קונפיגורציה לאפליקציות בקובץ ה-web.config, מדוע וכיצד?

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/

קדם דבר:

שלום לכולם,

במאמר נערוך סקירה על מגוון האפשרויות לשמירה של קונפיגורציה לאפליקציות (Application Configuration) בתוך קובץ ה-web.config.

שתי השאלות הראשונות שנשאל הן: מהי קונפיגורציה אפלקטיבית? ומדוע לשמור ב-web.config?

ישנם נתונים בכל אפליקציה שאין צורך או יכולת לשמור במסד נתונים, למשל: רשימת סוגי מוצרים ואת מחרוזת החיבור למסד הנתונים. נהוג לשמור את מרבית הנתונים מהסוג הזה בקובץ XML חיצוני שיושב באפליקציה. ניצמד לדוגמאות שהעלינו קודם: רשימת סוגי מוצרים ומחרוזת חיבור.

את רשימת סוגי המוצרים במסד NorthWind, אפשר לשמור בקובץ XML בתוך האפליקציה. הרי מדובר על רשימה של מעט פריטים, שמיועדים בעיקר לקריאה, וביצוע שינויים מתרחש לעיתים רחוקות. זה יהיה הגיוני לחלוטין ליצור קובץ XML בתוך האפליקציה לשמור נתונים שעונים לקריטריונים הללו.

למה לשמור את מחרוזת החיבור בקובץ XML חיצוני למשל ולכתוב אותה בגוף הקוד (Hard-coded)? מחר עוברים שרת מסד נתונים, אז מה תעשו? תעברו קובץ קובץ באפליקציה ותשנו את מחרוזת החיבור? זה רק הגיוני לשמור את הנתון הזה במקום אחד כדי להקל על פריסת האפליקציה ועל התחזוקה שלה. סיכמנו שנשמור את מחרוזת החיבור בקובץ XML.

שני הנתונים לדוגמה שהעלנו נשמור בקבצי XML. אבל יש ביניהם הבדל מהותי: אחד הוא מידע\נתון אפלקטיבי (Application Data) והשני הוא קונפיגורציה אפלקטיבית (Application Configuration). מידע אפלקטיבי זה נתונים שדרושים לאפליקציה מבחינת הנתונים אשר היא מכילה. לעומת זאת, קונפיגורציה אפלקטיבית אלו נתונים אשר נדרשים לאפליקציה לפעול מבחינה שבעלדיהם לא תוכל לרוץ. אלו הם הנתונים שנדרשים להפעיל את האפליקציה.

עכשיו בואו נעמוד על ההבדל בין כל קובץ XML לבין קובץ ה-web.config. יש הבדל אחד ויחיד שמעניין אותנו כרגע - כאשר משנים את קובץ ה-web.config כל האפליקציה עושה ריסטרט. הווה אומר, ברגע ששיניתי איזשהו נתון בקובץ האפליקציה מאותחלת בכדי לפעול על פי ה-web.config הכי עדכני.

אם נשמור את רשימת סוגי המוצרים שלנו ב-web.config לאחר כל שינוי ברשימת סוגי המוצרים האפליקציה תאתחל, ולא רק שאין שום צורך בזה - זה גם יפגע בתפקוד השוטף של האפליקציה. לעומת זאת, אם שנינו את מחרוזת החיבור של למסד הנתונים **נרצה** שהאפליקציה תאתחל בכדי שתפעל על השרת\המשתמש הנכון.

החלטנו שאת כל הנתונים של קונפיגורציה אפלקטיבית נרצה לשמור ב-web.config. עכשיו נסקור את הדרכים האפשריות לעשות זאת. חשוב לציין שהסקירה היא ברמת קושי הולכת ועולה ובהתאם האפשרויות שנפתחות בפנינו בכל רמה הן גדולות יותר.

אופציה א': appSettings (או "הדרך הקלה, הטיפשה והפשוטה")

לפני שנתחיל להתעסק עם האפשרות הראשונה והשנייה לבצע את זה צריך לסקור מושג יסוד:

key/value pairs (בעברית: זוגות מפתח-ערך): הרעיון והביצוע מאוד פשוטים, ניצור תגיות שיש להן שתי attributes XML. התכונה ראשונה, key, שתשמש אותנו כמפתח ראשי לחפש ערך ספציפי. התכונה השנייה, Value, תקבע ערך לאותו מפתח. ככה למשל יכול להיות לי מפתח בשם "DataBaseConnectionString" אשר ערכו הוא מחרוזת החיבור למסד הנתונים.

כל זוג מפתח-ערך חייב לשבת בתוך תגית XML ראשית אשר מכילה אותו וזוגות מפתחות-ערכים נוספים. ביחס לאותה תגית XML ראשית אפשר לבצע שלוש פעולות: add, remove, clear. כשמו כן הוא add מוסיף מפתח עם ערך לתגית ה-XML. פעולת ה-remove מורידה מפתח מתגית ה-XML. ופעולת clear מורידה את כל המפתחות מתגית ה-XML.

במקום להגיד "תגית XML ראשית" נדבר על אוסף אמיתי - אוסף appSettings. לאוסף זה ניתן להוסיף, להוריד ולנקות ערכים. אוסף זה הוא תגית אפשרית בתוך קובץ ה-web.config. בואו נביט על דוגמה:

```
<configuration>
  <appSettings>
    <add key="DataBaseConnectionString" value="data
source=ServerName; initial catalog= Northwind;" />
  </appSettings>
</configuration>
```

בקוד למעלה הוספנו מפתח DataBaseConnectionString עם הערך: " data source=ServerName; initial catalog= Northwind;". אם נחליט בעוד שעה להתחבר למסד נתונים ערך, כל מה שנצטרך זה לשנות את הערך בתוך web.config. אם מחר עוברים מבנייה בסביבת פיתוח לסביבת ריצה כל מה שנצטרך לעשות זה לשנות את שם השרת בתוך web.config.

כדי לראות את המפתח-ערך שהוספנו צריך להשתמש במרחב השמות System.Configuration.

```
using System.Configuration;
...
private void Page_Load(object sender, System.EventArgs e)
{
```

```
  // We can print out the value from the web.config:
  Response.Write(System.Configuration.ConfigurationSettings.AppSetti
ngs["DataBaseConnectionString"]);
  // shorter way:
  Response.Write(ConfigurationSettings.AppSettings["DataBaseConnec
tionString"]);
  // Or we can set the ConnectionString of our Connection to it:
  SqlConnection conSql = new SqlConnection();
  conSql.ConnectionString =
ConfigurationSettings.AppSettings["DataBaseConnectionString"];
}
```

כמובן שניתן לעשות עוד המון דברים עם appSettings. אפשר להוסיף עוד מפתחות-ערכים וליישם אותם במקומות שונים באפליקציה. כמו בכל אוסף מפתחות-ערכים גם אפשר להוריד (remove) או לנקות (clear) ערכים מתוך האוסף.

אופציה ב': Custom Web.config tags, configSections (או "מאמא מיה Handlerים!")

עבדתם עם appSettings ונהניתם, והיה כיף, והכל פנאן. אבל יום אחד מבקשים ממכם לשמור רשימת נתוני קונפיגורציה אפלקטיבית. למשל, מקרה שקרה לי, היו לאפליקציית אינטרנט שכתבנו שלושה עיצובים אפשריים: אחד לפיתוח, אחד לריצה ואחד לבדיקות. לשם הדוגמה, ההבדל היה צבע הרקע וצבע הפונט. בפיתוח רצינו צבע רקע לבן נעים לעין וצבע פונט שחור, בריצה רצינו צבע רקע כחול וצבע פונט לבן, ובבדיקות רצינו צבע רקע אדום-דם וצבע פונט צהוב כדי שיכאב ל-QAים בעיינים.

איך נכתוב את זה ב-appSetting? אנחנו לא. במקום זה ניצור תגית XML חדשה (Custom XML tags) לכל תגית. אבל במקום להגיד תגית XML חדשה נגיד - Section. נבנה שלושה Section-ים חדשים כ-XML רגיל.

```
<Riza> <!-- קונפיגורציה  לריצה -->
  <add key="BackGroundColor" value="blue" />
  <add key="FontColor" value="white" />
</Riza>
<Pitoch> <!-- קונפיגורציה  לפיתוח -->
  <add key="BackGroundColor" value="white" />
  <add key="FontColor" value="black" />
</Pitoch>
<Testing> <!-- קונפיגורציה  לבדיקות -->
  <add key="BackGroundColor" value="red" />
  <add key="FontColor" value="yellow" />
</Testing>
```

בקוד למעלה הוספנו שלושה Sectionים חדשים והוספנו להם את המפתחות-ערכים המתאימים. נעתיק את הטקסט הזה ל-web.config.

עכשיו כל זה היה קצת פשוט מדי, נכון? אז הגיע הזמן לסבך. אי-אפשר סתם ליצור Sectionים חדשים. לא כותבים Sectionים איך שבא לנו ומקווים שה-web.config ידע לקרוא אותם. לא, צריך להגיד ל-web.config מי בדיוק יטפל בהם. הגדרנו Sectionים חדשים וכעת נקבע מי יטפל בכל אחד מהם. בשביל זה נוסיף את תגית ה-<configSections>. בתוך אותה תגית נצהיר על ה-Sectionים החדשים ונקבע מי יטפל בהם.

```
<configSections>
  <section name="Riza"
type="System.Configuration.NameValueSectionHandler" />
  <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler" />
  <section name="Testing"
type="System.Configuration.NameValueSectionHandler" />
</configSections>
```

הצהרנו על שלושה Section-ים חדשים ואמרנו שמי שיטפל בהם הוא System.Configuration.NameValueSectionHandler. למעשה מה שאמרנו זה "שאנחנו ניגש באפליקציה שלנו ונוציא את ה-Section הזה תחזיר לנו בבקשה NameValueCollection". אנחנו נדבר בהמשך על לקבל סוגים שונים של פקדים. סה"כ קובץ ה-web.config שלנו נראה ככה:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

<configSections>
  <section name="Riza"
type="System.Configuration.NameValueSectionHandler" />
  <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler" />
  <section name="Testing"
type="System.Configuration.NameValueSectionHandler" />
</configSections>

 <Riza> <!-- לריצה  קונפיגורציה --!>
  <add key="BackGroundColor" value="blue" />
  <add key="FontColor" value="white" />
</Riza>
<Pitoch> <!-- קונפיגורציה  לפיתוח --!>
  <add key="BackGroundColor" value="white" />
  <add key="FontColor" value="black" />
```

```
</Pitoch>
<Testing> <!-- קונפיגורציה  לבדיקות -->
  <add key="BackGroundColor" value="red" />
  <add key="FontColor" value="yellow" />
</Testing>
  <system.web>
  ...
 </system.web>
</configuration>
```

ועכשיו, בואו נראה איך ניגשים למידע הזה מתוך האפליקציה שלנו:

```
using System.Configuration;
using System.Collections;
using System.Collections.Specialized;
...
 private void Page_Load(object sender, System.EventArgs e)
  {
  // We Retrive our “Pitoch“ Section
   System.Collections.Specialized.NameValueCollection myNVC =
(System.Collections.Specialized.NameValueCollection)ConfigurationSet
tings.GetConfig("Pitoch");

  // We will print out the Background Color & Font Color
   Response.Write(myNVC.GetValues("BackGroundColor")[0].ToString()
+
                        "<br>" +
                        myNVC.GetValues("FontColor")[0].ToString() +
                        "<br>");

  }
```

נעבור על הדוגמה. דבר ראשון אמרנו אילו NameSpaced צריך כדי שכל הפקדים שלנו יפעלו. ספציפית צריך את System.Collections.Specialized בשביל NameValueCollection. אחר עברנו על ה-NameValueCollection והדפסנו את BackGroundColor ואת- FontColor. ואכן, יודפיס white ו-black.

בקובץ ה-web.config שלנו קבענו כי ה-Sectionים שלנו יחזירו NameValueCollection. נשנה את זה ונגיד שאנו רוצים ש-Riza יחזיר Dictionary.

```
<configSections>
  <section name="Riza"
type="System.Configuration.DictionarySectionHandler" />
```

```
  <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler" />
  <section name="Testing"
type="System.Configuration.NameValueSectionHandler" />
</configSections>
```

וברמת האפליקציה ניגש לנתונים של Riza באופן הבא:

```
using System.Configuration;
using System.Collections;
...
private void Page_Load(object sender, System.EventArgs e)
  {
   // We Retrive our “Riza“ Section
   System.Collections.IDictionary myIDC =
(System.Collections.IDictionary)
ConfigurationSettings.GetConfig("Riza");

   // We will print out the Background Color & Font Color
   Response.Write(myIDC["BackGroundColor"].ToString() +
                          "<Br>" +
                          myIDC["FontColor"].ToString());

 }
```

עד כמה הראנו את הדברים הבאים: איך לעבוד עם מפתחות-ערכים, איך ליצור Section עם מפתחות-ערכים, איך להצהיר ב-configSections על ה-Sectionים החדשים, איך לקבוע אלו פקדים מחזיר כל Section, איך לעבוד עם הפקדים עצמם.

בואו נשים כאן סיכום של מה שלמדנו עד כה:

```
/* Web.config */
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

    <configSections>
       <section name="Riza"
type="System.Configuration.DictionarySectionHandler, System,
Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
       <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral,
```

```
PublicKeyToken=b77a5c561934e089" />
       <section name="Testing"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
    </configSections>


<Riza> <!-- לריצה  קונפיגורציה -->
  <add key="BackGroundColor" value="blue" />
  <add key="FontColor" value="white" />
</Riza>
<Pitoch> <!-- קונפיגורציה  לפיתוח -->
  <add key="BackGroundColor" value="white" />
  <add key="FontColor" value="black" />
</Pitoch>
<Testing> <!-- קונפיגורציה  לבדיקות -->
  <add key="BackGroundColor" value="red" />
  <add key="FontColor" value="yellow" />
</Testing>
  <system.web>
  ....
 </system.web>
</configuration>


/* WebForm */
using System.Configuration;
using System.Collections;
using System.Collections.Specialized;
...
 private void Page_Load(object sender, System.EventArgs e)
  {
  // We Retrive our “Pitoch“ Section
   System.Collections.Specialized.NameValueCollection myNVC =
(System.Collections.Specialized.NameValueCollection)ConfigurationSet
tings.GetConfig("Pitoch");

  // We will print out the Background Color & Font Color
   Response.Write(myNVC.GetValues("BackGroundColor")[0].ToString()
+
                        "<br>" +
                        myNVC.GetValues("FontColor")[0].ToString() +
                        "<br>");
```

```
  // We Retrive our “Riza“ Section
  System.Collections.IDictionary myIDC =
(System.Collections.IDictionary)
ConfigurationSettings.GetConfig("Riza");

  // We will print out the Background Color & Font Color
  Response.Write(myIDC["BackGroundColor"].ToString() +
                        "<Br>" +
                        myIDC["FontColor"].ToString());

 }
```

אופציה ג': sectionGroup (או "ועכשיו כולם ביחד!")

האופציה הזאת היא למעשה הארכה של האופציה הקודמת. אם אתם זוכרים את הדוגמה הקודמת כתבנו לכל סביבת פיתוח section נפרד לחלוטין. עכשיו, למעשה זה לא נכון, יש להם מכנה משותף. כולן סביבות פיתוח. כל section שכזה צריך להיות שייך לאותה ל-sectionGroup. ככה אנחנו רוצים שיראה ה-web.config שלנו:

```
<SvivotPitoch> <!-- קבוצת קונפיגיורציות בשם סביבות פיתוח -->
  <Riza> <!-- קונפיגורציה  לריצה -->
  <add key="BackGroundColor" value="blue" />
    <add key="FontColor" value="white" />
  </Riza>
  <Pitoch> <!-- קונפיגורציה  לפיתוח -->
    <add key="BackGroundColor" value="white" />
    <add key="FontColor" value="black" />
  </Pitoch>
  <Testing> <!-- קונפיגורציה  לבדיקות -->
    <add key="BackGroundColor" value="red" />
    <add key="FontColor" value="yellow" />
  </Testing>
</SvivotPitoch>
```

נכון שזה נראה הרבה יותר מסודר? אולי במצב הזה של שלושה sectionsים זה לא נראה הרבה, אבל תחשבו שאם היינו יוצרים למשל Section לכל שרת שהמסד יכול להתחבר אליו. בארגון בינוני+ מדובר על 20+ שרתים, זה המון Sectionים סתם לזרוק בתוך ה-web.config.

אנחנו כבר למודי web.config ויודעים שאי-אפשר סתם ככה להוסיף Sectionים. באותו אופן גם אי-אפשר להוסיף SectionGroup בלי להצהיר אליה ב-<configSections>.

```
<configSections>
  <sectionGroup name="SvivotPitoch">
    <section name="Riza"
type="System.Configuration.DictionarySectionHandler" />
    <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler" />
    <section name="Testing"
type="System.Configuration.NameValueSectionHandler" />
  </sectionGroup>
</configSections>
```

חשוב להדגיש שינוי גדול בצורת העבודה שלנו - קודם לכן השתמשנו ב-
ConfigurationSettings.GetConfig בכדי להחזיר Section בכך שמסרנו כפרמטר
את שמו. עכשיו אנו נכתוב את הנתיב המלא שצריך לעבור כדי להגיע לאותו
Section.

```
// Previously:
IDictionary myIDC = (IDictionary)
ConfigurationSettings.GetConfig("Riza");

// Now:
IDictionary myIDC = (IDictionary)
ConfigurationSettings.GetConfig("SvivotPitoch/Riza");
```

שימו לב לשינוי, אנחנו למעשה עובדים עם Section בתוך sectionGroup.

נסכם את המצב עד כה:

```
/* Web.config */
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

<configSections>
  <sectionGroup name="SvivotPitoch">
    <section name="Riza"
type="System.Configuration.DictionarySectionHandler, System,
Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
    <section name="Pitoch"
type="System.Configuration.NameValueSectionHandler, System,
```

```
Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
    <section name="Testing"
type="System.Configuration.NameValueSectionHandler, System,
Version=1.0.5000.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
  </sectionGroup>
</configSections>

<SvivotPitoch> <!-- קבוצת קונפיגיורציות בשם סביבות פיתוח --!>
  <Riza> <!-- קונפיגורציה  לריצה --!>
  <add key="BackGroundColor" value="blue" />
    <add key="FontColor" value="white" />
  </Riza>
  <Pitoch> <!-- קונפיגורציה  לפיתוח --!>
    <add key="BackGroundColor" value="white" />
    <add key="FontColor" value="black" />
  </Pitoch>
  <Testing> <!-- קונפיגורציה  לבדיקות --!>
    <add key="BackGroundColor" value="red" />
    <add key="FontColor" value="yellow" />
  </Testing>
</SvivotPitoch>
  <system.web>
  ...
 </system.web>
</configuration>


/* WebForm.aspx */
using System.Configuration;
using System.Collections.Specialized;
...
private void Page_Load(object sender, System.EventArgs e)
  {
   // We Retrive our “Pitoch“ Section
   System.Collections.Specialized.NameValueCollection myNVC
= (System.Collections.Specialized.NameValueCollection) Configuration
Settings.GetConfig("SvivotPitoch/Pitoch");

   // We will print out the Background Color & Font Color
   Response.Write(myNVC.GetValues("BackGroundColor")[0].ToString()
+  "<br>" + myNVC.GetValues("FontColor")[0].ToString() + "<br>");

   // We Retrive our “Riza“ Section
   System.Collections.IDictionary myIDC =
```

```
(System.Collections.IDictionary) ConfigurationSettings.GetConfig("SvivotPitoch/Riza");

  // We will print out the Background Color & Font Color
  Response.Write(myIDC["BackGroundColor"].ToString() +
           "<Br>" + myIDC["FontColor"].ToString());
 }
```

(השינויים מודגשים)

<u>אופציה ד': XmlNode, CustomHandlers (או "סוף סוף נפתרנו מהמפתחות-ערכים האלו!")</u>

הגענו דרך ארוכה עד כאן... פיתחנו בתוך ה-web.config שלנו תגיות שנראות ממש כמו XML והכל במינימום מאמץ. אבל יש עוד שני דברים שמציקים לי:

הראשון, זה שאנחנו עובדים עם מפתחות-ערכים. תכלס', זה כמו XML, אבל זה נראה מוזר. למה לא לעבוד עם XML כמו שהוא אמור להיראות? (לפתוח תגית, טקסט, לסגור תגית)

השני, ברמת האפליקציה אנחנו עובדים עם איזה פקדי-נתונים פקקטא מצחיקים. למה לא לעבוד עם איזה XmlNode גברי וחסון? תכלס' הסיבה היא שכבר התרגלתי לעבוד עם פקדי XPath וזה גם הרבה יותר הגיוני (לפחות בראש שלי) לעבוד עם XPath שאתה מנווט בתוך קוד XML.

מאוד רציתי SectionHandler שיוכל לעשות את שני הדברים האלו. אבל אין בנמצא. אז כתבתי אחד.

```
public class XmlNodeSectionHandler : IConfigurationSectionHandler
{
   public object Create(object parent, object configContext, XmlNode
section)
   {
     return section;
   }
}
```

אני יודע שעכשיו חלקכם תוהים קצת מה בדיוק כתוב למעלה. אתם זוכרים שהגדרנו configSections? כשהגדרנו <section> תמיד גם אמרנו type. התכונה הזאת, type, מתייחסת לאיזה פקד מסוג sectionHandler יטפל בבקשה שלנו ל-ConfigurationSettings.GetConfig. בקוד למעלה כתבתי sectionHandler שממש את הממשק הנדרש ל-SectionHandlerים ומחזיר XmlNode לפי הנתיב שנבקש. קצת מסובך, אני יודע, אבל זה הכי פשוט שיש.

בואו נביט על קובץ ה-web.config שלנו:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

    <configSections>
    <section name="SvivotPitoch"
type="ProjectNameSpace.XmlNodeSectionHandler,
ProjectNameSpace" />
  </configSections>

<SvivotPitoch>
  <Riza>
    <BackGroundColor>blue</BackGroundColor>
    <FontColor>white</FontColor>
  </Riza>
  <Pitoch>
     <BackGroundColor>white</BackGroundColor>
    <FontColor>black</FontColor>
  </Pitoch>
  <Testing>
    <BackGroundColor>red</BackGroundColor>
    <FontColor>yellow</FontColor>
  </Testing>
</SvivotPitoch>
  <system.web>
  ...
 </system.web>
</configuration>
```

תראו איזה יופי, כל כך נקי, כל כך מסודר. סה"כ הצהרנו על Section אחדבלבד וכתבנו אותו ב-XML רגיל לחלוטין. תשימו לב שבהצהרה על ה-Section כתבנו הפנייה לProjectNameSpace.XmlNodeSectionHandler שזו הפנייה ל-SectionHandler שלנו.

עכשיו נביט על הקוד שלנו:

```
using System.Configuration;
using System.Xml;
...
namespace ProjectNameSpace
{
  public class WebForm1 : System.Web.UI.Page
  {
```

```
    private void Page_Load(object sender, System.EventArgs e)
    {
      // Retrive the XMLnode of SvivotPitoch into myXML
      XmlNode myXML = (XmlNode)
ConfigurationSettings.GetConfig("SvivotPitoch");
      // Print the Testing/BackGroundColor
      Response.Write(myXML.SelectNodes("Testing/BackGroundColor")[
0].InnerText);
    }
    ...
  }

  public class XmlNodeSectionHandler : IConfigurationSectionHandler
  {
    public object Create(object parent, object configContext, XmlNode
section)
    {
      return section;
    }
  }
}
```

איזו פשטות, במקום לעבוד עם Dictionary או NameValueCollection, אנחנו עובדים עם XmlNode. ובביטוי XPath קטן הגענו לנתון שחיפשנו. כמובן שעכשיו שיש לנו פקד רציני לעבוד איתו, נפתחות בפנינו אפשרויות נוספות.

אחרית דבר

במהלך מאמר זה הראינו מהם זוגות מפתחות-ערכים, איך לעבוד עם appSettings (שאם תחשבו על זה הוא section), הראנו איך להצהיר על Sectionים חדשים, איך לגשת אליהם מהקוד, איך ליצור SectionGroups, על השינויים שצריך בקוד, ולבסוף איך לעשות חצי ממה שעשינו קודם בשביל פי שניים כוח. תחשבו על זה.

כמו בכל אפשרות בדוט נט הטריק הוא לזהות מתי צריך להשתמש בה. בהצלחה!