גירסה 1.00 – 22.11.2003

# כתיבת Web Applications בעזרת C#

ניר אדר



Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

אנא שלחו תיקונים והערות אל המחבר.

# כתיבת Web Applications בעזרת C#

## התחלת העבודה

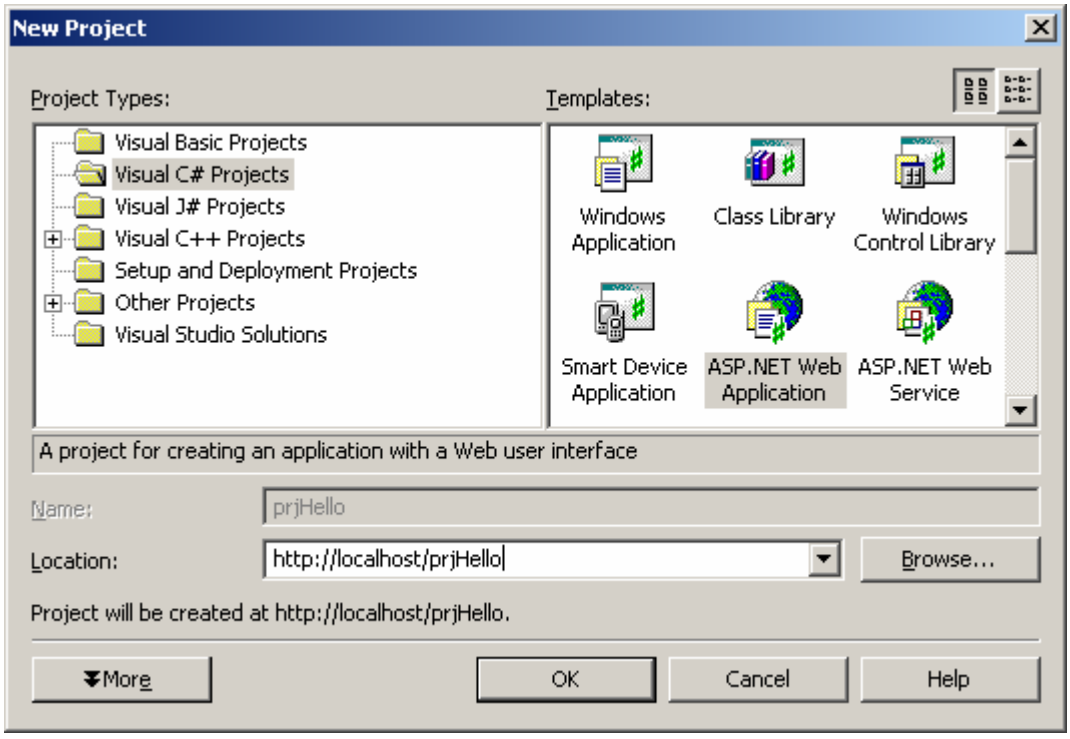
כדי להתחיל Web Application נבחר ASP.Net Web Application מתפריט New Project.

ייפתח טופס אליו בדומה לתכנות Windows Application אנחנו מסוגלים לגרור Controls ולעצבם כרצוננו. כמו כן אפשר לערוך את דף ה-HTML ישירות כדי לקבל את המראה הרצוי.
ב-ASP.Net יש הפרדה בין קטע הקוד האחראי להצגת ממשק המשתמש לבין קטע הקוד האחראי לטפל בתפקודו של הממשק. עיצוב הדף נעשה בשפת HTML ואילו קידוד הפונקציונאליות שלו נעשה ב-C#.

לכל ה-Controls שאנו משתמשים בהם שני מאפיינים שחוזרים על עצמם: ID שהוא שם ה-Control בו אנחנו משתמשים בקוד, ו-Text שהוא הטקסט שיופיע עליו.

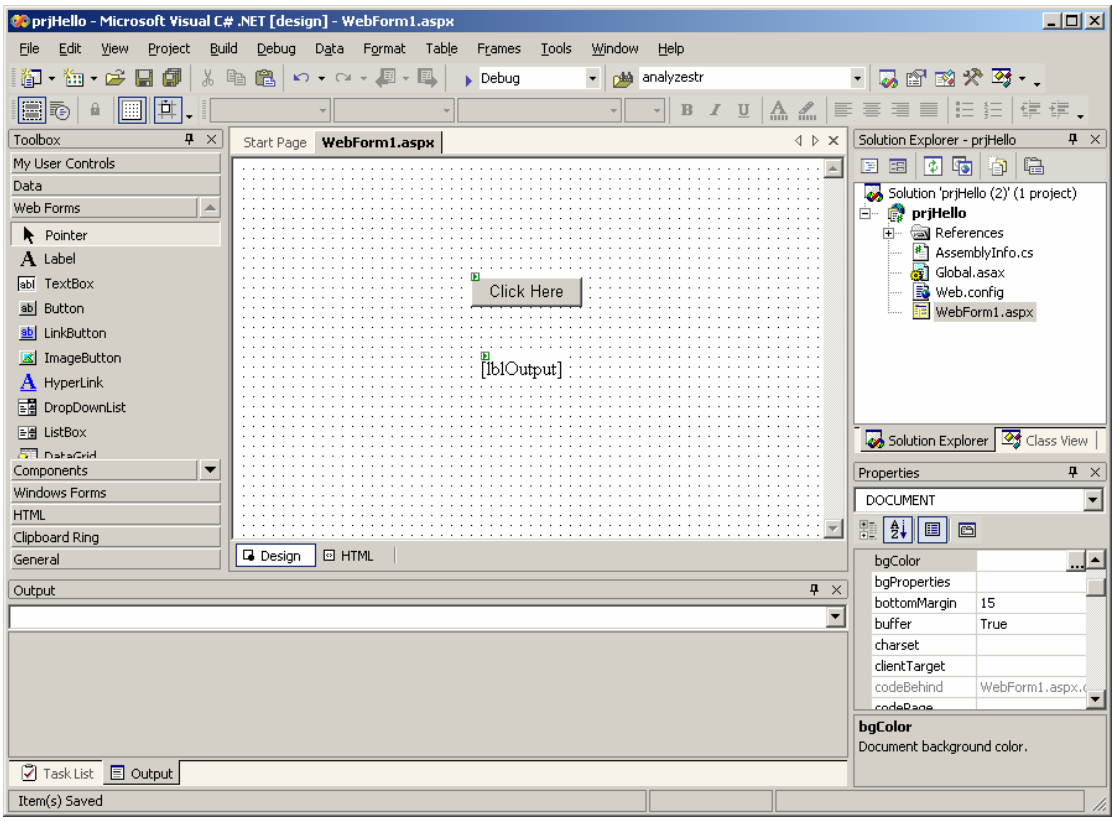
## דוגמא בסיסית

נפתח פרויקט חדש בשם prjHello.



נגרור Button ו-Label לטופס שייפתח, ונשנה את שמם ל-btnDoPrint ו-lblOutput בהתאמה.

נקבע את הטקסט של ה-label להיות טקסט ריק, ואת זה של ה-button להיות "Click Here".




לאחר מכן נלחץ לחיצה כפולה על הכפתור, כדי ש-C# ייצור את הקוד עבור האירוע לחיצה על הכפתור, וייפתח את הפונקציה המתאימה.

נכתוב את הקוד הבא:

```
private void btnDoPrint_Click(object sender, System.EventArgs e)
{
        lblOutput.Text = "Hello, World!";
}
```

כעת, כאשר נקמפל ונריץ את התוכנית, אם נלחץ על הכפתור, המילים "Hello, World!" יופיעו בתוך ה-label.

## התקנה של אפליקצית ASP.Net

לאחר שכתבנו אפליקציה, נרצה להתקין אותה על השרת.

התקנת אפליקצית ASP.Net תעשה בצורה הבאה:

1. ניצור ספריה אליה נעתיק את התוכנית שלנו, למשל C:\MyWebApplication.
   אל ספריה זו נעתיק את כל קבצי ה-aspx, את הקובץ Web.Config, את הקובץ Global.asax, ובמידה וקיימים קבצי Resource או תמונה השייכים לאפליקציה שלנו נעתיק גם אותם.
2. ניצור תת ספריה בשם bin ואליה נעתיק את קובץ ה-DLL שיצרנו.
3. ניצור Virtual Directory שתמופה אל הספרייה שלנו דרך מסכי הניהול של IIS.

## Global.asax

קובץ חשוב של ASP.Net הינו Global.asax. הקובץ הזה מאפשר לנו לבצע פעולות מיוחדות כאשר התוכנית שלנו מתחילה, או כאשר משתמש חדש מתחיל להשתמש בתוכנית.

נביט בקוד של קובץ זה:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Web;
using System.Web.SessionState;

namespace prjHello
{
	/// <summary>
	/// Summary description for Global.
	/// </summary>
	public class Global : System.Web.HttpApplication
	{
		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.IContainer components = null;

		public Global()
		{
			InitializeComponent();
		}

		protected void Application_Start(Object sender, EventArgs e)
		{

		}

		protected void Session_Start(Object sender, EventArgs e)
		{

		}

		protected void Application_BeginRequest(Object sender,
                                                EventArgs e)
		{

		}

		protected void Application_EndRequest(Object sender, EventArgs
                                                e)
		{

		}

		protected void Application_AuthenticateRequest(Object sender,
                                                        EventArgs e)
		{

		}
```

```
        protected void Application_Error(Object sender, EventArgs e)
        {

        }

        protected void Session_End(Object sender, EventArgs e)
        {

        }

        protected void Application_End(Object sender, EventArgs e)
        {

        }

        #region Web Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
        }
        #endregion
    }
}
```

האירועים החשובים הם Application_Start, Application_End, Session_Start, Session_End.

בדומה להתנהגותם של אירועים אלו בשפת ASP, Application_Start נקרא פעם אחת בכל חיי אפליקצית ה-Web שלנו, כאשר התוכנית מתחילה. Application_End נקרא פעם אחת בסיום התוכנית.

עבור כל משתמש חדש שנכנס לאתר ומתחיל להשתמש בתוכנית, נקרא האירוע Session_Start. כאשר פג תוקף שהיית המשתמש, נקרא האירוע Session_End.

## המחלקה Page

המחלקה Page, ממנה נגזר הדף אותו אנחנו מעצבים, כוללת מספר תכונות ואירועים החשובים לנו.

**מאפיינים עיקריים:**

| תכונה | תיאור |
|---|---|
| Application | מאפשר לשמור אינפורמציה שתהיה משותפת לכל המשתמשים |
| IsPostBack | נותן אינדיקציה אם הדף נטען בפעם הראשונה, או שהוא נטען בתגובה לבקשת משתמש |
| Request | מייצג את האובייקט Request הנותן אינפורמציה לגבי בקשות http. |
| Response | מייצג את האובייקט Response המאפשר שליחת מידע אל ה-Web Browser. |
| Server | מייצג את האובייקט HttpServerUtility.<br>אובייקט זה מאפשר לנו לבצע פעולות מגוונות, כגון לעבור לדף אחר, להריץ דף אחר ולהחזיר את התוצאה שלו, לנתח מחרוזות המכילות כתובות ועוד. |
| Session | מאפשר לשמור מידע אישי עבור כל משתמש בתוכנית |

**אירועים חשובים**

| אירוע | תיאור |
|---|---|
| Init | אירוע זה קורה עם אתחול הדף. הבקרים אינם קיימים בשלב זה. |
| Load | אירוע זה קורה לאחר שהבקרים קיימים ומאותחלים. אתחול נוסף נעשה כאן. |
| UnLoad | אירוע זה קורה רגע לפני שהדף יורד מהזיכרון. הוא מיועד לפעולות ניקוי שונות. |

## IsPostBack

המאפיין IsPostBack אומר האם הדף נטען בפעם הראשונה או שהוא נטען כתוצאה מ-PostBack של המשתמש.

כדי להבין את החשיבות של מאפיין זה, ניצור את תוכנית הדוגמא הבאה:

ניצור טופס ריק ונגרור לתוכו רשימה אחת. נקבע את שמה ל-lstSample.

כמו כן נוסיף כפתור (ללא קוד מאחוריו).

ב-Page_Load נרשום את הקוד הבא:

```
private void Page_Load(object sender, System.EventArgs e)
{
        lstSample.Items.Add("Item 1");
        lstSample.Items.Add("Item 2");
        lstSample.Items.Add("Item 3");
}
```

כאשר הדף יעלה, יהיו ברשימה שלושה איברים. כעת נלחץ על הכפתור. המידע נשלח לשרת, והדף יעלה שוב בצד הלקוח.

הנתונים שהיו ברשימה קודם נשמרו, אולם הדף נטען שוב ולכן Page_Load נקראת שוב.

התוצאה: ברשימה יהיו שישה איברים לאחר שנלחץ על הכפתור.

על מנת לפתור את הבעיה, נשתמש במאפיין IsPostBack. נכתוב את הקוד הבא:

```
private void Page_Load(object sender, System.EventArgs e)
{
        if (!IsPostBack)
        {
                lstSample.Items.Add("Item 1");
                lstSample.Items.Add("Item 2");
                lstSample.Items.Add("Item 3");
        }
}
```

## Application

Application הוא אובייקט המאפשר לנו לשמור מידע שיהיה משותף לכל המשתמשים בתוכנית.
ניתן להגדיר מעין "משתנים גלובליים" עבור התוכנית על ידי Application.

נכתוב תוכנית שתשמור במשתנה Application את מספר האנשים שהשתמשו בתוכנית.
נפתח את הקובץ global.asax ונכתוב בו את הקוד הבא:

```
protected void Application_Start(Object sender, EventArgs e)
{
        Application["Counter"] = 0;
}

protected void Session_Start(Object sender, EventArgs e)
{
        Application.Lock();
        Application["Counter"] = (int)Application["Counter"] + 1;
        Application.UnLock();
}
```

כאשר התחלנו את המשתנה Application["Counter"] בפעם הראשונה יצרנו אותו למעשה.
אין יצירה מפורשת של משתני Application.
כמו כן, נשים לב לשימוש בפונקציות Application.Lock() ו-Application.UnLock().
מכיוון שמספר משתמשים עשויים להתחיל להשתמש בתוכנית שלנו בו זמנית אנחנו צריכים לממש מנגנון של מניעה הדדית, כאשר אנחנו משנים משתנים המשותפים להם. המניעה ההדדית ממומשת בעזרת פונקציות אלו.

## Session

בצורה דומה למשתני Application, משתני Session מאפשרים לנו לשמור מידע עבור כל משתמש.
למשל, אם נרצה לשמור את שם המשתמש שהמשתמש הכניס לנו, כך שילווה אותנו לאורך כל העבודה מול אותו משתמש, נוכל לעשות זאת בצורה הבאה:

```
Session["UserName"] = txtUserName.Text;
```

## בקרים ב-Web Application

Net. מחלק את הבקרים בהם ניתן להשתמש ב-WebForms ל-4 קטגוריות:

- HTML Controls
- WebForm Controls
- Validation Controls
- Data Controls

### HTML Controls

HTML Controls הינם בקרים המייצגים באופן ישיר אלמנטים בשפת HTML.
לרוב לא נשתמש בהם ונעדיף להשתמש ב-WebForm Controls. בקרים אלו מצויים בעיקר לשם תמיכה לאחור עם ASP3, בהם היה ניתן להשתמש רק בבקרים אלו.
בניקוד ל-WebForm Controls, הרצים כברירת המחדל בצד השרת, HTML Controls רצים כברירת מחדל בצד הלקוח (למרות שניתן לשנות מצב זה).

### WebForm Controls

WebForm Controls הם בקרים הדומים לבקרים בהם אנו משתמשים ב-WinForms, אם כי בעלי תכונות מצומצמות מעט. בקרי WebForm מורכבים לרוב ממספר רכיבי HTML ולא מאלמנטים בודדים. לרוב נשתמש בבקרים אלו כאשר נעצב את האפליקציה שלנו.

### Validation Controls

Validation Controls הינם בקרים המאפשרים לנו לבצע בדיקה של ערכי Input עבור בקרים אחרים.
הטבלה הבאה תיתן לנו תקציר של הבקרים השונים מסוג זה הקיימים ב-ASP.Net:

| תיאור | בקר |
|---|---|
| מחייב שהשדה יהיה מלא בערך כלשהו | RequiredFieldValidator |
| בודק את טווח הערכים של הקלט | RangeValidator |
| מבצע השוואה בין הקלט לערך כלשהו | CompareValidator |
| מבצע בדיקה באמצעות ביטוי רגולארי | RegularExpressionValidator |

השימוש בבקרים אלו נעשה בצורה גראפית בסביבת העבודה על ידי גרירת הבקר המתאים לטופס, וקביעת ControlToValidate בתור הבקר שאת תוכנו אנו רוצים לבדוק. לאחר מכן, בהתאם לכל בקר בדיקה, אנו מגדירים את הבדיקה הרצויה.

### Data Controls

בקרים אלו מספקים כלים וממשק לעבודה מול בסיסי נתונים.

## Smart Navigation

תכונה שימושית הקיימת ב-ASP.Net הינה Smart Navigation.
בדרך כלל כאשר מבצעים PostBack נטען מחדש כל דף ה-HTML, והרענון נראה לעין. כמו כן, פעמים רבות במקרה זה הפוקוס עובר אל תחילת הדף, ובמידה ואנחנו היינו בסופו, עלינו לגלול שוב על מנת להגיע לסופו.
Internet Explorer בגירסה 5 ומעלה תומך בתכונה הנקראת Smart Navigation. אם אנחנו משתמשים באפשרות זו, אזי בתגובה ל-PostBack מתרענן רק החלק שהשתנה בדף, והפוקוס בו נשאר ללא שינוי. תכונה זו נותנת הרגשה הרבה יותר נוחה למשתמש בתוכניות.
כדי לנצל את התכונה כל שעלינו לעשות הוא לשנות מעט את שורת הכותרת של דף ה-ASPX, ולהוסיף אליו את תכונת ה-SmartNavigation, בצורה הבאה:

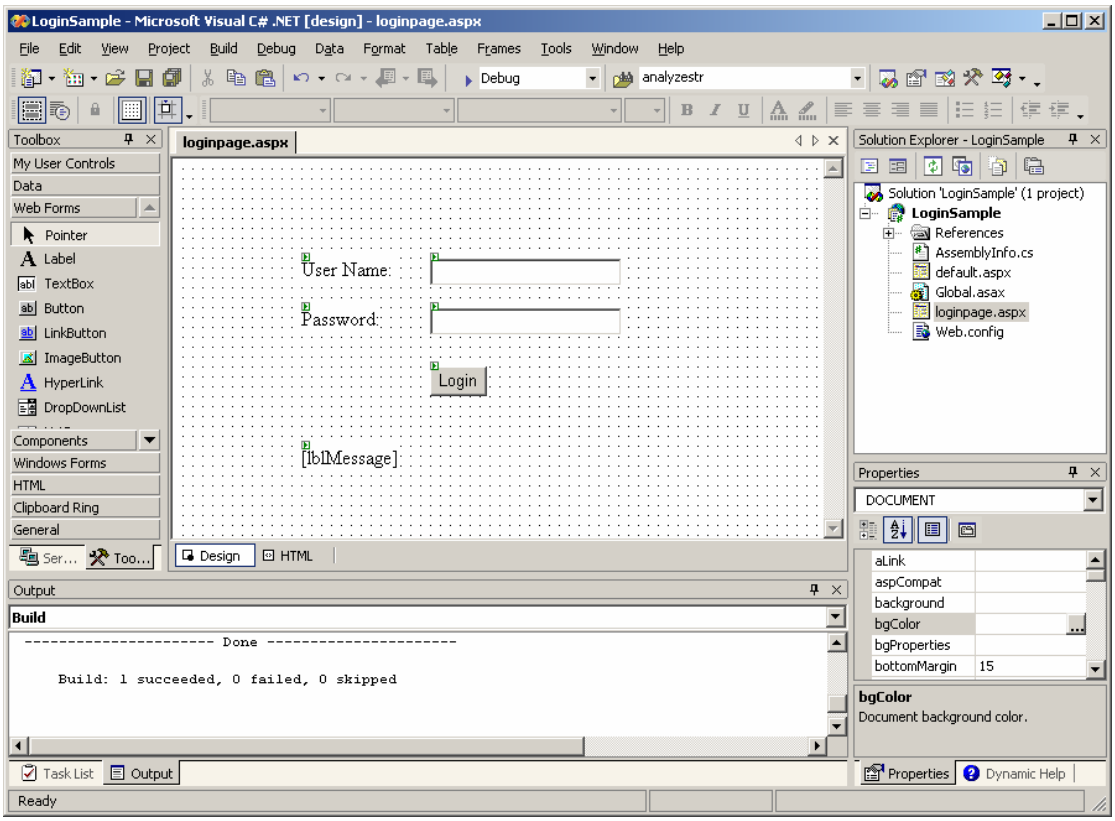
```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs" AutoEventWireup="false"
Inherits="UserValidation.WebForm1" SmartNavigation = "True" %>
```

## אבטחת טפסים

ASP.Net מכיל מספר מנגנוני אבטחה מובנים. נציג כרגע אחד מהם – אבטחה באמצעות טופס. הרעיון – כאשר המשתמש מתחיל להשתמש בתוכנית שלנו הוא נדרש לשם וסיסמא. המשתמש יורשה להשתמש בתוכנית שלנו רק במידה והסיסמא שהקיש נכונה. כמו כן, אם המשתמש ינסה לקפוץ אל אחד מדפי האפליקציה בלי להזדהות תחילה, נרצה שיופיע מולו מסך הפתיחה וידרוש ממנו את הסיסמא.

ביצוע משימה זו יורכב משני חלקים. החלק הראשון יהיה לבנות טופס שיוודא את שמו וסיסמתו של המשתמש. החלק השני יהיה שינוי הקובץ Web.Config כדי לגרום לאפליקציה להשתמש בטופס זה.

נראה את שיטת אבטחה זו בעזרת דוגמא. ניצור תוכנית Web חדשה, וניצור בה דף בשם loginpage. אל דף זה נגרור שלוש תוויות, שתי תיבות טקסט וכפתור.
לאחת מתיבות הטקסט נקרא txtUserName ולשנייה נקרא txtPassword. לכפתור נקרא btnDoLogin ולאחת משלושת התוויות נקרא lblMessage ונקבע את הטקסט שלה להיות מחרוזת ריקה. נעצב את הדף עד שיראה כך:



הקוד שמפעיל btnDoLogin ייראה כך:

```
private void btnDoLogin_Click(object sender, System.EventArgs e)
{
        if (txtUserName.Text == "nir" && txtPassword.Text == "123")
        {
                FormsAuthentication.RedirectFromLoginPage(txtUserName.Text, true);
        }
        else
        {
                lblMessage.Text = "Invalid Username or password";
        }
}
```

כמו כן נוסיף את ההצהרה הבאה בראש הדף שלנו, על מנת שנוכל להשתמש במחלקה FormsAuthentication:

```
using System.Web.Security;
```

הקוד של דף זה הוא קוד פשוט. הקוד יאפשר גישה רק אם שם המשתמש הוא "nir" והסיסמא היא "123". במידה ונכתוב מערכת אמיתית שתבוסס על טופס נשתמש לרוב בבסיס נתונים שהשם והסיסמא יושוו מולו.

במידה וזיהוי המשתמש הצליח אנו משתמשים בשיטה RedirectFromLoginPage. שיטה זו מקבלת את שם המשתמש, ויוצרת cookie אצל המשתמש כדי לציין שההזדהות הצליחה. כמו כן היא מחזירה את המשתמש אל הדף שהוא היה בו לפני ההזדהות (בדרך כלל זהו מסך הפתיחה של התוכנית).

לאחר מכן נלך ונערוך את הקובץ Web.Config על מנת להגיד ל-ASP.Net להשתמש בטופס שיצרנו. נמצא את השורות הבאות:

```
    <authentication mode="Windows" />

        <!--  AUTHORIZATION
          This section sets the authorization policies of the application. You can
          allow or deny access to application resources by user or role. Wildcards:
          "*" mean everyone, "?" means anonymous (unauthenticated) users.
    -->

    <authorization>
        <allow users="*" /> <!-- Allow all users -->
            <!--  <allow     users="[comma separated list of users]"
                             roles="[comma separated list of roles]"/>
                  <deny      users="[comma separated list of users]"
                             roles="[comma separated list of roles]"/>
            -->
    </authorization>
```

שורות אלו מאפשרות לכל משתמש להיכנס ולהשתמש בתוכנית שלנו ללא כל סיסמא. נחליף אותן בשורות הבאות, שיאפשרו רק למשתמשים שהזדהו להשתמש בתוכנית:

```
    <authentication mode="Forms">
                <forms name="logincookie" loginUrl="loginpage.aspx" protection="All"
timeout="30" />
    </authentication>

        <!--  AUTHORIZATION
          This section sets the authorization policies of the application. You can
          allow or deny access to application resources by user or role. Wildcards:
          "*" mean everyone, "?" means anonymous (unauthenticated) users.
    -->

    <authorization>
                <deny users="?" />
    </authorization>
```

חשוב לזכור לשנות שורות אלו. אם לא נשנה אותן, כל משתמש יוכל להשתמש בתוכנית שלנו ללא כל הזדהות.

דגש נוסף הוא ששיטת אבטחה זו קיימת לפי שעה רק במערכות הבאות: Windows 2000, Windows XP Professional, Windows Server 2003.