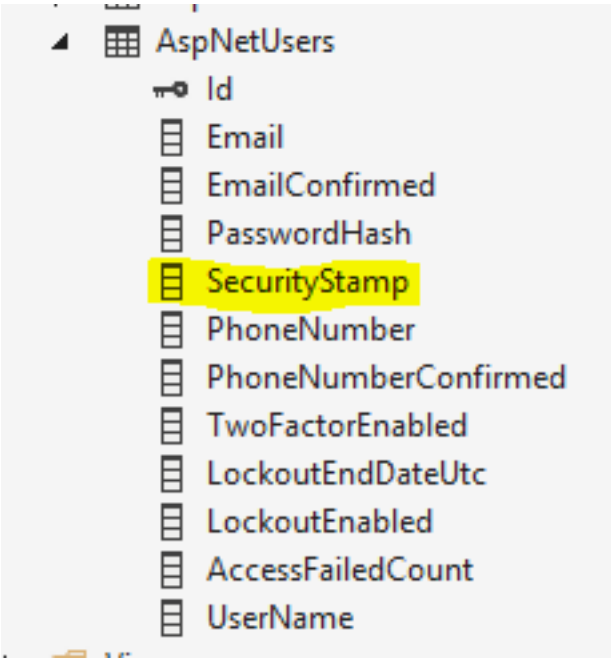
عنوان: **جلوگیری از ورود همزمان کاربران با نام کاربری و رمز عبور یکسان**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۴/۰۶/۱۲ ۱۶:۱۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, ASP.NET Identity

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:

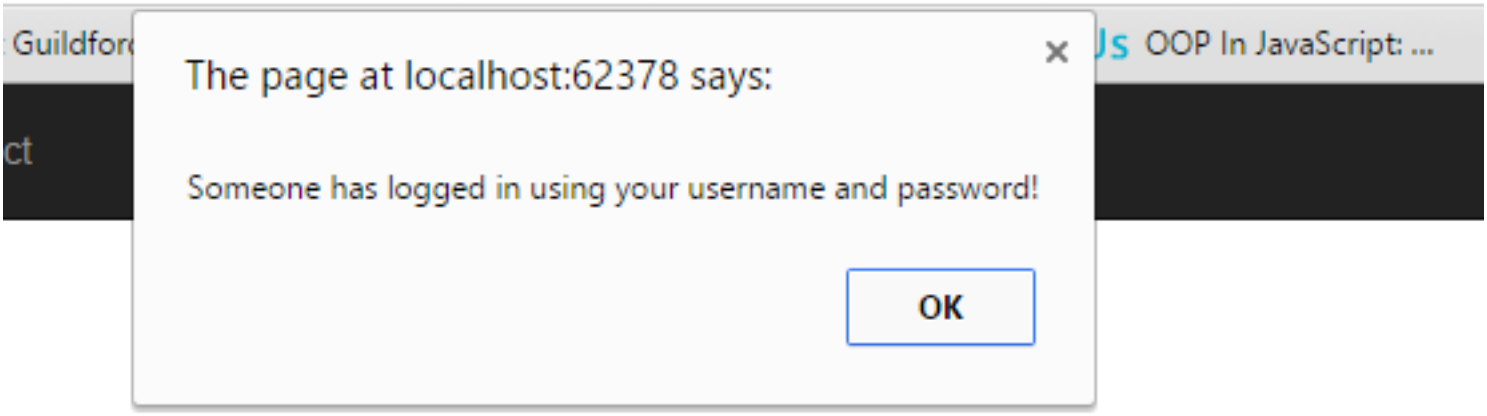
```
window.setInterval(function() {
            $.ajax({
                url:,
                type: "Post",
                dataType: "json",
                success: function(data) {
                    if (!data) {
                        alert("Someone has logged in using your username and password!");
                        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();
```