عنوان: ASP.NET MVC #17
نویسنده: وحید نصیری
تاریخ: ۰۸:۲۵:۰۰ ۱۳۹۱/۰۱/۲۹
آدرس: www.dotnettips.info
گروه‌ها: MVC

فیلترهای امنیتی ASP.NET MVC

ASP.NET MVC به همراه تعدادی فیلتر امنیتی توکار است که در این قسمت به بررسی آن‌ها خواهیم پرداخت.

بررسی اعتبار درخواست (Request Validation) در ASP.NET MVC

ASP.NET MVC امکان ارسال اطلاعاتی را که دارای تگ‌های HTML باشند، نمی‌دهد. این قابلیت به صورت پیش فرض فعال است و جلوی ارسال انواع و اقسام اطلاعاتی که ممکن است سبب بروز حملات XSS Cross site scripting attacks شود را می‌گیرد. نمونه‌ای از خطای نمایش داده:

```
A potentially dangerous Request.Form value was detected from the client (Html="<a>").
```

بنابراین تصمیم گرفته شده صحیح است؛ اما ممکن است در قسمتی از سایت نیاز باشد تا کاربران از یک ویرایشگر متنی پیشرفته استفاده کنند. خروجی این نوع ویرایشگرها هم HTML است. در این حالت می‌توان صرفا برای متدی خاص امکانات Request Validation را به کمک ویژگی ValidateInput غیرفعال کرد:

```
[HttpPost]
[ValidateInput(false)]
public ActionResult CreateBlogPost(BlogPost post)
```

از ASP.NET MVC 3.0 به بعد راه حل بهتری به کمک ویژگی AllowHtml معرفی شده است. غیرفعال کردن ValidateInput ایی که معرفی شد، بر روی تمام خواص شیء BlogPost اعمال می‌شود. اما اگر فقط بخواهیم که مثلا خاصیت Text آن از مکانیزم بررسی اعتبار درخواست خارج شود، بهتر است دیگر از ویژگی ValidateInput استفاده نشده و به نحو زیر عمل گردد:

```
using System;
using System.Web.Mvc;

namespace MvcApplication14.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public DateTime AddDate { set; get; }
        public string Title { set; get; }

        [AllowHtml]
        public string Text { set; get; }
    }
}
```

در اینجا فقط خاصیت Text مجاز به دریافت محتوای HTML ایی خواهد بود. اما خاصیت Title چنین مجوزی را ندارد. همچنین دیگر

نیازی به استفاده از ویژگی ValidateInput غیرفعال شده نیز نخواهد بود.
به علاوه همانطور که در قسمت‌های قبل نیز ذکر شد، خروجی Razor به صورت پیش فرض Html encoded است مگر اینکه صریحا آن‌را تبدیل به HTML کنیم (مثلا استفاده از متد Html.Raw). به عبارتی خروجی Razor در حالت پیش فرض در مقابل حملات XSS مقاوم است مگر اینکه آگاهانه بخواهیم آن‌را غیرفعال کنیم.

مطلب تکمیلی
مقابله با XSS ؛ یکبار برای همیشه!

فیلتر RequireHttps

به کمک ویژگی یا فیلتر RequireHttps، تمام درخواست‌های رسیده به یک متد خاص باید از طریق HTTPS انجام شوند و حتی اگر شخصی سعی به استفاده از پروتکل HTTP معمولی کند، به صورت خودکار به HTTPS هدایت خواهد شد:

```
[RequireHttps]
public ActionResult LogOn()
{
}
```

فیلتر ValidateAntiForgeryToken

نوع دیگری از حملات که باید در برنامه‌های وب به آن‌ها دقت داشت به نام CSRF یا Cross site request forgery معروف هستند. برای مثال فرض کنید کاربری قبل از اینکه بتواند در سایت شما کار خاصی را انجام دهد، نیاز به اعتبار سنجی داشته باشد. پس از لاگین شخص و ایجاد کوکی و سشن معتبر، همین شخص به سایت دیگری مراجعه می‌کند که در آن مهاجمی بر اساس وضعیت جاری اعتبار سنجی او مثلا لینک حذف کاربری یا افزودن اطلاعات جدیدی را به برنامه ارائه می‌دهد. چون سشن شخص و کوکی مرتبط به سایت اول هنوز معتبر هستند و شخص سایت را نبسته است، «احتمال» اجرا شدن درخواست مهاجم بالا است (خصوصا اگر از مرورگرهای قدیمی استفاده کند).
بنابراین نیاز است بررسی شود آیا درخواست رسیده واقعا از طریق فرم‌های برنامه ما صادر شده است یا اینکه شخصی از طریق سایت دیگری اقدام به جعل درخواست‌ها کرده است.
برای مقابله با این نوع خطاها ابتدا باید داخل فرم‌های برنامه از متد Html.AntiForgeryToken استفاده کرد. کار این متد ایجاد یک فیلد مخفی با مقداری منحصربفرد بر اساس اطلاعات سشن جاری کاربر است، به علاوه ارسال یک کوکی خودکار تا بتوان از تطابق اطلاعات اطمینان حاصل کرد:

```
@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
```

در مرحله بعد باید فیلتر ValidateAntiForgeryToken جهت بررسی مقدار token دریافتی به متد ثبت اطلاعات اضافه شود:

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult CreateBlogPost(BlogPost post)
```

در اینجا مقدار دریافتی از فیلد مخفی فرم :

```
<input name="__RequestVerificationToken" type="hidden" value="C0iPfy/3T....=" />
```

با مقدار موجود در کوکی سایت بررسی و تطابق داده خواهند شد. اگر این مقادیر تطابق نداشته باشند، یک استثنا صادر شده و از پردازش اطلاعات رسیده جلوگیری می‌شود.
علاوه بر این‌ها بهتر است حین استفاده از متد و فیلتر یاد شده، از یک salt مجزا نیز به ازای هر فرم، استفاده شود:

```
[ValidateAntiForgeryToken(Salt="1234")]
@Html.AntiForgeryToken(salt:"1234")
```

به این ترتیب tokenهای تولید شده در فرم‌های مختلف سایت یکسان نخواهند بود.
به علاوه باید دقت داشت که ValidateAntiForgeryToken فقط با فعال بودن کوکی‌ها در مرورگر کاربر کار می‌کند و اگر کاربری پذیرش کوکی‌ها را غیرفعال کرده باشد، قادر به ارسال اطلاعاتی به برنامه نخواهد بود. همچنین این فیلتر تنها در حالت HttpPost قابل استفاده است. این مورد هم در قسمت‌های قبل تاکید گردید که برای مثال بهتر است بجای داشتن لینک delete در برنامه که با HttpGet ساده کار می‌کند، آن‌را تبدیل به HttpPost نمود تا میزان امنیت برنامه بهبود یابد. از HttpGet فقط برای گزارشگیری و خواندن اطلاعات از برنامه استفاده کنید و نه ثبت اطلاعات.
بنابراین استفاده از AntiForgeryToken را به چک لیست اجباری تولید تمام فرم‌های برنامه اضافه نمائید.

مطلب مشابه
Anti CSRF module for ASP.NET

فیلتر سفارشی بررسی Referrer

یکی دیگر از روش‌های مقابله با CSRF، بررسی اطلاعات هدر درخواست ارسالی است. اگر اطلاعات Referrer آن با دومین جاری تطابق نداشت، به معنای مشکل دار بودن درخواست رسیده است. فیلتر سفارشی زیر می‌تواند نمونه‌ای باشد جهت نمایش نحوه بررسی UrlReferrer درخواست رسیده:

```
using System.Web.Mvc;

namespace MvcApplication14.CustomFilter
{
    public class CheckReferrerAttribute : AuthorizeAttribute
    {
        public override void OnAuthorization(AuthorizationContext filterContext)
        {
            if (filterContext.HttpContext != null)
            {
                if (filterContext.HttpContext.Request.UrlReferrer == null)
                    throw new System.Web.HttpException("Invalid submission");

                if (filterContext.HttpContext.Request.UrlReferrer.Host != "mysite.com")
                    throw new System.Web.HttpException("This form wasn't submitted from this site!");
            }

            base.OnAuthorization(filterContext);
        }
    }
}
```

و برای استفاده از آن:

```
[HttpPost]
[CheckReferrer]
[ValidateAntiForgeryToken]
public ActionResult DeleteTask(int id)
```

نکته‌ای امنیتی در مورد آپلود فایل‌ها در ASP.NET

هر جایی که کاربر بتواند فایلی را به سرور شما آپلود کند، مشکلات امنیتی هم از همانجا شروع خواهند شد. مثلا در متد Upload قسمت 11 این سری، منعی در آپلود انواع فایل‌ها نیست و کاربر می‌تواند انواع و اقسام شل‌ها را جهت تحت کنترل گرفتن سایت و سرور آپلود و اجرا کند. راه حل چیست؟
از همان روش امنیتی مورد استفاده توسط تیم ASP.NET MVC استفاده می‌کنیم. فایل web.config قرار گرفته در پوشه Views را باز کنید (نه فایل وب کانفیگ ریشه اصلی سایت‌را). چنین تنظیمی را می‌توان مشاهده کرد:
برای IIS6 :

```
<system.web>
    <httpHandlers>
      <add path="*" verb="*" type="System.Web.HttpNotFoundHandler"/>
    </httpHandlers>
</system.web>
```

برای IIS7 :

```
<system.webServer>
    <handlers>
      <remove name="BlockViewHandler"/>
      <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode"
type="System.Web.HttpNotFoundHandler" />
    </handlers>
  </system.webServer>
```

تنظیم فوق، موتور اجرایی ASP.NET را در این پوشه خاص از کار می‌اندازد. به عبارتی اگر شخصی مسیر یک فایل aspx یا cshtml یا هر فایل قرار گرفته در پوشه Views را مستقیما در مرورگر خود وارد کند، با پیغام HttpNotFound مواجه خواهد شد.
این روش هم با ASP.NET Web forms سازگار است و هم با ASP.NET MVC؛ چون مرتبط است به موتور اجرایی ASP.NET که هر دوی این فریم ورک‌ها برفراز آن معنا پیدا می‌کنند.
بنابراین در پوشه فایل‌های آپلودی به سرور خود یک web.config را با محتوای فوق ایجاد کنید (و فقط باید مواظب باشید که این فایل حین آپلود فایل‌های جدید، overwrite نشود. مهم!). به این ترتیب این مسیر دیگر از طریق مرورگر قابل دسترسی نخواهد بود (با هر محتوایی). سپس برای ارائه فایل‌های آپلودی به کاربران از روش زیر استفاده کنید:

```
public ActionResult Download()
{
      return File(Server.MapPath("~/Myfiles/test.txt"), "text/plain");
}
```

مزیت مهم روش ذکر شده این است که کاربران مجاز به آپلود هر نوع فایلی خواهند بود و نیازی نیست لیست سیاه تهیه کنید که مثلا فایل‌هایی با پسوندهای خاص آپلود نشوند (که در این بین ممکن است لیست سیاه شما کامل نباشد ...).

علاوه بر تمام فیلترهای امنیتی که تاکنون بررسی شدند، فیلتر دیگری نیز به نام Authorize وجود دارد که در قسمت‌های بعدی بررسی خواهد شد.

نظرات خوانندگان

نویسنده: AhmadalliShafiee
تاریخ: ۲۲:۳۵:۱۹ ۱۳۹۱/۰۱/۲۹

سلام
با تشکر از شما بابت نوشتن این آموزش‌ها تا قسمت هفدهم.
من سر بخش آپلود فایل‌ها یه سوال دارم.
در زمینه هکینگ و ویروس‌نویسی تقریبا هیچ اطلاعی ندارم ولی فکر کنیم یک سری ویروس‌ها و فایل‌های اجرایی بدون نیاز به فراخوانی خودشون در محل آپلود اجرا میشن و باعث هک‌شدن ویا خراب‌شدن سرور میشن. درباره‌ی درست‌بودن این زیاد مطمئن نیستم ولی خوب، این ممکنه خودش یه مشکل امنیتی باشه.

نویسنده: Omid Shahryari
تاریخ: ۰۸:۱۵:۴۵ ۱۳۹۱/۰۱/۳۰

سلام
با تشکر از زحمتی که کشیدید . می خواستم بدونم بخش MVC به صورت کتاب هم منتشر خواهد شد ؟

ممنون

نویسنده: Salehi
تاریخ: ۱۰:۵۲:۳۹ ۱۳۹۱/۰۱/۳۰

-1 جایی خونده بودم افزونه هایی برای بعضی از مرورگرها وجود داره که می تونه UrlReferrer رو به دلخواه کاربر تغییر بده و بنابراین استفاده از این ویژگی کمکی نمی کنه.درسته؟
-2 به نظر شما کار درستیه که به طور کلی اگه مرورگر کاربر، کوکی رو غیرفعال کرده باشه اجازه دسترسی به برنامه بهش داده نشه و درخواست فعال کردن کوکی مطرح بشه. حداقل در برنامه های تحت وب و نه در وبسایت ها

نویسنده: وحید نصیری
تاریخ: ۰۹:۱۹:۰۹ ۱۳۹۱/۰۲/۰۱

- درسته. ولی محض احتیاط بررسی این مطلب ضرری نداره.
- بله. در برنامه‌های وب اصلا چنین اجازه‌ای رو نباید داد.

نویسنده: وحید نصیری
تاریخ: ۰۹:۲۰:۱۷ ۱۳۹۱/۰۲/۰۱

این یک کار سورس باز هست. می‌تونید مطابق قوانین آن، خودتون این کار رو انجام بدید و یک مرحله اون رو جلو ببرید. حاصل نهایی باید فایل ورد قابل ویرایش باشد. pdf درست نکنید. ممنون.

نویسنده: وحید نصیری
تاریخ: ۰۹:۲۱:۵۹ ۱۳۹۱/۰۲/۰۱

فایل تا زمانیکه فراخوانی و اجرا نشه، تخریبی رو نمی‌تونه ایجاد کنه. فیلتر یاد شده هم قابلیت اجرایی فایل‌های asp.net رو در یک پوشه خاص (مثلا پوشه آپلود) حذف می‌کنه.

نویسنده: asrin
تاریخ: ۱۲:۵۵ ۱۳۹۱/۰۶/۳۰

اگر داخل فرم اطلاعاتمان را با ajax به اکشن‌ها فرستاده باشیم چطور می‌شه از ValidateAntiForgeryToken داخل فرم استفاده

کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۳۰ ۱۳:۲۹

در متن فوق عنوان شد که این فیلتر «یک فیلد مخفی را به فرم» اضافه می‌کند:

```
<input name="__RequestVerificationToken" type="hidden" value="C0iPfy/3T....=" />
```

این فیلد مخفی اگر داخل فرم باشد، توسط درخواست‌های Ajax نیز ارسال خواهد شد . (مابقی آن خودکار است و نیاز به تنظیم خاصی ندارد)

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۴ ۱۲:۵

سلام
در مورد حملات csrf شما به عنوان مثال گفتید اگر کاربر ما login کرد و بعد وارد سایت دیگر شد
و مهاجم از ان سایت اقدام به هک کردن سایت میکنه
مگه هر کاربر وقتی که login می‌کنه و یا در کل وقتی یک request به سرور میفرسته web server یک session برای هر کاربر
نمیسازه که همراه ان guid که browser تولید می‌کنه و برای هر کاربر منحصر به فرد هست ساخته میشه ؟
اگر هم میشه یک link برای حملات csrf بزارید
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۴ ۱۲:۱۸

- درخواست تحت مجوز و مشخصات کامل شخص لاگین کرده اجرا میشه.
- مرتبط به زمینه فعالیت سایت ما نیست و چنین لینک‌هایی اینجا گذاشته نخواهد شد.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۴ ۱۴:۴۸

با سلام
ایا می‌توان فقط بعضی از تگ‌های html را اجازه‌ی استفاده داد به عنوان مثال کاربران فقط بتوانند از تگ bold استفاده کنند .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۴ ۱۵:۱۱

به صورت پیش فرض در ابتدا وجود موارد ذیل در یک درخواست بررسی می‌شوند:

```
<>*%&:\?
```

ولی در کل امکان تغییر موتور توکار آن هست:

```
<httpRuntime requestValidationType="CustomRequestValidation"/>

public class CustomRequestValidation : RequestValidator
{
   //...
}
```

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱:۲۷

سلام؛ در برنامه از DomainLayer, dataLayer استفاده کردم

برای استفاده از AllowHtml و همچنین HiddenInput می‌بایست فضای System.web.mvc را یوز کنم در حالی که DataLayer من در یک پروژه و Dll مجزا است .

یوز کردن یا Refrence دادن در لایه‌های دیتا به نیم اسپیس Mvc صحیح است؟ چرا که من تنها نیاز به متادیتای AllowHtml و HiddenInput دارم و همواره در دیتالایر ، نیم اسپیس‌های مربوط به EntityFrameWorkو دادها یوز شده.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۴ ۸:۴۸

از ViewModel استفاده کنید تا نیازی نداشته باشید AllowHtml رو به Domain model خودتون اعمال کنید.

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۳:۵۴

سلام

من ViewModel استفاده کردم و ViewModelها را هم در یک فایل مجزا و داخل پروژه DataLayer تعریف کرده ام ، با توضیحات شما ViewModel خود یک پروژه مجزا مثل DataLayer بشود، صحیح‌تر است و یا در خود پروژه اصلی تعریف شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۰۴ ۱۴:۸

مراجعه کنید به مطلب « چک لیست تهیه یک برنامه ASP.NET MVC ».

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۲/۰۴/۱۵ ۲۲:۲۰

سلام اقای نصیری
شما گفتید در هنگام اپلود فایل نباید بزاریم فایل وب کانفیگ overwrite بشه
با چه روشی این کار صورت میگره و چطور باید پیشگیری کنیم
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۵ ۲۲:۳۴

« کنترل عمومی فایل‌های آپلودی در ASP.NET MVC »

نویسنده: لیلا
تاریخ: ۱۳۹۲/۰۹/۱۱ ۶:۴۱

با تشکر
-1 زمانیکه از salt به صورت زیر استفاده میکنم

```
[ValidateAntiForgeryToken(Salt="1234")]
@Html.AntiForgeryToken(salt:"1234")
```

خطا زیر تولید می‌شود

Error 1 'System.Web.Mvc.ValidateAntiForgeryTokenAttribute.Salt' is obsolete: 'The 'Salt' property is deprecated. To specify custom data to be embedded within the token, use the static AntiForgeryConfig.AdditionalDataProvider property

-2 برای امنیت بیشتر چگونه مقدار salt را تولید کنیم؟"به جای رشته ساده"

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۱ ۹:۰

ذکر دستی Salt از MVC4 به بعد حذف شده. این مورد به صورت خودکار محاسبه و اعمال می‌شود.

نویسنده: لیلا
تاریخ: ۱۳۹۲/۰۹/۱۵ ۲۲:۵۲

با تشکر
به چه روشی میتوان از AntiForgeryToken در Ajax.ActionLink استفاده کرد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۶ ۱۳:۹

برای ارسال صحیح anti forgery token در حین اعمال Ajax ایی مراجعه کنید به مطلب ذیل:
« افزونه‌ای برای کپسوله سازی نکات ارسال یک فرم ASP.NET MVC به سرور توسط jQuery Ajax »

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۵:۵۶

سلام

من مرتبا با خطای AntiForgeryToken مواجه میشم.

یکبار این پیغام صرفا بر روی host ایجاد میشه وگاهی وقت‌ها هم بر روی لوکال. کلیه ورود اطلاعات را با این فیلتر امنیتی مشخص کردم . در اینترنت جستجو کردم و خیلی‌ها این مشکل داشتند و پیشنهاد کردن که در WebConfig نوع الگوریتم و کلید کد و دیکد را مشخص کنم، اما می‌خواستم بدونم اصلا علت بروز گه گاه این خطا چیست و چرا همیشگی نیست و مشخص کردن کد و الگوریتم کار صحیحی هست یا خیر؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۶:۴۷

چه خطایی دقیقا گرفتید؟ (من تابحال مشکلی نداشتم با فیلتر آنتی‌فورجری)

نویسنده: رضا گرمارودی

تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۷:۱۷

عنوان خطا: The anti-forgery token could not be decrypted

جزئیات خطا

```
The anti-forgery token could not be decrypted.
If this application is hosted by a Web Farm or cluster,
ensure that all machines are running the same version of ASP.NET Web Pages and
that the <machineKey> configuration specifies explicit encryption and validation keys.
AutoGenerate cannot be used in a cluster

Description: An unhandled exception occurred during the execution of the current web request.
Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Web.Mvc.HttpAntiForgeryException:
The anti-forgery token could not be decrypted.
If this application is hosted by a Web Farm or cluster,
ensure that all machines are running the same version of ASP.NET Web Pages and that the <machineKey>
configuration
specifies explicit encryption and validation keys. AutoGenerate cannot be used in a cluster.
```

شرایط برنامه: در زمان ذخیره سازی اطلاعات.(در Mvc دات نت فریم ورک 4.5 و Vs2013)
موارد مشابه در سایر انجمن‌ها : اینجا و اینجا و ...

نتیجه حاصله: درج MachinKey در WebConfig برای مثال:

```
<system.web>
  <machineKey validationKey="mykey" decryptionKey="myotherkey" validation="SHA1" decryption="AES" />
</system.web>
```

سوال: علت بروز خطا بر روی بعضی از سیستم‌ها چیست؟ چرا همیشه این خطا نمایش داده نمی‌شود. چرا پیغام نمی‌توانم Decrypt کنم؟ پس چرا در همین برنامه ولی دربخش دیگری که از anti-forgery token استفاده شده مشکل وجود ندارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۸:۷

خلاصه موارد ممکن:
- در یک صفحه چندبار از Html.AntiForgeryToken استفاده شده‌است. این مورد کوکی آنتی‌فورجری را تخریب می‌کند و نهایتا اطلاعات آن قابل رمزگشایی و مقایسه در سمت سرور نخواهد بود.
- تنظیم AntiForgeryConfig.SuppressIdentityHeuristicChecks = true را به فایل global.asax.cs نیز اضافه کنید.
آنتی‌فورجری توکن وضعیت کاربر لاگین شده به سیستم را نیز نگهداری و رمزنگاری می‌کند. در این حالت اگر در یک برگه‌ی دیگر لاگ آف کنید و در برگه‌ی قبلی سعی در ارسال فرم، آنتی‌فورجری توکن یک پیام خطا را نمایش می‌دهد. با تنظیم SuppressIdentityHeuristicChecks = true این بررسی وضعیت لاگین شخص حذف خواهد شد.
- حالت تنظیم machine key یاد شده، در یک web farm با چندین سرور ممکن است رخ دهد. اگر چنین حالتی را ندارید، تنظیمی را تغییر ندهید.

نویسنده: ali.rezayee
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۹:۴۶

با سلام و عرض خسته نباشید.
من در پروژه ام به شکل ذیل عمل کردم؛ آیا اشتباه است؟
-1 بجای آنکه ابتدای هر فرم @Html.AntiForgeryToken را قرار بدهم، فقط یکبار در _Layout.cshtml و در Body آنرا ذکر کردم.
-2تمامی ارسال‌ها به سرور را توسط Post Ajax انجام میدهم؛ چه درخواست نمایش یک View باشد چه ثبت کالای جدید در

دیتابیس همه توسط کدی مثل کد ذیل به سرور Post میشوند:

```
var token = $('[name=__RequestVerificationToken]').val();
data.__RequestVerificationToken = token;
...
...
...
 $.ajax({
  url: url,
  type: "POST",
  data: data
});
```

ممنون

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۱/۲۱ ۲۳:۹

سلام؛ ممنون بابت پاسختون. این طوری به forgeryToken نگاه نکرده بودم.

در صفحه برای Delete و ثبت اطلاعات و یا حتی فیلترکردن اطلاعات و هرجا که کاربر اطلاعاتی وارد می‌کند از AntiForgeryToken استفاده کردم و حذف و ورود اطلاعات و فیلتر کلا در یک صفحه اما هرکدام در تگ Form مجزایی ارسال و دریافت می‌شود و در هر تگ Form یک ForgeryToken گذاشتم.

برای اونها چه کار کنم؟ اصلا کار بنده درست بوده؟ من هرکجا کاربر اطلاعاتی وارد کرده از Token استفاده کردم ولی مشکل اینجاست که با توجه به طراحی کل اونها نیازه که در یک صفحه باشه.

می تونم یک AntiForgeryToken کلی داشته باشم و با جاوا اسکریپت اون و بخونم و به Form.Serilaise() اضافه کنم. اما می‌خواستم راه بهینه اون و بپرسم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۱ ۲۳:۴۱

کلا یک آنتی فورجری توکن برای کل صفحه کافی است. روشی که آقای رضایی کمی بالاتر مطرح کردند (قرار دادن آن در فایل layout) هم جالب است. به این صورت در همه‌جا و در تمام صفحات حضور خواهد داشت؛ آن‌هم فقط یکبار و نه بیشتر. نحوه‌ی خواندن و اضافه کردن آن نیز به صورت زیر خواهد بود:

```
function addToken(data) {
   data.__RequestVerificationToken = $("input[name=__RequestVerificationToken]").val();
   return data;
}

$.ajax({
   // .....
   data: addToken({ postId: postId }), // اضافه کردن توکن
   dataType: "html", // نوع داده مهم است
   // .....
});
```

سه نکته اینجا مهم است:

- data type حتما باید در این حالت html باشد.
- در قسمت data متد addToken کار افزودن خودکار آنتی فورجری‌توکن صفحه را انجام می‌دهد (محل آن مهم نیست که داخل فرم باشد یا خارج از آن).
- سطر contentType نباید ذکر شود.

نویسنده: سعیدج
تاریخ: ۱۳۹۳/۰۱/۰۶ ۱۳:۱۹

با سلام
من از روش ایجاد فایل web.config برای upload فایل استفاده کردم. جواب هم داد. ولی مشکل این هست که در داخل خود صفحات هم به آن فایل‌ها دسترسی پیدا نمیکنه.
اون صفحه خطای عدم اجازه جهت دسترسی به پوشه ای که مستقیم در مرروگر وارد شده را چه طور سفارشی کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۶ ۱۳:۴۵

- مدیریت خطاها در یک برنامه ASP.NET MVC
- محدود کردن کاربرها به آپلود فایل‌هایی خاص در ASP.NET MVC
- کنترل عمومی فایل‌های آپلودی در ASP.NET MVC

نویسنده: امیر نوروزیان
تاریخ: ۱۳۹۳/۱۰/۱۱ ۹:۲

با سلام
۱- با توجه به اینکه برای این فیلتر ValidateAntiForgeryToken باید حتما کوکی فعال باشه آیا درسته استفاده کنیم. اگه خواسته نباشیم از این فیلتر استفاده کنیم روشی دیگری هست.
۲- بین Referrer و ValidateAntiForgeryToken کدوم فیلتر استفاده شود. توصیه به استفاده کدوم فیلتر. آیا همزمان میشه استفاده کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۱ ۱۰:۱۰

- شما هستید که تصمیم می‌گیرید چه کسی از سایتتان استفاده کند یا خیر. تمام برنامه‌های مدرن، نیاز به فعال بودن کوکی و جاوا اسکریپت، دارند. بدون این‌ها، وب به سال‌های اول آن باز خواهد گشت و صرفا کاربرد ارائه‌ی محتوا را پیدا می‌کند و نه تهیه‌ی یک برنامه‌ی وب که مفاهیمی مانند اعتبارسنجی، سشن و غیره آن نیز بدون وجود کوکی‌ها مشکلات عمده‌ای را پیدا خواهند کرد.
استفاده از کوکی‌های رمزنگاری شده، از لحاظ کمتر مصرف شدن منابع در سمت سرور نیز مهم است؛ چون بار ذخیره سازی یک سری از اطلاعات، به مرورگر کاربر منتقل می‌شود (بالا رفتن مقیاس پذیری با کمتر مصرف شدن حافظه‌ی سرور).
- AntiForgeryToken مهم‌تر است. امکان استفاده‌ی همزمان هم وجود دارد. نمونه‌ی استفاده از چند ویژگی در یک اکشن متد، ذیل فیلتر مربوط به آن، در متن هست.