DataBinder.Eval(Container.DataItem, "intFinalCost")%>
  </ItemTemplate>
</asp:DataList>
```

אם נריץ הדוגמה נקבל "לאה עלתה לאוטובוס עם 4 סלים שעלו סה"כ 100. אסתר עלתה לאוטובוס עם 8 סלים שעלו סה"כ 2,000. סימה עלתה לאוטובוס עם 2 סלים שעלו סה"כ 50.". עד עכשיו, הראנו כיצד בתוך ItemTemplete של DataList, DataGrid ו-Repeater ניתן להציג מידע בצורה חוזרת מתוך מסד הנתונים.

עכשיו בואו ונכיל איזהשהו פורמט על אחד הנתונים. נכון שזה נראה מצחיק שרשום שארבעה הסלים של לאה עלו מאה? מאה מה? מאה רובל? מאה שקלים? מאה דולר? אנו צריכים להבהיר כי מדובר בסוג של מטבע.

השיטה DataBinder.Eval מקבלת שלושה פרמטרים: הפקד שמכיל את המידע (DataContainer), לאיזה שדה\תא אנו מתייחסים בפקד שמכיל את המידע והשלישי שהוא אופציונאלי זה כיצד לפרמט את המחרוזת. בואו נראה דוגמה כיצד נפרמט את מחיר הסלים כך שיהיה בתצורת מטבע.

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# DataBinder.Eval(Container.DataItem,
"strOldLadyWithBags_name")%>
    עם עלתה לאוטובוס
    <%# DataBinder.Eval(Container.DataItem, "intNumberOfBags")%>
  סלים
  שעלו סה"כ
  <%# DataBinder.Eval(Container.DataItem, "intFinalCost",
{0:c})%>
  </ItemTemplate>
</asp:DataList>
```

וכאשר נציג את המסך נראה: "לאה עלתה לאוטובוס עם 4 סלים שעלו סה"כ 100₪. אסתר עלתה לאוטובוס עם 8 סלים שעלו סה"כ 2,000₪. סימה עלתה לאוטובוס עם 2 סלים שעלו סה"כ 50₪."

נכון שזה נראה הרבה יותר טוב? ישנן אפשרויות רבות רבות להכיל פורמט תצוגה מסויים על מידע. אפשר לפרמט תאריך להיות תאריך ארוך, קצר, לפי שעון בינלאומי, לועזי קצר, אירופאי קצר. אפשר לפרמט שעות להכיל רק שעות, רק שעות ודקות, להציג AM/PM. אפשר לפרמט להציג מספר כמטבע, כאחוזים, כמספר מדעי, לאכוף מספר ספרות אחרי הנקודה. והאפשרויות רבות. כנראה שהחבר'ה במיקרוסופט באמת חשבו על לפרמט נתונים שונים לפורמטים מסויימים כשהם יצרו את האפשרות הזאת. בהמשך נראה את הכוח האמיתי של Late Bound Data Expressions.

ב. String.Format (או: יותר מרחב פעולה, הרבה יותר פעולות סיזיפיות)

DataBinder.Eval עושה בשבילנו שתי פעולות: הראשונה, חוסכת מאתנו פעולות סיזפיות של המרה לפני שנוכל לגשת למידע. השנייה, אם ביקשנו לפרמט את המידע היא קוראת בשבילנו ל-String.Format. מבחינת נקודת המבט שלנו מדובר על מתודות שקולות ועד סוף האפשרות הזו נדגים זאת.

מה זאת אומרת ש-DataBinder.Eval חוסכת מאתנו פעולות סיזפיות של המרה? בואו נביט על כיצד הדוגמה למעלה הייתה נראית אם היינו עושים אותה ב-String.Format:

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# String.Format("{0}", ((DataRowView)Container.DataItem)
["strOldLadyWithBags_name"]) %>
    עם עלתה לאוטובוס
    <%# String.Format("{0}", ((DataRowView)Container.DataItem)
["intNumberOfBags"]) %>
   סלים
   שעלו סה"כ
    <%# String.Format("{0:c}", ((DataRowView)Container.DataItem)
["intFinalCost"]) %>
  </ItemTemplate>
</asp:DataList>
```

פשוט מאוד לראות שעכשיו בכל DataBind צריך לעשות גם המרה (Casting) של Contrainer.DataItem לסוג DataRowView ולגשת לתא מסויים בתוכו לפי שם העמודה. כמו כן שימו לב ש-intFinalCost (עלות הסלים) עדיין בפורמט של מטבע.

למרות שהפתרון למעלה רץ ושקול לחלוטין מבחינה תחבירית לדוגמה של DataBinder.Eval, יש דרך יותר מקצועית לכתוב את הקוד למעלה:

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# String.Format("{0} 2} סלים במחיר {1} עלתה לאוטובוס עם {0:c}",
      ((DataRowView)Container.DataItem)
["strOldLadyWithBags_name"]),
      ((DataRowView)Container.DataItem) ["intNumberOfBags"]),
      ((DataRowView)Container.DataItem) ["intFinalCost"])   %>
  </ItemTemplate>
</asp:DataList>
```

המשפט שכתבנו, {0} עלתה לאוטובוס עם {1} סלים במחיר {c:2}, דומה מאוד למשפט מוכר של Console.WriteLine. למעשה הוא מציב את הערכים שבאים אחריו לתוכו בתוך משפט. שתי פעולות מתבצעות: פירמוט לתוך המשפט, פירמוט אישי לכל נתון (למשל, הפירמוט של המחיר למטבע). הדוגמה הזאת שקולה לחלוטין מבחינת תוצאות לדוגמה שלפניה.

ג. String.Format ופונקציות (או: כמה פשוט לשכלל מקרי קצה)

עד עכשיו עבדנו עם נתונים יפים ועגולים, לכל גברת עם סלים באמת יש סלים, לכל גברת יש שם, וכל סכום סלים הוא לא אפס. נשנה קצת את הנתונים המושלמים שלנו:

שם הגברת עם סלים	כמות הסלים	מחיר הסלים
strOldLadyWithBags_name	intNumberOfBags	intFinalCost
לאה	2	10
null	8	2,000
סימה	4	100

שימו לב לשינוי: אסתר שכחה את תג השם שלה ב-Dot net on the beach. אם ננסה להריץ את הדוגמה הקודמת (מועתקת למטה) נקבל: "לאה עלתה לאוטובוס עם 2 סלים במחיר 10₪. עלתה לאוטובוס עם 8 סלים במחיר 2,000₪. סימה עלתה לאוטובס עם 4 סלים במחיר 100₪.".

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# String.Format("{0} עלתה לאוטובוס עם {1} סלים במחיר {2:c}",
      ((DataRowView)Container.DataItem)
["strOldLadyWithBags_name"]),
      ((DataRowView)Container.DataItem) ["intNumberOfBags"]),
      ((DataRowView)Container.DataItem) ["intFinalCost"])   %>
  </ItemTemplate>
</asp:DataList>
```

בתקווה אתם כמתכנתים מבינים את התוצאה של הקוד הזה ואת הסיבות שהוא נוצר, אבל אני מבטיח לכם ששום לקוח-קצה לא יבין מה רוצים מהחיים שלו. צריך לפרמט עוד קצת את הנתונים. ניצור פונקציה שתקבל את שם הגברת עם סלים ויחזיר "גברת ללא שם" אם הוא ריק.

```
namespace myProject
{
  public class myPage : System.Web.UI.Page
  {
    ...
   public string FormatGivratName(object GivartName)
   {
     // Check if Givart Im Salim has name
     if (GivartName == null)
     {  // If Givart Im Salim has no name
        return "שם גברת ללא";
     }
     else
     { // Givart Im Salim has name
         return GivartName.ToString();
     }
   }
  }
}
```

נביט על הפונקציה הזאת. היא תקבל מחרוזת שהיא השם של הגברת, אם השם ריק היא תחזיר "גברת ללא שם" ואחרת תחזיר את השם מיוצג כמחרוזת. ניישם את הפונקציה הזאת על הדוגמה למעלה:

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# String.Format("{0} 2} סלים במחיר {1} עלתה לאוטובוס עם :c}",
      FormatGivratName(((DataRowView)Container.DataItem)
["strOldLadyWithBags_name"]),
      ((DataRowView)Container.DataItem) ["intNumberOfBags"]),
      ((DataRowView)Container.DataItem) ["intFinalCost"])   %>
  </ItemTemplate>
</asp:DataList>
```

כאשר נריץ את הדוגמה הזאת נקבל: "לאה עלתה לאוטובוס עם 2 סלים במחיר 10₪. **גברת ללא שם** עלתה לאוטובוס עם 8 סלים במחיר 2,000₪. סימה עלתה לאוטובס עם 4 סלים במחיר 100₪."

כמו שפירמטנו נתון אחד לפי תנאי מאוד קל, אפשר גם לפרמט כל נתון אחר לפי כל תנאי שנבחר. שימוש נפוץ בטכניקה הזאת היא כאשר אנו מחזירים ממסד הנתונים סימול שהפירוש שלו לא במסד הנתונים, אפשר להשתמש בכזו פונקציה בכדי להחליף את הסימול בטקסט המתאים ובכך ליישם Late Bound Data Expressions. למעשה הכוח של הפונקציות האלו שהוא נותן לנו את האפשרות

לפרמט איך שנרצה את הנתונים שאנו מקבלים וכל זאת מתוך תצוגת Design של האפליקציה שלנו.

ד. רק פונקציות (או: When the shit hits the fan we get gold)

נשנה עוד את הטבלה שלנו:

שם הגברת עם סלים	כמות הסלים	מחיר הסלים
strOldLadyWithBags_name	intNumberOfBags	intFinalCost
לאה	**0**	**0**
null	8	2,000
סימה	4	**0**

השינויים: לאה אומנם גברת אבל בלי סלים ואיזה משתמש גאון החליט להוסיף אותה לרשימה , וסימה בכלל ביצעה הימלטות נועזת מקוסמוס בלי לשלם.

אם נריץ את הדוגמה לעיל עם הנתונים האלו נקבל: "לאה עלתה לאוטובוס עם 0 סלים במחיר 0₪.גברת ללא שם עלתה לאוטובוס עם 8 סלים במחיר 2,000₪. סימה עלתה לאוטובס עם 4 סלים במחיר 0₪."

למרות שהכל מודפס ויחסית ברור, הנתונים עצמם נראים מגוחך. מה זה 0 סלים של לאה? ללאה אין סלים! מה זה מחיר 0 ש"ח? זה חינם! ככה אנחנו רוצים שיראה בסוף המשפט שלנו: "לאה עלתה לאוטובוס בלי סלים. גברת ללא שם עלתה לאוטובוס עם 8 סלים במחיר 2,000₪. סימה עלתה לאוטובס עם 4 סלים בחינם."

אין לנו אפשרות להמשיך יותר הלאה עם String.Format אם אנחנו רוצים שהמשפטים שלנו יהיו הגיוניים עם תחביר נורמלי.

ועכשיו שאלה, עד כה עבדנו עם String.Format ו-DataBinder.Eval שהן סה"כ פונקציות שמקבלות נתונים ומחזירות מחרוזת. בואו גם אנחנו נבנה אחת כזאתי במיוחד לדוגמה הזאת!

הפונקציה שלנו תקבל: שם הגיברת עם סלים, כמות הסלים ומחיר הסלים ותחזיר מחרוזת לפי הדוגמאות למעלה. היא גם תשתמש בפונקציה מהדוגמה הקודמת.

```
using System.Text;
...
namespace myProject
{
  public class myPage : System.Web.UI.Page
  {
```

```
  ...
  public string FormatGivrat(object strGivartName, object
intSalimAmount, object intPrice)
  {
    // Use a stringBuilder
    StringBuilder mySB = new StringBuilder();

    // Add GivartName to mySB
    mySB.Append(FormatGivratName(strGivartName));
    mySB.Append(“ עלתה לאוטובוס“);

    // Check if Givart Has salim
    if ((intSalimAmount == null) || (intSalimAmount.ToString() == “0“))
    { // Givart Doesn't have Salim
      mySB.Append(“ סלים ללא.“);
    }
    else
    { // Givart Has Salim
      mySB.Append(“ עם “ +  intSalimAmount.ToString() + “ סלים“);

      // Check If there is a Price
      if ((intPrice == null) || (intPrice.ToString() == “0“))
      { // No price
        mySB.Append(” בחינם.”);
      }
      else
      { // there is a price
        mySB.Append(“ במחיר “);
        // Add intPrice and format it to Currency
        mySB.Append( String.Format( “{0:c}“, intPrice.ToString() ) );
      }
    }

   // Return result from StringBuilder
   return mySB.ToString();
  }

  public string FormatGivratName(object GivartName)
  {
    // Check if Givart Im Salim has name
    if (GivartName == null)
    {  // If Givart Im Salim has no name
      return “שם גברת ללא“;
    }
    else
    { // Givart Im Salim has name
```

```
        return GivartName.ToString();
    }
  }
 }
}
```

נקרא לפונקציה מתוך הקוד בתצוגת Design:

```
<asp:DataList id="myDataList" runat="server">
  <ItemTemplate>
    <%# FormatGivrat(
      ((DataRowView)Container.DataItem)
["strOldLadyWithBags_name"]),
      ((DataRowView)Container.DataItem) ["intNumberOfBags"]),
      ((DataRowView)Container.DataItem) ["intFinalCost"]) )   %>
  </ItemTemplate>
</asp:DataList>
```

והתוצאה עכשיו אכן תהיה: "לאה עלתה לאוטובוס בלי סלים. עלתה לאוטובוס עם 8 סלים במחיר 2,000₪. סימה עלתה לאוטובס עם 4 סלים בחינם."

הכוח שקיבלנו עכשיו זה להפסיק את התלות הנמשכת בפונקציות כמו String.Format ו-DataBinder.Eval ואם אנו צריכים ליישם חוקיות משלנו מסיבות של תחביר או תצוגה או כל דבר, נוכל לעשות זאת מתוך תצוגת Design.

אחרית דבר

במאמר זה סקרנו את מגוון האפשרויות הקיימות לפירמוט מחרוזות מתוך תצוגת Design. התחלנו מלעבוד על הפונקציה DataBinder.Eval הבסיסית שכולם מכירים. עברנו ל-String.Format שמאוד דומה (ולמעשה גם היה אפשר לעשות עם DataBinder.Eval את כל מה שעשינו עם String.Format). אחר כך הראנו מצב שבו עיצוב סטטי רגיל והצבת ערכים פשוט לא מתאימים. אצלנו זה היה מהסיבה של תחביר עברי, אבל יש עוד מגוון סיבות שהעתקה של הנתונים אחד-על-אחד לא מתאימים וצריך לשלב חוקיות משלנו. אחר כך הראנו כיצד לעבוד אם כל הביטוי עצמו בתוך <itemTemplete> צריך חוקיות משלו.

כמו כל דבר בדוט נט, כל אפשרות היא האפשרות הנכונה אבל צריך לבחור בקפידה את הכלי שבו נחליט לגשת לבעיה. אישית: אם מדובר בהצבת נתונים רגילה הייתי משתמש ב-DataBinder.Eval. אם מדובר בהצבת נתונים לביטוי שמכיל יותר מנתון אחד הייתי משתמש ב-String.Format. אם יש כמה נתונים שצריכים פירמוט שלא בנוי במערכת הייתי משתמש בפונקציות המיוחדות בשילוב עם String.Format. אם כל הביטוי צריך להשתנות לפי תוכן הנתונים הייתי משתמש בפונקציה מיוחדת. הכלי המתאים לעבודה הנכונה זה הכלל הכי חשוב.