عنوان: **Cookie - قسمت اول**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۰۵ ۱۲:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: Cookie, Http Cookie, Magic Cookie

مقدمه

مدت زیادی است که کوکی‌ها در عرصه وب نقش مهمی ایفا میکنند، اما متاسفانه مفهوم روشن و واضحی از آن و نحوه کار آن در اختیار اکثر کاربران و توسعه دهندگان وب نیست. شاید اولین مشکل ناشی از سوءتفاهم‌های بسیاری باشد که درباره کوکی وجود دارد. مثلا برخی آن را ابزاری صرفا برای جاسوسی از کاربران اینترنتی میدانند. برخی دیگر، از آنها و نحوه کارکردشان کلا صرفنظر میکنند. مشکل دیگری که در رابطه با کوکی‌ها میتوان برشمرد، عدم وجود رابط کاربری مناسب برای بررسی و مدیریت کوکی هاست. اما با وجود این مشکلات و برخی دیگر امروزه کوکی‌ها جزئی بسیار مهم در وب هستند که درصورت حذفشان، بسیاری از وب سایتها و برنامه‌های مبتنی بر وب از کار خواهند افتاد.

یک کوکی (cookie به معنی شیرینی یا کلوچه! که با عناوین دیگری چون Http Cookie و Web Coockie و Browser Cookie نیز شناخته میشود)، به داده‌های ارسالی از یک وب سرور (که معمولا بصورت داده‌های متنی کدگذاری شده هستند) اطلاق میشود که در مخزنی مخصوص در مرورگر کاربر به هنگام بازدید از یک سایت ذخیره میشود. وقتیکه کاربر سایت مذبور را در آینده دوباره مرور کند، این داده‌های ذخیره شده توسط مروگر به وب سرور ارسال میشود تا مثلا فعالیتهای قبلی کاربر مورد بررسی قرار گیرد. کوکی‌ها برای فراهم کردن مکانیزمی قابل اعتماد جهت ذخیره فعالیتهای قبلی یا آخرین وضعیت کاربر در یک وب‌سایت طراحی شده‌اند. با اینکه کوکی‌ها دسترسی بسیار محدودی در سمت کلاینت دارند (تقریبا هیچ دسترسی‌ای به هیچیک از منابع سیستم کاربر ندارند) اما با پیگیری هوشمند و هدفمند برخی از آنها میتوان به داده‌هایی از تاریخچه فعالیتهای کاربر در یک مرورگر و سایت خاص دست یافت که به نوعی **نقض حریم شخصی** کاربران به حساب می‌آید.

نکته: درواقع میتوان گفت که از کوکی به نوعی برای فراهم کردن "حافظه" موقت برای مرورگرها در ارتباط با وب سرورها استفاده میشود.

پروتوکل HTTP که برای تبادل داده‌ها میان مرورگر و وب سرور در بارگذاری صفحات وب استفاده میشود، پروتوکلی بدون حالت یا وضعیت (state-less) است. بدین معنی که به محض ارسال داده‌های یک صفحه وب به سمت مرورگری که آنرا درخواست کرده، وب سرور هیچ چیزی از این تبادل داده را ذخیره و نگهداری نمیکند. بنابراین در درخواست‌های دوباره و سه باره و ... بعدی، وب سرور با آنها همچون اولین درخواست برخورد کرده و رفتاری کاملا یکسان در برخورد با این درخواستها نشان خواهد و دقیقا همان داده‌ها را به سمت مرورگر ارسال خواهد کرد.

این رفتار در موارد زیادی میتواند دردسرساز باشد. مثلا وب سرور نمیتواند بفهمد که یک کاربر لاگ‌آن (LogOn یا همان SignIn) کرده و یا اینکه یکسری تنظیمات شخصی اعمال کرده است، چون جایی برای ذخیره و نگهداری این حالات یا وضعیتها در پروتوکل HTTP وجود ندارد. خوشبختانه وجود کوکی‌ها یکی از بهترین راه حل‌ها برای رفع مشکلات اشاره شده است.

بنابراین همانطور که اشاره شده یکی از مهمترین انواع کاربردهای کوکی‌ها در زمینه اعتبار سنجی کاربران است. با استفاده از این نوع کوکی وب سایتها میتوانند از وضعیت ورود یا خروج کاربران و نیز انواع دسترسی‌ها و تنظیمات آنها باخبر شوند. البته با توجه به حساسیت این موضوع، درباره نحوه ذخیره داده‌ها در این نوع کوکی‌ها باید دقت خاصی اعمال شود. اگر در این زمینه سهل انگاری هایی انجام شود، ممکن است خوراک جذابی برای هکرها فراهم شود! تبلیغات درون سایتها نیز از قسمتهایی است که استفاده بسیاری از کوکی میکند که بعضا موجب بروز خطراتی برای کاربران میشود.

تاریخچه

واژه Cookie از عبارت Magic Cookie برگرفته شده است. به طور خلاصه Magic Cookie به مجموعه‌ای از داده‌های «بدون نیاز به تغییر» میگویند که بین برنامه‌هایی که در ارتباط با یکدیگرند، ردوبدل میشود. داده‌های موجود در Magic Cookieها معمولا برای سمت دریافت کننده مفهوم خاصی ندارد و به نوعی برای ذخیره وضعیت «سمت دریافت کننده» در «برنامه ارسال کننده» و استفاده از آن در ارتباطهای بعدی کاربرد دارد. به بیان دیگر حالت یا وضعیت یا تنظیمات «برنامه مقصد در برنامه مبدأ» با استفاده

از کوکی در «خود برنامه مقصد» نگهداری میشود!
در سال 1994 آقای Lou Montulli هنگامیکه در شرکت Netscape Communications در توسعه یک برنامه تجاری تحت وب مشارکت داشت، ایده استفاده از این تکنولوژی را در ارتباطات وب ارائه داد که بعدها عنوان HTTP Coockie را بخود گرفت. برای اولین بار از این کوکی‌ها در نسخه 0.9 بتای نت اسکیپ که در همان سال ارائه شد، پشتیبانی و استفاده شد. مروگر IE هم در سال 1995 و در نسخه 2.0 آن، پشتیبانی از کوکی را آغاز کرد. آقای مانتولی پتنت (Patent) تکنولوژی کوکی را در سال 1995 ارائه داد اما ثبت نهایی آن به دلیل مشکلات و مباحث حریم شخصی کاربران تا سال 1998 طول کشید.

کوکی واقعا چیست؟
یک کوکی در واقع یک فایل متنی کوچک است که در قسمتی مشخص از کامپیوتر کلاینت که توسط مرورگر تنظیم شده است، ذخیره میشود. این فایل متنی کوچک حاوی اطلاعات زیر است:
- یک جفت داده نام-مقدار (name-value pair) که داده اصلی کوکی را نگهداری میکند.
- خاصیتی برای مشخص کردن زمان انقضای کوکی (پس از این زمان این فایل متنی کوچک از درون مرورگر حذف خواهد شد)
- خاصیتهایی برای مشخص کردن محدوده‌ها و مسیرهای قابل دسترسی کوکی
- خاصیتهایی برای تعیین نحوه تبادل داده‌های کوکی و نوع دسترسی به این داده‌ها (مثلا الزام به استفاده از پروتوکلهای امن)

انواع کوکی
بطور کلی دو نوع اصلی کوکی وجود دارد:
1. **Session cookie**

از این نوع کوکی برای نگهداری موقت داده‌ها نظیر داده‌های مربوط به وضعیت یک کاربر، تنها در زمان مرور وب سایت استفاده میشود. معمولا با بستن مرورگر (یا اتمام سشن) این کوکی‌ها ازبین میروند.
2. **Persistent cookie**

برخلاف کوکی‌های سشنی این نوع کوکی‌ها در سیستم کلاینت به صورت دائمی ذخیره میشوند. معمولا دارای یک تاریخ انقضا هستند که پس از آن از بین میروند. در طول زمان حیات این کوکی ها، مرورگرها داده‌های ذخیره شده در آنها را با توجه به تنظیمات درونشان در هر درخواست به سمت وب سرور سایت مربوطه ارسال میکنند.

با توجه به کاربردهای فراوان کوکی، دسته بندیها و انواع دیگری از کوکی را هم میتوان نام برد. مانند انواع زیر:
Secure cookie

معمولا به کوکی‌هایی که خاصیت امن (Secure Attribute) در آنها فعال است این عنوان اطلاق میشود. این نوع از کوکی‌ها تنها قابل استفاده در ارتباطهای امن (با استفاده از پروتوکل HTTPS) هستند. این خاصیت اطمینان میدهد که داده‌های موجود هنگام تبادل بین سرور و کلاینت همواره کدگذاری میشود.
HttpOnly cookie

در این کوکی‌ها خاصیت HttpOnly فعال است، که موجب میشود که از آنها تنها در ارتباطات از نوع HTTP و HTTPS بتوان استفاده کرد. در سایر روشهای دسترسی به کوکی (مثلا از طریق برنامه نویسی سمت کلاینت) نمیتوان به محتوای این نوع از کوکی‌ها دسترسی پیدا کرد.
Third-party cookie

این نوع از کوکی‌ها در مقابل کوکی‌های First party (یا شخص اول) وجود دارند. کوکی‌های شخص اول توسط وب سایت جاری تولید شده اند، یعنی نشانی دامِین این کوکیها مربوط به سایت جاری است. منظور از سایت یا دامین جاری، سایتی است که آدرس آن در نوار آدرس مرورگر نشان داده میشود. کوکی‌های Third party (یا شخص سوم) به آن دسته از کوکی‌ها میگویند که توسط دامین یا وب سایت دیگری غیر از وب سایت جاری تولید و مدیریت میشوند. مثلا کوکی‌های مربوط به سایتهای تبلیغاتی. البته در مرورگرهای مدرن این نوع از کوکی‌ها به دلیل مشکلات امنیتی و نقض حریم شخصی کاربران عموما بلاک میشوند.
Supercookie

یک سوپرکوکی به آن دسته از کوکی‌ها گفته میشود که خاصیت دامین آنها به یک پسوند خیلی کلی مثل com. تنظیم شده باشد. به دلیل مسائل امنیتی بیشتر مرورگرهای مدرن تمامی انواع این سوپرکوکی‌ها را بلاک میکنند. امروزه لیستی از این پسوندهای کلی با عنوان Public Suffix List موجود است که در مرورگرهای مدرن برای کنترل کوکی‌ها استفاده میشود.

موارد استفاده از کوکی

- مدیریت جلسات (Session Management):

از کوکی میتوان برای نگهداری داده‌های مربوط به یک کاربر در بازدید از صفحات سایت و تعامل با آنها (که ممکن است در زمانهای مختلف رخ دهد) استفاده کرد. یکی از موارد بسیار پرکاربرد در این زمینه، کوکی‌های تعیین اعتبار یک کاربر است که پس از ورود به سایت در هر درخواست توسط مرورگر به سمت سرور ارسال میشود.

مثال دیگری در مورد این کاربرد نگهداری از داده‌های سبد خرید یک کاربر است. این داده‌ها را میتوان قبل از تسویه نهایی درون یک کوکی ذخیره کرد. معمولا در تمام این موارد از یک کلید منحصر به فرد که در سمت سرور تولید شده و درون کوکی به همراه سایر اطلاعات ذخیره میشود، برای تعیین هویت کاربر استفاده میشود.

- شخصی سازی (Personalization):

یکی دیگر از موارد پرکاربرد کوکیها ذخیره تنظیمات یا داده‌های مرتبط با شخصی سازی تعامل کاربر با سایت است. مثلا میتوان تنظیمات مربوط به استایل یک سایت یا زبان انتخابی برای یک کاربر مشخص را درون کوکی در سمت کلاینت ذخیره کرد. سایتهای بزرگ معمولا از این روش برای ذخیره تنظیمات استفاده میکنند، مثل گوگل و ویکیپدیا. همچنین میتوان از کوکی برای ذخیره شناسه آخرینکاربری که در سایت لاگ‌آن کرده استفاده کرد تا در مراجعه بعدی به عنوان اولین انتخاب در صفحه ورود به سایت به کاربر نمایش داد (هرچند این کار معایب خودش را دارد).

- پیگیری یا ردیابی (Tracking):

از کوکی‌ها میتوان برای پیگیری بازدیدهای یک کاربر در یک کلاینت از یک سایت یا مسیری به خصوص از یک سایت بهره برد. بدین صورت که برای هر کاربر یک کد شناسایی منحصر به فرد تولید شده و درون کوکی مخصوص این کار ذخیره میشود. سپس برای هردرخواست میتوان مثلا نشانی صفحه موردنظر و زمان و تاریخ آن را درون یک منبع ذخیره کرد تا برای استفاده‌های آتی به کار روند.

البته کاربردها و استفاده‌های دیگری نیز برای کوکی میتوان برشمرد که بدلیل طولانی شدن بحث از آنها صرفنظر میشود.

ایجاد کوکی

همانطور که در بالا نیز اشاره شد، کوکی‌ها داده هایی هستند که توسط وب سرور برای ذخیره در کلاینت (مرورگر) تولید میشوند. مرورگرها نیز باید این داده‌ها را بدون هیچ تغییری در هر درخواست عینا به سمت سرور برگردانند. درواقع با استفاده از کوکی‌ها میتوان به ارتباط بدون حالت HTTP به نوعی خاصیتی از جنس state اضافه کرد. به غیر از خود وب سرور، برای تنظیم و یا تولید کوکی میتوان از زبان‌های برنامه نویسی سمت کلاینت (مثل جاوا اسکریپت) البته درصورت پشتیبانی توسط مرورگر نیز استفاده کرد.

در جدیدترین استانداردهای موجود (RFC 6265) درباره کوکی آورده شده که مرورگرها باید بتوانند حداقلهای زیر را پشتیبانی کنند:

- توانایی ذخیره حداقل 3000 کوکی
- توانایی ذخیره کوکیها با حجم حداقل 4 کیلوبایت
- توانایی ذخیره و نگهداری حداقل 50 کوکی به ازای هر سایت یا دامین

نکته: توانایی مرورگرهای مدرن در مدیریت کوکی‌ها ممکن است فراتر از استانداردهای اشاره شده در بالا باشد.

انتقال داده‌های صفحات وب سایتها از طریق پروتوکل HTTP انجام میشود. مرورگر برای بارگذاری صفحه موردنظر کاربر از یک وب سایت، معمولا یک متن کوتاه به وب سرور مربوطه ارسال میکند که به آن HTTP Request میگویند. مثلا برای دریافت صفحه http://www.dotnettips.info/index.html درخواستی به شکل زیر به سمت وب سرور ارسال میشود:

```
GET /index.html HTTP/1.1
Host: www.dotnettips.info
```

مثلا نمونه یک درخواست کامل خام (Raw) از صفحه اول سایت جاری در نرم افزار Fiddler بصورت زیر است:

```
GET http://www.dotnettips.info/ HTTP/1.1
```

```
Host: www.dotnettips.info
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17 (KHTML, like Gecko) Chrome/24.0.1312.56
Safari/537.17
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: BlogPost-1175=NLOpR%2fgHcUGqPL8dZYv3BDDqgd4xOtiiNxHIp1rD%2bAQ%3d; BlogComment-
5002=Wl51iaIsiBnQN1UDD4p%2fHFvuoxC3b8ckbw78mAWXZOSWMpxPlLo65%2bA40%2flFVR54;
ReaderEmail=DP%2bx4TEtMT2LyhNQ5QqsArka%2fWALP5LYX8Y
```

وب سرور با ارسال محتویات صفحه موردنظر به این درخواست پاسخ میدهد که به آن HTTP Response میگویند.
در پاسخ ارسالی، وب سرور میتواند با استفاده از یک header مخصوص با نام **Set-Cookie** یک کوکی را ایجاد کند. در زیر یک نمونه از این پاسخها را مشاهده میکنید:

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: cookieName=cookieValue
Set-Cookie: cookieName 2= cookieV alue2; Expires=Thr, 10-Jun-2021 10:18:14 GMT
...
```

نمونه پاسخ ارسالی خام (Raw) در نرم افزار Fiddler مربوط به درخواست صفحه اول سایت جاری بصورت زیر است:

```
HTTP/1.1 200 OK
Date: Wed, 30 Jan 2013 20:25:15 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
X-Compressed-By: HttpCompress
Set-Cookie: .ASPXROLES=NzZ9qIRpCWHofryYglbsQFv_SSGPn7ivo0zKFoS94gcODVdIKQAe_IBwuc-TQ-
03jGeIkZabTuxA0A3k2-
nChy7iAWw9rPMYXSkqzMkizRFkDC0k3gQTkdLqLmmeIfnL9UjfMNWO8iVkYQrSv24ecbpFDSQCH827V2kEj8k2oCm_5sKRSmFpifh4N
7kinEi0vomG1vW4Rbg9JWMhCgcvndvsFsXxpj-NiEikC1RqHpiLArIyalEMEN-cIuVtRe7uoo938u9l-
7OXb8yzXucVl4bdqPy2DXM3ddWzb3OH1jSFM6gxwJ8qRZDlSGmEEbhji7rA-
efI4aYGTKx6heWfUsY6E2k73jJLbuZ3RB4oNwRYmz8FRB0-vm1pO7rhF1JIoi1YB17ez-Ox5chNEFkPVREanHVU9DxboJ5dKgN-
2B5udUFPunnshbN8EBhixbFQOpqRiiOK4uWWaWy3rVEJYpCCDBRctKCfEyYD1URFYeajB0AXmiMUTcGeuUtwb-
XFjbQZnbylmMF3EJgG16bcc1IEkTAUv1JfKjaql0XGWJI1; path=/; HttpOnly
Cache-Control: private
Content-Type: text/html; charset=utf-8
Content-Length: 106727

<!DOCTYPE html>
<html>
...
```

وب سرور دستور Set-Cookie را تنها برای ثبت کوکی در مرورگر در پاسخ ارسالی قرار میدهد. برای آشنایی بیشتر با این هدر و ساختار آن به RFC 6265 مراجعه کنید. این دستور برای مرورگر مشخص میکند که (درصورت پشتیبانی از کوکی و فعال بودن آن در مرورگر) در درخواستهای بعدی باید این کوکی‌ها را در متن درخواست ارسالی به سمت سرور ضمیمه کند. البته اینکار با توجه به تنظیمات خاصیتهای کوکی مربوطه انجام میشود که در ادامه بحث میشود. برای ارسال کوکی به سمت وب سرور، مرورگر از هدر Cookie در درخواست خود استفاده میکند.
مثلا با توجه به مثال قبل برای درخواست صفحه http://www.dotnettips.info/index2.html مرورگر میتواند متن زیر را به سمت وب سرور ارسال میکند:

```
GET /index2.html HTTP/1.1
Host: www.example.org
Cookie: cookieName=cookieValue; cookieName2=cookieValue2
Accept: */*
```

این درخواست که برای صفحه دیگری از همان سایت قبلی است، با درخواست اول متفاوت است. با استفاده از این درخواست وب سرور میفهمد که این درخواست با درخواست قبلی مرتبط است. وب سرور در پاسخ ارسالی میتواند کوکی‌های دیگری نیز ثبت کند.

وب سرور میتواند مقدار یک کوکی ثبت شده در مروگر را با استفاده از دستوری مشابه Set-Cookie: cookieName=cookieNewValue در پاسخ ارسالی به سمت کلاینت تغییر دهد. مرورگر با دیدن این خط در پاسخ دریافتی از وب سرور مقدار کوکی را به روز رسانی میکند. البته به شرطی که سایر خواص تنظیم شده برای کوکی **عینا** یکسان باشد.

نکته: identity یا هویت کوکی با استفاده از **تمام خواص آن به جز expires یا max-age** تعیین میشود.

مقدار یک کوکی میتواند شامل هر کاراکتر قابل چاپ اسکی (از کاراکتر ! تا کاراکتر ~ یا از کد یونیکد \u0021 تا \u007E) به غیر از کاراکترهای , و ; و فضای خالی باشد (استفاده از این سه کاراکتر و سایر کاراکترهای غیر اسکی تنها با انکدینگ میسر است). نام یک کوکی نیز از این قاعده پیروی میکند، البته شامل کاراکتر = نیز نمیتواند باشد چون این کاراکتر نقش جداکننده «مقدار» از «نام» کوکی را ایفا میکند. استانداردهای کوکی که در RFC 6265 آورده شده است، محدودیتهای بیشتری نیز دارد که ممکن است توسط مروگرهای امروزی رعایت نشود!

نکته: انکدینگ کوکیها کمی بحث دارد. ازآنجاکه کوکی‌ها به صورت هدرهای پروتوکل HTTP انتقال داده میشوند، بنابراین کاراکترهای موجود در آن باید تنها از نوع ASCII باشند. اما چون ارسال کننده و دریافت کننده نهایی کوکی یک وب سرور یکسان است بنابراین وب سرور برای ذخیره کاراکترهای غیر اسکی میتواند از انکدینگ خاص خود استفاده کند. مثلا عموم وب سرورها و نیز مرورگرها از URL Encoding برای انکدینگ کوکی‌ها استفاده میکنند (^). ظاهرا در تمام مرورگرهای مدرن برای ذخیره کوکی ها، حداقل نام و مقدار کوکی به صورت جداگانه (مثلا برای ذخیره کاراکترهای نامعتبر) انکد میشود به غیر از کاراکتر تساوی (=) بین نام و مقدار کوکی.

با استفاده از زبانهای برنامه نویسی سمت کلاینت نیز میتوان کوکی‌ها را مدیریت کرد. مثلا در جاوا اسکریپت از **document.cookie** برای اینکار استفاده میشود. نحوه کاربرد و استفاده از این پراپرتی کمی غیرعادی است. مثلا دستور document.cookie = 'dummy=a11a' یک کوکی با نام dummy و با مقدار a11a ایجاد میکند! در ادامه با این دستور و نحوه کارکردن با کوکی در جاوا اسکریپت بیشتر آشنا میشویم.

نکته: برای انکد رشته‌ها در جاوا اسکریپت از دستور escape استفاده میشود. عملیات عکس آن با دستور unescape انجام میشود.

نکته: با اینکه استاندارد تعریف کوکی مشخص کرده که برای تعریف کوکی وجود عبارتی به صورت name=value اجباری است، اما ظاهرا بیشتر مرورگرها صحت تعریف کوکی و اعتبار آنرا برای پیروی از این طرح بررسی نمیکنند و بنابراین میتوان صرفا با استفاده از یک رشته بدون علامت مساوی یک کوکی را ایجاد کرد.

خواص یک کوکی

به غیر از نام و مقدار، کوکی‌ها خواص دیگری همچون دامین (domain)، مسیر (path)، تاریخ انقضا (expiration date) یا حداکثر طول عمر (maximum age)، و Secure و HttpOnly دارند که میتوانند توسط وب سرور و یا با استفاده از زبانهای برنامه نویسی کلاینتی تنظیم شوند. مرورگرها این خاصیتها را به وب سرور ارسال نمیکنند. تنها مقادیری که به سمت وب سرور برگشت داده میشوند، نام و مقدار کوکی‌هاست. مرورگرها با استفاده از خواص کوکی زمان حذف کوکی و یا کنترل دسترسی به مقدار آن با توجه به آدرس جاری مرورگر و نیز اینکه آیا اصلا کوکی را به وب سرور ارسال کنند یا نه، را تعیین میکنند. خواص کوکی در ادامه شرح داده شده است:

1. تاریخ انقضا و حداکثر طول عمر (Expires و Max-Age)

با استفاده از یکی از این دو خاصیت، تاریخی که دیگر نیازی نیست تا کوکی به سمت سرور ارسال شود، تعیین میشود. بنابراین پس از این تاریخ، ممکن است کوکی از مخزن مرورگر پاک شود. برپایه اطلاعات موجود در RFC 6265 ، در خاصیت Expires، تاریخ انقضای کوکی باید به فرمت "Wdy, DD Mon YYYY HH:MM:SS GMT" باشد. مثل Mon, 17-Mar-2014 1:00:00 GMT. همچنین خاصیت Max-Age طول عمر کوکی را برحسب **ثانیه** از لحظه دریافت توسط مرورگر مشخص میکند. به نمونه‌های زیر توجه کنید:

```
... ;Set-Cookie: cookie1=abc; Expires=Mon, 17-Mar-2014 01:00:00 GMT
... ;Set-Cookie: cookie2=123
... ;Set-Cookie: cookie3=abc; Expires=Thu, 01-Jan-1970 00:00:01 GMT
... ;Set-Cookie: cookie3=abc; max-age=31536000
.....
```

در دستور اول، cookie1 برای حذف در تاریخ مشخص شده تنظیم شده است. در خط دوم که بدون این دو خاصیت است، یک

نوع کوکی سشنی تعریف شده است. این کوکی پس از بسته شدن مرورگر (اتمام سشن) از حافظه پاک خواهد شد.

نکته: اتمام یک سشن برای کوکی‌های سشنی دقیقا به معنی بستن مرورگر (یا تب مربوطه در مروگرهای مدرن) است.

دستور سوم که تاریخ انقضای کوکی را به تاریخی در گذشته تنظیم کرده است به مرورگر اعلام میکند که باید cookie3 را پاک کند. این روش استاندارد برای حذف یک کوکی است.

نکته: استفاده از روش تنظیم یک تاریخ انقضا در گذشته برای حذف یک کوکی تنها وقتی کار خواهد که سایر خواص تعیین شده در دستور Set-Cookie با مقادیر موجود در حافظه مرورگر دقیقا یکی باشد تا هویت کوکی موردنظر به صورت منحصربه فرد تعیین شود.

در خط چهارم به مرورگر اعلام میشود که cookie4 باید دقیقا یک سال پس از لحظه دریافت کوکی، حذف شود.

نکته: خاصیت max-age در مرورگر IE8 و نسخه‌های قبل از آن پشتیبانی نمیشود.

نکته: گزینه ای که معمولا در صفحات لاگ‌آن (LogOn) یا ساین‌این (SignIn) برای ذخیره داده‌های کاربر وجود دارد (مثل «مرا به خاطر بسپار»)، مرتبط با این خاصیت از کوکی هاست. در صورت عدم انتخاب این گزینه معمولا یک کوکی سشنی (بدون خاصیت expires) ایجاد میشود. اما با انتخاب این گزینه، یک کوکی ماندگار (Persistent) با خاصیت expires برابر با تاریخی در آینده ایجاد میشود تا درصورت بسته شدن مرورگر (اتمام سشن) داده‌های کاربر پاک نشود.

نکته: تاریخ انقضای کوکی با استفاده از تاریخ کلاینت تعیین میشود. متاسفانه هیچ راه مستقیمی برای همزمانی این تاریخ با تاریخ سرور وجود ندارد.

2. دامین و مسیر (Domain و Path)

خاصیتهای دامین و مسیر کوکی، محدوده قابل دسترس کوکی را مشخص میکنند. با استفاده از این دو خاصیت مرورگر متوجه میشود که آیا کوکی را در موقعیت و آدرس جاری باید به سمت وب سرور ارسال کند یا خیر. همچنین دسترسی به کوکی‌ها در سمت کلاینت با توجه به این دو خاصیت محدود میشود. مقدار پیش فرض این دو خاصیت برابر مسیر و دامین جاری مرورگر است که اگر مقداری برای این دو خاصیت تعیین نشود، به کوکی تعلق میگیرد.

نکته: منظور از وضعیت جاری، موقعیتی است که کوکی مذبور در آن ایجاد شده است. مثلا آدرس صفحه ای که هدر Set-Cookie را ارسال کرده و یا آدرس صفحه ای که در آن با استفاده از دستوری مشابه document.cookie = 'a=b'; کوکی مربوطه ایجاد شده است. به عنوان نمونه اگر یک کوکی در صفحه جاری همین سایت ایجاد شود و خاصیتهای دامین و مسیر آن مقداردهی نشود، مقدار دامین به www.dotnettips.info و مقدار مسیر آن به post/ تنظیم خواهد شد.

نکته: مرورگر بررسی دامین کوکی را ازطریق «مقایسه از انتها» انجام میدهد. یعنی اگر مثلا دامین یک کوکی برابر dotnettips.info باشد، این کوکی در ساب دامین‌های d1.dotettips.info و یا www.dotnettips.info و از این قبیل در دسترس خواهد بود. برای کسب اطلاعات بیشتر میتوان به RFC 6265 (قسمت Domain Matching) مراجعه کرد.

نکته: بررسی مسیر کوکی برخلاف دامین آن، ازطریق «مقایسه از ابتدا» انجام میشود. یعنی آدرس صفحه جاری پس از مقدار دامین سایت باید با مقدار مشخص شده در خاصیت مسیر شروع شود. مثلا مسیر یک کوکی برابر post/ و دامین آن نیز برابر www.dotnettips.info باشد، این کوکی در آدرسهایی چون www.dotnettips.info/post/1286 و یا www.dotnettips.info/post/1 و یا www.dotnettips.info/post/test/test2 و ... در دسترس خواهد بود. برای کسب اطلاعات بیشتر میتوان به RFC 6265 (قسمت Paths and Path-Match) مراجعه کرد.

برای روشنتر شدن مطلب به هدرهای Set-Cookie زیر توجه کنید:

```
... Set-Cookie: MyCookie1=hi; Domain=d1.d2.com; Path=/employee
... /=Set-Cookie: MyCookie2=bye; Domain=.d3.com; Path
... Set-Cookie: MyCookie3=nth
.....
```

اولین دستور به مرورگر میگوید تا یک کوکی با نام MyCookie1 و مقدار hi را با دامین d1.d2.com و مسیر employee/ ثبت کند.
بنابراین مرورگر از این کوکی تنها درصورتیکه آدرس درخواست موردنظر شامل d1.d2.com/employee باشد، استفاده میکند.
دستور دوم به مرورگر میگوید تا از کوکی MyCookie2 در مسیرهای شامل d3.com. استفاده کند.
در دستور سوم دامین و مسیر با توجه به آدرس صفحه جاری تنظیم میشود. در واقع تنها مسیرهایی که شامل آدرس صفحه جاری باشند به این کوکی دسترسی دارند.

نکته: مقدار domain تنها میتواند مربوط به دامین اصلی جاری و یا زیرمجموعه‌های آن باشد. یعنی نمیتوان یک کوکی با دامین www.d1.com در صفحه‌ای با آدرس www.d2.com ایجاد کرد.

نکته: همچنین کوکی هایی که مثلا دارای دامین www.dotnettips.info هستند از آدرسی نظیر my.dotnettips.info در دسترس نیستند. کوکی‌ها تنها در دامین و ساب دامین‌های مربوط به خود قابل خواندن هستند.

نکته: اگر مقدار خاصیت domain کوکی به چیزی شبیه dotnettip.info تنظیم شود آنگاه این کوکی در آدرسهایی چون www.dotnettips.info و یا d1.dotnettips.info نیز در دسترس است.

نکته: اگر برای خاصیت path مقدار / تنظیم شود، بدین معنی است که کوکی در تمام محدوده دامین کوکی در دسترس است.

3. Secure و HttpOnly

این دوخاصیت برخلاف خواص قبلی مقداری را تنظیم نمیکنند! بلکه همانند یک flag عمل کرده که هر کدام رفتار خاصی را برای مرورگر الزام میکنند.
خاصیت Secure مرورگر را مجبور به استفاده از ارتباطات امن و کدگذاری شده (Https) برای تبادل داده‌های کوکی میکند.
درضمن طبیعی است که وب سرور دستور ثبت چنین کوکی‌هایی را خود از طریق یک ارتباط امن به مرورگر ارسال کند تا مبادا طی یک فرایند خرابکارانه داده‌های مهم درون کوکی در بین راه دزدیده نشود.

نکته: یک کوکی Secure تنها درصورتی به سمت سرور ارسال میشود که درخواست مذکور با استفاده از SSL و ازطریق پروتوکل HTTPS ایجاد شده باشد.

خاصیت HttpOnly به مرورگر اعلام میکند که استفاده از این کوکی تنها در ارتباطات از نوع پروتوکل HTTP مجاز است. بنابراین سایر روشهای دسترسی موجود (مثل document.cookie در جاوا اسکریپت) برای این نوع کوکی‌ها کار نخواهد کرد. درواقع نحوه برخورد با این نوع کوکی‌ها در سمت سرور با سایر انواع کوکی تفاوتی ندارد و تنها در سمت کلاینت و در مرورگر است که رفتاری متفاوت متوجه این کوکی‌ها میشود و اجازه دسترسی به برنامه‌های سمت کلاینت داده نمیشود.

نکته: این خاصیت ابتدا توسط مایکروسافت در نسخه IE 6 SP1 معرفی شد و بعدها بتدریج توسط سایر مرورگرها نیز پشتیبانی شد. این ویژگی همانطور که از آن برمی‌آید برای مقابله با حملات XSS پیاده سازی شده است. البته علاوه برای جلوگیری از دسترسی به این کوکی‌ها از طریق document.cookie، در مرورگرهای مدرن از دسترسی به هدر این کوکی‌ها ازطریق متدهای شی XMLHttpRequest نیز جلوگیری میشود.

نکته: امکان تنظیم این خاصیت از طریق document.cookie در جاوا اسکریپت وجود ندارد!

مدیریت کوکی‌ها در مرورگر

همانطور که قبلا اشاره شد هویت یک کوکی با استفاده تمامی خواص آن به جز expires یا max-age مشخص میشود. یعنی ترکیب name-domain-path-secure/httponly هویت یک کوکی را منحصربه‌فرد میکند. بنابراین تا زمانیکه حتی یکی از این خواص دو کوکی با هم فرق داشته باشد این دو کوکی از هم متمایز خواهند بود. دو کوکی زیر را درنظر بگیرید:

```
Set-Cookie: cookie1=value1
Set-Cookie: cookie1=value2; domain=dotnettips.info;
```

با اینکه به نظر میرسد که این دو کوکی یکسان هستند و اجرای دستور دوم موجب بازنویسی کوکی اول میشود، اما بررسی یک درخواست ارسالی از این صفحه نشان میدهد که دو کوکی مجزا با نام مشابه به سمت سرور ارسال میشود:

Cookie: cookie1=value1; cookie1=value2

حال اگر کوکی سومی به صورت زیر تعریف شود:

Set-Cookie: cookie1=value3; domain=dotnettips.info; path=/post

وضعیت از این نیز پیچیده‌تر میشود:

Cookie: cookie1=value1; cookie1=value2; cookie1=value3

بنابراین اجرای دستور زیر در همان صفحه:

Set-Cookie: cookie1= **value4**

تنها مقدار کوکی اول را تغییر خواهد داد. یعنی در درخواست ارسالی به سمت سرور خواهیم داشت:

Cookie: cookie1= **value4** ; cookie1=value2; cookie1=value3

بنابراین دقت مضاعف به این نکته که «هویت یک کوکی با استفاده از تمامی خواص آن به جز expires یا max-age تعیین میشود» مهم است. برای قسمت «به جز expire یا max-age» هم به مثال زیر توجه کنید:

Set-Cookie: cookie2=value1; max-age=1000

بنابراین خواهیم داشت:

Cookie: cookie1=value1; cookie1=value2; cookie1=value4; cookie2=value1

یک کوکی با طول عمر 1000 ثانیه تولید میکند. بنابراین با دستور زیر میتوان مقدار همین کوکی را تغییر داد:

Set-Cookie: cookie2= **value2**

پس داریم:

Cookie: cookie1=value4; cookie1=value2; cookie1=value3; cookie2= **value2**

هرچند در دستور آخر به نظر میرسد که کوکی آخر به نوع سشنی تغییر یافته است (چون خاصیت expires یا max-age ندارد) اما درواقع این چنین نیست. تنها اتفاقی که رخ داده است این است که مقدار کوکی مذبور تغییر یافته است، درحالیکه تغییری در خاصیت expires یا max-age آن رخ نداده است.

نکته: با تغییر خواص یک کوکی، میتوان آنرا از نوع سشنی به نوع ماندگار (Persistent) تغییر داد، اما عکس این عملیات ممکن **نیست** .

SubCookie

بدلیل محدودیت موجود در تعداد کوکی‌ها به ازای هر دامین، روشی برای نگهداری تعداد بیشتری تنظیمات درون همین تعداد محدود کوکی‌ها توسط توسعه گران ابداع شده است. در این روش از طرح ساده ای که نمونه ای از آن در زیر نشان داده شده است برای نگهداری داده‌های چندین کوکی درون یک کوکی استفاده میشود:

/=Set-Cookie: cookieName=cookie1=value1&cookie2=value2&cookie3=value3&cookie4=value4; path

در نمونه بالا با اینکه عملا تنها یک کوکی تعریف شده است اما درواقع داده‌های 4 کوکی مختلف درون یک کوکی آورده شده است. تنها عیب این روش این است که زحمت بیشتری برای استخراج داده‌های کوکی‌ها باید کشید. البته امروزه برخی از فریمورکها امکاناتی جهت کار با این کوکی‌ها فراهم کرده اند.

Cookie2

در ابتدای هزاه سوم! مدتی بحثی مطرح شد برای بهبود کارایی و امنیت کوکی‌ها و پیشنهادهایی مبنی بر پیاده سازی نوع جدیدی از کوکی‌ها با عنوان Cookie2 نیز ارائه شد. حتی در نسخه جدیدی از استانداردهای HTTP State Management Mechanism که در RFC 2965 آورده شده است کلا به این نوع جدید از کوکی‌ها پرداخته شده است. هرچند برخی از مرورگرها پشتیبانی از این نوع

جدید را آغاز کردند (مثل Opera) اما بعدها به دلیل عدم استقبال از آن، این نوع از کوکی‌ها منسوخ شد و حالت آن در نسخه‌های جدید استانداردها به Obsolete تغییر یافت (RFC 6265)! درهرصورت برای آشنایی با این نوع کوکی‌ها میتوان به مراجع زیر رجوع کرد: (Cookie2 (The Grinder Documentation

RFC2965

?What is the current state of the Cookie2 specification

منابع: HTTP cookie

All About Cookies

Cookies

HTTP cookies explained

The Unofficial Cookie FAQ

نظرات خوانندگان

نویسنده: saleh
تاریخ: ۱۳۹۲/۰۲/۰۶ ۰:۲۷

مطلب بسیار خوب و کاملی بود ممنون

نویسنده: میثم هوشمند
تاریخ: ۱۳۹۲/۰۲/۰۶ ۲۳:۵۶

رسما مطلب دیگری برای گفتن باقی نگذاشتید! خیلی ممنونم بابت توضیحات کامل!

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۰۹ ۲۳:۴۶

البته هنوز مطالب زیادی درباره کوکی‌ها مونده که در قسمتهای بعدی به اونا پرداخته میشه.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۲/۰۲/۱۵ ۱۹:۵۱

داشتم مقالتونو میخوندم که به مشکل پروژه‌ی خودم برخوردم و مقاله‌ی شما باعث شد مشکل منم حل بشه و کلا بفهمم که کوکی دقیقا چیه :دی
خیلی ممنون :)

نویسنده: احمد
تاریخ: ۱۳۹۲/۰۵/۰۲ ۱۶:۴۸

سلام
چند تا سوال:
-1چرا expiration مربوط به کوکی رو مرورگر در مراجعات بعدی به سرور نمیفرسته؟اگه لازمش داشته باشیم چه جور باید بدستش بیاریم؟
-2اگه زمان expire مربوط به کوکی برسه کی مسئولیت نابودی cookie رو داره؟
-3مسیر دخیره سازی کوکی‌ها کجا هست؟چه زمانی که IsPersistent برابر true یا false باشه(هر چی میگردم پیداشون نمیکنم)؟
البته مورد 3 رو در استفاده formsauthentication بررسی کردم طبق همون نامی که توی تگ forms توی web.config تنظیم شده.

عنوان: **Cookie - قسمت دوم**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۱۳ ۲۲:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, Cookie

کوکی در جاوا اسکریپت

همانطور که در قسمت قبل اشاره کوتاهی شد، مدیریت کوکی‌های در دسترس در وضعیت جاری، در جاوا اسکریپت ازطریق پراپرتی cookie از شی document امکان‌پذیر است. این پراپرتی کاری همانند هدرهای Set-Cookie و Cookie (که در قسمت قبل درباره آن‌ها بحث شد) انجام می‌دهد. این پراپرتی یک مورد کاملا استثنایی و نسبتا عجیب در زبان جاوا اسکریپت است. در نگاه اول ظاهرا document.cookie از نوع رشته است، اما قضیه کاملا فرق می‌کند. برای روشن شدن مطلب به ادامه بحث توجه کنید.

افزودن کوکی

- برای افزودن یا ویرایش یک کوکی باید از ساختاری مانند ساختار هدر Set-Cookie که چیزی شبیه به عبارت زیر است، پیروی کرد:

```
document.cookie = "name=value; expires=date; domain=theDomain; path=thePath; secure";
```

نکته: با توجه به توضیحاتی که در قسمت قبل ارائه شد، بدیهی است که امکان ثبت یک کوکی با فلگ HttpOnly در جاوا اسکریپت وجود ندارد!

اجرای دستوری شبیه با ساختار نشان داده شده در بالا، موجب حذف کوکی‌های قبلی نمی‌شود. از این دستور برای ایجاد یک کوکی و یا ویرایش یک کوکی موجود استفاده می‌شود. کوکی‌های ایجادشده با این روش تفاوتی با کوکی‌های ایجادشده توسط هدر Set-Cookie ندارند و همانند آنها در درخواست‌های بعدی با توجه به خواص تنظیم شده، به سمت سرور ارسال خواهند شد.

همانطور که مشاهده می‌کنید خاصیت‌های کوکی به صورت جفت‌های نام-مقدار درون یک رشته به document.cookie نسبت داده می‌شوند. این خاصیت‌ها توسط یک کاراکتر ; از یکدیگر جدا می‌شوند. شرح ساختار فوق در زیر آورده شده است:

1. همیشه اولین جفتِ نام-مقدار همانند مثال بالا **باید** «عنوان و مقدار» کوکی را مشخص سازد. این قسمت تنها عضو اجباری ساختار فوق است.
2. سپس یک سمی‌کالن و یک فاصله
3. تاریخ انقضا (expires) **یا** حداکثر طول عمر کوکی (max-age)
4. سپس یک سمی‌کالن و یک فاصله
5. دمین و یا مسیر مربوط به کوکی
6. سپس یک سمی‌کالن و یک فاصله
7. سایر خواص چون Secure

نکته: این ساختار عجیب معرفی شده را **عینا** رعایت کنید. بقیه کار توسط مرورگر انجام خواهد شد.

نکته: قسمت‌های مختلف این ساختار case-sensitive **نیست،** البته به‌جز نام کوکی که کاملا case-sensitive است.

مثلا برای ثبت یک کوکی با عنوان myCookie و مقدار myValue و دمین d.com و مسیر test و طول عمر 5 روزه باید از دستور زیر استفاده کرد:

```
document.cookie = 'myCookie=myValue; max-age=432000; domain=d.com; path=/test';
```

خواندن کوکی

- برای خواندن کوکی‌ها تنها کافی است مقدار پراپرتی document.cookie بررسی شود. با اینکه از دستور نشان داده شده در بالا اینگونه برمی آید که پراپرتی document.cookie به رشته معرفی شده مقداردهی شده است، اما به محض خواندن این پراپرتی چیزی شبیه به عبارت زیر برگردانده میشود:

```
myCookie=myValue
```

از بقیه خواص اثری نیست! این رفتار به دلیل حفظ امنیت کوکی‌ها در تمام مرورگرها رعایت می‌شود.

- برای ثبت کوکی دیگری در وضعیت جاری کافی است یکبار دیگر دستور بالا را برای کوکی جدید به کار ببریم. مثلا به صورت زیر:

```
document.cookie = 'mySecondCookie=mySecondValue; path=/'
```

اینار یک کوکی سشنی بدون دمین و با مقدار / برای مسیر کوکی ثبت می‌شود! در این حالت کوکی قبلی دوباره نویسی و یا حذف نمی‌شود و تنها یک کوکی جدید به لیست کوکیهای مرورگر اضافه می‌شود! این رفتار عجیب از ویژگی‌های جالب document.cookie است.
- اگر مقدار document.cookie در این حالت خوانده شود مقدار زیر برگشت داده می‌شود:

```
myCookie=myValue; mySecondCookie=mySecondValue
```

باز هم خبری از سایر خاصیت‌ها نیست. ولی همانطور که می‌بینید کوکی دوم به لیست کوکی‌های مرورگر اضافه شده است.

نکته: عبارت برگشت داده شده از پراپرتی document.cookie همانند مقداری است که در هدر Cookie هر درخواست توسط مرورگر گنجانده می‌شود، یعنی جفت نام-مقدار کوکی‌ها به همراه یک ; و یک فاصله بین مقادیر هر کوکی. بنابراین برای بدست آوردن مقدار یک کوکی یکسری عملیات جهت Parse کردن داده‌های آن نیاز است!

متدها

امروزه کتابخانه‌های متعددی با استفاده از زبان جاوا اسکریپت برای برنامه نویسی سمت کلاینت وجود دارد که بیشتر آنها قابلیت‌هایی برای کار با کوکی‌ها نیز دارند. ازجمله می‌توان به jQuery و YUI اشاره کرد. پلاگین مخصوص کوکی‌ها در jquery اینجا بحث شده است. برای کسب اطلاعات بیشتر درباره قابلیت‌های کار با کوکی در YUI نیز به اینجا مراجعه کنید. مطالب زیر صرفا برای روشن شدن بحث ارائه می‌شوند. بدیهی است که برای کارهای عملی بهتر است از کتابخانه‌های موجود استفاده شود. با توجه به اطلاعات بالا از متدهای زیر می‌توان برای خواندن، افزودن و حذف کوکی‌ها استفاده کرد.

نکته: متدهای زیر از ترکیب چندین ریفرنس مختلف بدست آمده است. هرچند برای موارد خاص‌تر می‌توانند بیشتر سفارشی شوند.

افزودن و یا ویرایش کوکی

```
function setCookie(data, value) {
  if (typeof data === "string") {
    data = { name: data, value: value };
  };
  if (!data.name) throw "Cookie's name can not be null.";

  var cookie = escape(data.name) + "=" + escape(data.value);

  var expDate = null;
  if (data.expDays) {
    expDate = new Date();
    expDate.setDate(expDate.getDate() + data.expDays);
  }
  else if (data.expYear && data.expMonth && data.expDay) {
    expDate = new Date(data.expYear, data.expMonth, data.expDay);
  }
  else if (data.expires) {
    expDate = data.expires;
  }
  else if (data.maxAge) {
    expDate = new Date();
```

```
    expDate.setSeconds(expDate.getSeconds() + data.maxAge);
  }
  if (expDate != null) cookie += "; expires=" + expDate.toGMTString();

  if (data.domain)
    cookie += "; domain=" + escape(data.domain);

  if (data.path)
    cookie += "; path=" + escape(data.path);

  if (data.secure)
    cookie += "; secure";

  document.cookie = cookie;
  return document.cookie;
}
```

در کد فوق برای انکد کردن رشته‌های مورد استفاده از متد escape استفاده شده است. برای آشنایی با این متد به اینجا مراجعه کنید.
هم‌چنین کار کردن با نوع داده تاریخ در جاوا اسکریپت کمی متفاوت است. بنابراین برای آشنایی بیشتر با این نوع داده به اینجا رجوع کنید.
نکته: در متد بالا بدلیل عدم پشتیبانی از خاصیت max-age در نسخه‌های قدیمی اینترنت اکسپلورر (نسخه 8 و قبل از آن) تنها از خاصیت expires استفاده شده است.
نحوه استفاده از متد بالا به صورت زیر است:

```
setCookie('cookie1', 'Value1');
setCookie({name:'cookie1', value:'Value1'});
setCookie({name:'cookie2', value:'Value2', expDays:10});
setCookie({name:'cookie3', value:'Value3', expires:new Date()});
setCookie({name:'cookie4', value:'Value4', expYear:2013, expMonth:0, expDay:13});
setCookie({name:'cookie3', value:'Value3', maxAge:365*24*60*60});
setCookie({name:'cookie5', value:'Value5', domain:'d.net', path:'/'});
setCookie({name:'cookie6', value:'Value6', secure:true});
setCookie({name:'cookie7', value:'Value7', expDays:100, domain:'dd.com', path:'/employee',
secure:true});
```

حذف کوکی

همانطور که در قسمت قبل هم اشاره شد، برای حذف یک کوکی، کافی است تا تاریخ انقضای آن به تاریخی در گذشته مقداردهی شود. بنابراین برای اینکار می‌توان از متد زیر استفاده کرد:

```
function delCookie(data) {
  if (typeof data === "string") {
    data = { name: data };
  };
  data.expDays = -1;
  return setCookie(data);
}
```

در متد فوق از متد setCookie که در بالا معرفی شد، استفاده شده است. نحوه استفاده از این متد هم به صورت زیر است:

```
delCookie('myCookie');
delCookie({ name: 'myCookie', domain: 'd.com', path: '/test' });
```

خواندن کوکی

برای خواندن مقدار یک کوکی می‌توان از متد زیر استفاده کرد:

```
function getCookie(name) {
  var cookies = document.cookie.split(";");
  for (var i = 0; i < cookies.length; i++) {
    var cookie = cookies[i].split("=");
    if (cookie[0].trim() == escape(name)) {
      return unescape(cookie[1].trim());
    }
```

```
  }
  return null;
}
```

برای آشنایی با متد unescape که در بالا از آن استفاده شده است به اینجا مراجعه کنید. در متد فوق از متد trim زیر استفاده شده است:

```
String.prototype.trim = function () {
  return this.replace(/^\s+|\s+$/g, "");
};
```

```
String.prototype.trimStart = function () {
  return this.replace(/^\s+/, "");
};
String.prototype.trimEnd = function () {
  return this.replace(/\s+$/, "");
};
```

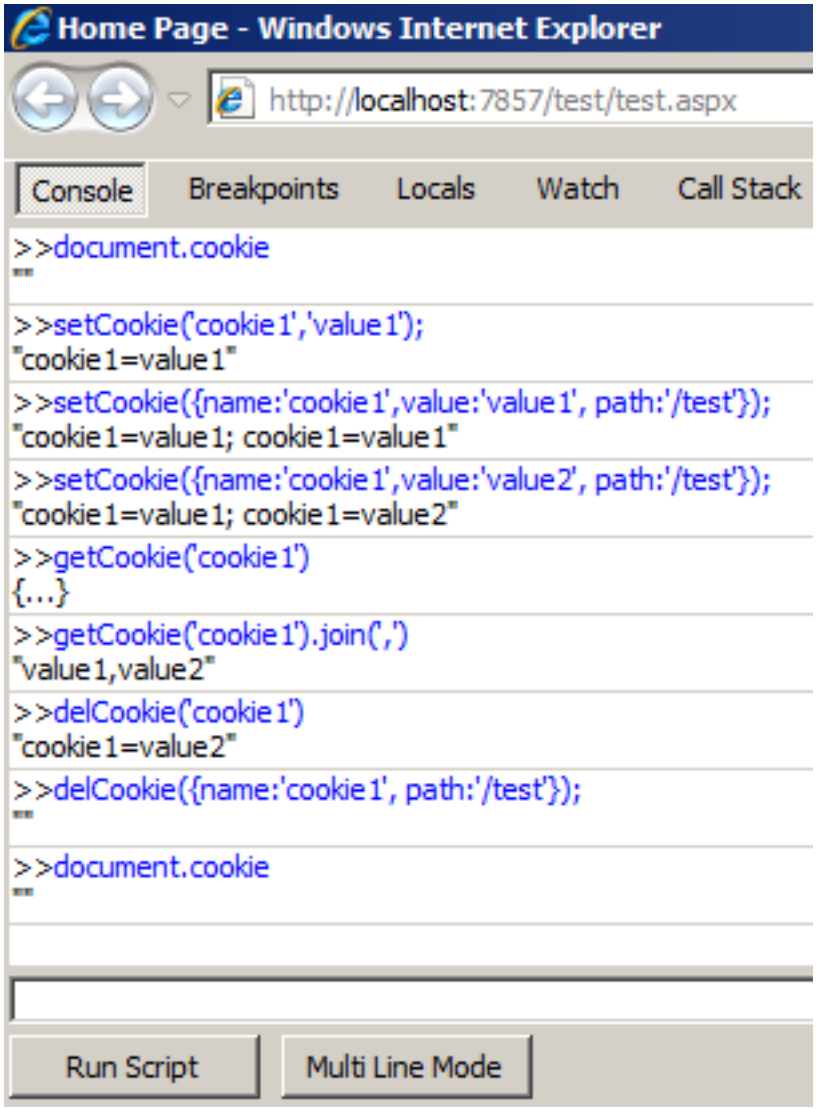
این متدها از اینجا گرفته شده است.
روش استفاده شده برای خواندن مقادیر کوکی‌ها در متد بالا بسیار ساده و ابتدایی است و صرفا برای آشنایی با نحوه Parse کردن رشته برگشت داده شده توسط document.cookie ارائه شده است. روش‌های مناسب‌تر و مطمئن‌تر با یک جستجوی ساده در دسترس هستند. البته همانطور که قبلا هم اشاره شد، استفاه از کتابخانه‌های موجود راه‌حل بهتری است.
هم‌چنین ازآنجاکه مقدار یک کوکی می‌تواند شامل کاراکتر = نیز باشد، بنابراین قسمت return متد فوق را می‌توان به صورت زیر تغییر داد:

```
cookie.shift(1);
return unescape(cookie.join('=').trim());
```

نکته: با توجه به مطالب ارائه شده در قسمت قبل بدست آوردن مقادیر کوکی‌ها کمی پیچیده‌تر از دیگر عملیات‌هاست. ازآنجاکه راه مستقیمی با استفاده از جاوا اسکریپت برای یافتن سایر خواص کوکی وجود ندارد، بنابراین بدست آوردن مقدار دقیق کوکی موردنظر ممکن است غیرممکن باشد! (با توجه به اینکه کوکی‌های متفاوت می‌توانند نام‌های یکسانی داشته باشند).
با توجه به نکته بالا، حال اگر با یک نام بخصوص، چندین کوکی ثبت شده باشد (با خواص متفاوت)، یکی از راه‌حل‌ها این است که آرایه‌ای از مقادیر این کوکی‌های همنام برگشت داده شود. بنابراین متد فوق را می‌توان به صورت زیر تکمیل کرد:

```
function getCookie(name) {
  var foundCookies = [];
  var cookies = document.cookie.split(";");
  for (var i = 0; i < cookies.length; i++) {
    var cookie = cookies[i].split("=");
    if (cookie[0].trim() == escape(name) && cookie.length >= 2) {
      cookie.shift(1);
      foundCookies.push(unescape(cookie.join('=').trim()));
    }
  }
  return foundCookies.length > 1
           ? foundCookies
           : foundCookies.length == 1
              ? foundCookies[0]
              : null;
}
```

خلاصه‌ای از نحوه استفاده از متدهای بالا در IE8 (برای نمایش اجرای درست در مرورگری قدیمی!) در تصویر زیر نشان داده شده است:



نکته: کار توسعه این متدها را میتوان برای پشتیبانی از SubCookieها نیز ادامه داد، اما به دلیل دورشدن از مبحث اصلی، این موضوع در این مطلب ارائه نمیشود (درباره این نوع از کوکی‌ها در قسمت قبل شرح کوتاهی داده شده است). اگر علاقه‌مند و نیازمند به این نوع کوکی‌ها هستید، کتابخانه YUI پشتیبانی کاملی از آنها ارائه میکند.
در قسمت بعدی به نکات کار با کوکی در ASP.NET میپردازیم.

منابع: http://www.w3schools.com/js/js_cookies.asp
http://www.quirksmode.org/js/cookies.html
https://developer.mozilla.org/en-US/docs/DOM/document.cookie http://www.nczonline.net/blog/2009/05/05/http-cookies-explained

عنوان: **Cookie - قسمت سوم**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۳/۳۱ ۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Cookie, Http Cookie

Cookie - قسمت اول : مقدمه، تاریخچه، معرفی، و شرح کامل

Cookie - قسمت دوم : کوکی در جاوا اسکریپت

نکته مهم: خواندن قسمت‌های قبلی این سری (مخصوصا قسمت اول) برای درک بهتر مطالب پیشنهاد می‌شود.

.

کوکی در ASP.NET - بخش اول

در قسمت‌های قبلی مقدمات و مباحث کلی راجع به کوکی‌ها و انواع آن، شرح کامل خواص، نحوه رفتار مرورگرها با انواع کوکی‌ها و درنهایت نحوه کار کردن با کوکی‌ها در سمت کلاینت با استفاده از زبان محبوب جاوا اسکریپت پرداخته شد.

در ادامه این سری مطالب به نحوه برخورد ASP.NET با کوکی‌ها و چگونگی کار کردن با کوکی در سمت سرور آشنا خواهیم شد. در بخش اول این قسمت مباحث ابتدایی و اولیه برای کار با کوکی‌ها در ASP.NET ارائه می‌شود. در بخش دوم مباحث پیشرفته‌تر همچون SubCookieها در ASP.NET و نیز سایر نکات ریز کار با کوکی‌ها در ASP.NET بحث خواهد شد.

.

Response و Request در ASP.NET

در قسمت اول این سری به مفاهیم Http Response و Http Request اشاره کوتاهی شده بود. به‌صورت خلاصه، درخواستی که از

سمت یک کلاینت به یک وب سرور ارسال می‌شود **Request** و پاسخی که وب سرور به آن درخواست می‌دهد **Response** نامیده می‌شود.

در ASP.NET، کلیه اطلاعات مرتبط با **درخواست** رسیده از سمت یک کلاینت در نمونه‌ای منحصر به فرد از کلاس HttpRequest نگه‌داری می‌شود. محل اصلی نگه‌داری این نمونه در پراپرتی Request از نمونه جاری کلاس System.Web.HttpContext (قابل دسترسی ازطریق HttpContext.Current) است. البته کلاس Page هم یک پراپرتی با نام Request دارد که دقیقا از همین پراپرتی کلاس HttpContext استفاده می‌کند.

هم‌چنین کلیه اطلاعات مرتبط با **پاسخ** ارسالی وب سرور به سمت کلاینت مربوطه در نمونه‌ای از کلاس HttpResponse ذخیره می‌شود. محل اصلی نگه‌داری این نمونه نیز در پراپرتی Response از نمونه جاری کلاس HttpContext است. همانند Request، کلاس Page یک پراپرتی با نام Response برای نگه‌داری این نمونه دارد که این هم دقیقا از پراپرتی متناظر در کلاس HttpContext استفاده می‌کند.

.

.

کوکی‌ها در Response و Request

هر دو کلاس HttpResponse و HttpRequest یک پراپرتی با عنوان Cookies (^ و ^) دارند که مخصوص نگهداری کوکی‌های مربوطه هستند. این پراپرتی از نوع System.Web.HttpCookieCollection است که یک کالکشن مخصوص برای ذخیره کوکی‌هاست.

- این پراپرتی (Cookies) در کلاس HttpRequest محل نگه‌داری کوکی‌های ارسالی توسط مرورگر در درخواست متناظر آن است. کوکی‌هایی که مرورگر با توجه به شرایط جاری و تنظیمات کوکی‌ها اجازه ارسال به سمت سرور را به آن‌ها داده و در درخواست ارسالی ضمیمه کرده است (با استفاده از هدر :Cookie که در قسمت اول شرح داده شد) و ASP.NET پس از پردازش و Parse داده‌ها، درون این پراپرتی اضافه کرده است.

- این پراپرتی (Cookies) در کلاس HttpResponse محل ذخیره کوکی‌های ارسالی از وب سرور به سمت مرورگر کلاینت در پاسخ به درخواست متناظر است. کوکی‌های درون این پراپرتی پس از بررسی و استخراج داده‌های موردنیاز توسط ASP.NET در هدر پاسخ ارسالی ضمیمه خواهند شد (با استفاده از هدر :Set-Cookie که در قسمت اول توضیح داده شد).

.

.

ایجاد و به‌روزرسانی کوکی در ASP.NET

برای ایجاد یک کوکی و ارسال آن به سمت کلاینت همان‌طور که در بالا نیز اشاره شد، باید از پراپرتی Cookies از کلاس HttpContext استفاده کرد. برای ایجاد یک کوکی روش‌های مختلفی وجود دارد.
در روش اول با استفاده از ویژگی مخصوص **ایندکسر** کلاس HttpCookieCollection عملیات تولید کوکی انجام می‌شود. در این روش، ابتدا بررسی می‌شود که کوکی موردنظر در لیست کوکی‌های جاری وجود دارد یا خیر. درصورتی‌که با این نام قبلا یک کوکی ثبت شده باشد، مقدار کوکی موجود بروزرسانی خواهد شد. اما اگر این نام وجود نداشته باشد یک کوکی جدید با این نام به لیست افزوده شده و مقدار آن ثبت می‌شود. مثال:

```
HttpContext.Current.Response.Cookies["myCookie"].Value = "myCookieValue";
```

روش بعدی استفاده از متد Add در کلاس HttpCookieCollection است. در این روش ابتدا یک نمونه از کلاس HttpCookie ایجاد شده و سپس این نمونه به لیست کوکی‌ها اضافه می‌شود. کد زیر چگونگی استفاده از این روش را نشان می‌دهد:

```
var myCookie = new HttpCookie("myCookie", "myCookieValue");
HttpContext.Current.Response.Cookies.Add(myCookie);
```

روش دیگر استفاده از متد Set کلاس HttpCookieCollection است. تفاوت این متد با متد Add در این است که متد Set ابتدا سعی می‌کند عملیات update انجام دهد. یعنی عملیات افزودن تنها وقتی‌که نام کوکی موردنظر در لیست کوکی‌ها یافته نشود انجام خواهد شد. برای مثال:

```
HttpContext.Current.Response.Cookies.Set(new HttpCookie("myCookie", "myCookieValue"));
```

نکته: باتوجه به توضیحات بالا، متد Set اجازه افزودن دو کوکی با یک نام را نمی‌دهد. برای اینکار باید از متد Add استفاده کرد. درباره این موضوع در قسمت بعدی بیشتر توضیح داده خواهد شد.
روش دیگری که برای ایجاد یکی کوکی می‌توان از آن استفاده کرد، بکارگیری متد AppnedCookie از کلاس HttpResponse است. در این روش نیز ابتدا باید یک نمونه از کلاس HttpCookie تولید شود. این روش همانند استفاده از متد Add از کلاس HttpCookieCollection است. کد زیر مثالی از این روش را نشان می‌دهد:

```
HttpContext.Current.Response.AppendCookie(new HttpCookie("myCookie", "myCookieValue"));
```

روش بعدی استفاده از متد SetCookie از کلاس HttpResponse است. فرق این متد با متد AppendCookie در این است که در متد SetCookie ابتدا وجود یک کوکی با نام ارائه شده بررسی می‌شود و درصورت وجود، مقدار این کوکی بروزرسانی می‌شود. درصورتی‌که قبلا یک کوکی با این نام وجود نداشته باشد، یک کوکی جدید به لیست کوکی‌ها اضافه می‌شود. این روش همانند استفاده از متد Set از کلاس HttpCookieCollection است. نمونه‌ای از نحوه استفاده از این متد در زیر آورده شده است:

```
HttpContext.Current.Response.SetCookie(new HttpCookie("myCookie", "myCookieValue"));
```

نکته: تمامی فرایندهای نشان داده شده در بالا تنها موجب تغییر محتویات کالکشن کوکی‌ها درون HttpContext می‌شود و تا زمانی‌که توسط وب سرور با استفاده از دستور Set-Cookie به سمت مرورگر ارسال نشوند تغییری در کلاینت بوجود نخواهند آورد.

برای آشنایی بیشتر با این روند کد زیر را برای تعریف یک کوکی جدید درنظر بگیرید:

```
HttpContext.Current.Response.Cookies["myCookie"].Value = "myValue";
```

برای مشاهده هدر تولیدی توسط وب سرور می‌توان از نرم افزار محبوب Fiddler استفاده کرد (از اواخر سال 2012 که نویسنده این ابزار به Telerik پیوسته، توسعه آن بسیار فعال‌تر شده و نسخه‌های جدید با لوگوی جدید! ارائه شده است).
تصویر زیر مربوط به مثال بالاست:

همانطور که مشاهده می‌کنید دستور ایجاد یک کوکی با نام و مقدار وارده در هدر پاسخ تولیدی توسط وب سرور گنجانیده شده است.

نکته: در ASP.NET به صورت پیش فرض از مقدار "/" برای پراپرتی Path استفاده می‌شود.

.

خواص کوکی در ASP.NET

برای تعیین یا تغییر خواص یک کوکی در ASP.NET باید به نمونه HttpCookie مربوطه دست یافت. سپس با استفاده از پراپرتی‌های این کلاس می‌توان خواص موردنظر را تعیین کرد. برای مثال:

```
var myCookie = new HttpCookie(string.Empty);
myCookie.Name = "myCookie";
myCookie.Value = "myCookieValue";
myCookie.Domain = "dotnettip.info";
myCookie.Path = "/post";
myCookie.Expires = new DateTime(2015, 1, 1);
myCookie.Secure = true;
myCookie.HttpOnly = true;
```

نکته مهم: امکان تغییر خواص یک کوکی به صورت مستقیم در سمت سرور وجود ندارد. درواقع برای اعمال این تغییرات در سمت کلاینت باید به ازای هر کوکی موردنظر یک کوکی جدید با مقادیر جدید ایجاد و به کالکشن کوکی‌ها در Http Response مربوطه اضافه شود تا پس از قرار دادن دستور Set-Cookie متناظر در هدر پاسخ ارسالی به سمت کلاینت و اجرای آن توسط مرورگر، مقادیر خواص مورنظر در سمت کلاینت بروزرسانی شوند. دقت کنید که تمامی نکات مرتبط با هویت یک کوکی که در قسمت اول شرح داده شد در اینجا نیز کاملا صادق است.

روش دیگری نیز برای تعیین **برخی** خواص کوکی‌ها به صورت کلی در فایل وب کانفیگ وجود دارد. برای اینکار از تگ httpCookies

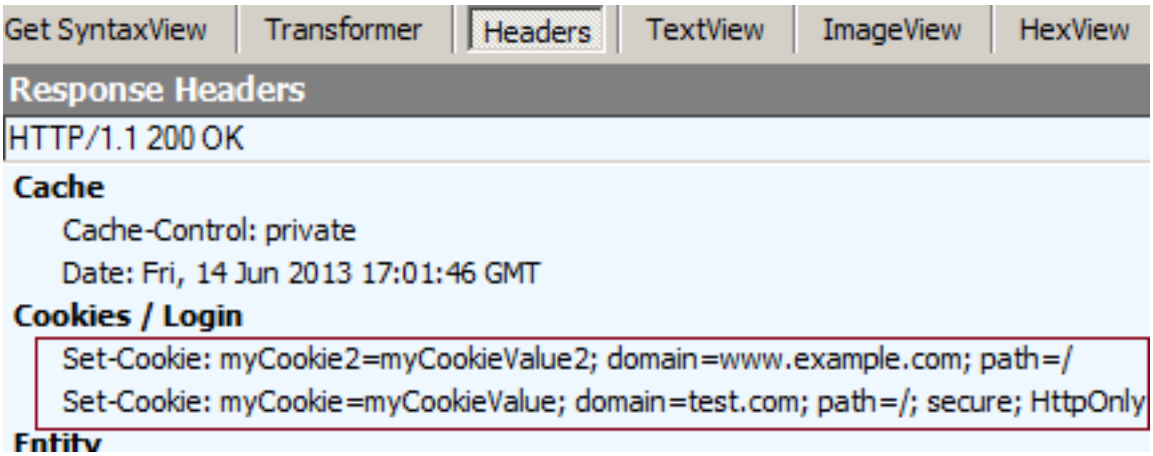
در قسمت system.web استفاده می‌شود. برای مثال:

```
<httpCookies domain="www.example.com" httpOnlyCookies="true" requireSSL="true" />
```

این امکان از ASP.NET 2.0 به بعد اضافه شده است. با استفاده از این تگ، تنظیمات اعمال شده برای تمامی کوکی‌ها درنظر گرفته می‌شود. البته درصورتی‌که تنظیم موردنظر برای کوکی به صورت صریح آورده نشده باشد. برای نمونه به کد زیر دقت کنید:

```
var myCookie = new HttpCookie("myCookie", "myCookieValue");
myCookie.Domain = "test.com";
HttpContext.Current.Response.Cookies.Add(myCookie);
var myCookie2 = new HttpCookie("myCookie2", "myCookieValue2");
myCookie2.HttpOnly = false;
myCookie2.Secure = false;
HttpContext.Current.Response.Cookies.Add(myCookie2);
```

با استفاده از تنظیمات تگ httpCookies که در بالا نشان داده شده است، هدر پاسخ تولیدی توسط وب سرور به صورت زیر خواهد بود:



همان‌طور که می‌بینید تنها مقادیر پراپرتی‌هایی که صراحتا برای کوکی آورده نشده است از تنظیمات وب کانفیگ خوانده می‌شود.

.

.

حذف کوکی در ASP.NET

برای حذف یک کوکی در ASP.NET یک روش کلی وجود دارد که در قسمت‌های قبلی نیز شرح داده شده است، یعنی تغییر خاصیت Expires کوکی به تاریخی در گذشته. برای نمونه داریم:

```
var myCookie = new HttpCookie("myCookie", "myCookieValue");
myCookie.Expires = DateTime.Now.AddYears(-1);
```

نکته مهم: در کلاس HttpCookieCollection یک متد با نام Remove وجود دارد. از این متد برای حذف یک کوکی از لیست موجود

در این کلاس استفاده می‌شود. دقت کنید که حذف یک کوکی از لیست کوکی‌ها با استفاده از این متد تاثیری بر موجودیت آن کوکی در سمت کلاینت نخواهد گذاشت و تنها روش موجود برای حذف یک کوکی در سمت کلاینت همان تنظیم مقدار خاصیت Expires است.

.

خواندن کوکی در ASP.NET

برای خواندن مقدار یک کوکی ارسالی از مرورگر کلاینت در ASP.NET، باتوجه به توضیحات ابتدای این مطلب، طبیعی است که باید از پراپرتی **Request.** Cookies در نمونه جاری از کلاس HttpContext استفاده کرد. برای این کار نیز چند روش وجود دارد.
روش اول استفاده از **ایندکسر** کلاس HttpCookieCollection است. برای اینکار نیاز به نام یا ایندکس کوکی موردنظر در لیست مربوطه داریم. برای مثال:

```
var myCookie = HttpContext.Current.Request.Cookies["myCookie"];
```

یا این نمونه با استفاده از **ایندکسر عددی** :

```
var myCookie = HttpContext.Current.Request.Cookies[0];
```

روش دیگری که برای خواند مقدار یک کوکی می‌توان بکار برد، استفاده از متد Get از کلاس HttpCookieCollection است. این متد همانند ایندکسر این کلاس نیاز به نام یا ایندکس کوکی موردنظر دارد. برای نمونه:

```
var myCookie = HttpContext.Current.Request.Cookies.Get("myCookie");
```

یا استفاده از ایندکس کوکی :

```
var myCookie = HttpContext.Current.Request.Cookies.Get(0);
```

.

.

بحث و نتیجه گیری

تا اینجا با مفاهیم اولیه درباره نحوه برخورد ASP.NET با کوکی‌ها آشنا شدیم. روش‌های مختلف ایجاد و یا به‌روزرسانی کوکی‌ها نشان داده شد. با تعیین انواع خواص کوکی‌ها آشنا شدیم. نحوه حذف یک کوکی در ASP.NET را دیدیم. روش‌های خواندن مقادیر کوکی‌ها را نیز مشاهده کردیم.

باز هم تاکید می‌کنم که تمامی تغییرات اعمالی در سمت سرور تا زمانی‌که به‌صورت دستورات Set-Cookie در هدر پاسخ وب سرور قرار نگیرند هیچ کاری در سمت کلاینت انجام نمی‌دهند.

در قسمت بعدی این سری مطالب به مباحث پیشرفته‌تری چون SubCookieها در ASP.NET و هویت منحصر به فرد کوکی‌ها در سمت سرور پرداخته می‌شود.

.

.

منابع

http://msdn.microsoft.com/en-us/library/ms178194(v=vs.100).aspx

http://msdn.microsoft.com/en-us/library/aa289495(v=vs.71).aspx

http://www.codeproject.com/Articles/31914/Beginner-s-Guide-To-ASP-NET-Cookies

http://www.codeproject.com/Articles/244904/Cookies-in-ASP-NET

نظرات خوانندگان

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۳/۰۸/۰۶ ۱۸:۱۸

یک کوکی ایجاد کردم و یه سری دیتا درون اون قرار دادم. در جایی از سایت برای پرداخت اینترنتی به درگاه بانک متصل میشم اما بعد از برگشت از بانک ، وقتی میخوام مقدار کوکی رو بخونم ، کوکی مقدارش خالیه. حتی دستور Page.User.Identity.IsAuthenticate هم مقدار false بر می‌گردونه.در صورتی که قبل از ارسال به درگاه، کاربر لاگین بوده.
ممنون میشم راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۰۷ ۱۰:۹

در دات نت 4.5 ، مشکل طولانی بودن حاصل BinaryFormatter serialization برطرف شده (January 2013). این مشکل سبب می‌شده تا حاصل RolePrincipal.ToEncryptedTicket بسیار طولانی شده و بیشتر از حد مجاز اندازه قابل ذخیره سازی در یک کوکی شود.
- وصله‌ی نسخه‌ی ویندوز 8 و ویندوز سرور 2012 آن از اینجا قابل دریافت است؛ نسخه‌ی ویندوز 7 و ویندوز سرور 2008 از اینجا .
+ آپدیت ویندوز را روشن کنید تا آخرین به روز رسانی‌ها و نگارش‌های دات نت نصب شده را به صورت خودکار دریافت کنید.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۳/۰۸/۰۷ ۱۲:۳۰

با تشکر؛ فقط یک نکته تکمیلی که فراموش کردم اینکه مشکل مورد نظر مربوط به asp.net web form هست و من از simple membership برای فرآیند احراز هویت استفاده میکنم.با توجه به این مسائل ، انجام مواردی که شما فرمودید برای حل مشکل کفایت میکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۰۷ ۱۲:۳۵

فرقی نمی‌کند. مباحث پایه Forms Authentication برای تمام فناوری‌هایی که از آن استفاده می‌کنند یکسان است.

عنوان: **ایجاد کوکی با jcookie**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۲۴ ۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, Cookie

همانطور که از نامش پیداست jcookie یک پلاگین jquery است. این پلاگین به شما این اجازه را می‌دهد تا هر نوع داده ای را که مایل هستید از قبیل رشته‌ها، آرایه‌ها و object را در قالب json با رمزگذاری base 64 ذخیره نمایید. استفاده از این رمزگذاری باعث کوچکتر شدن حجم کوکی تا 70 درصد می‌شود. در این مقاله شما یاد می‌گیرید که چطور برای ذخیره و بازیابی کوکی از آن استفاده کرده و چگونه در یک زبان سمت سرور، مثل سی شارپ نیز کوکی مورد نظر را با همان فرمت بخوانید. جهت دانلود فایل jcookie به اینجا مراجعه کنید. **ذخیره کوکی**
برای ساخت یک کوکی به روش زیر اقدام می‌کنیم. استفاده از $.jCookies دو خاصیت به نام‌های نام کوکی و مقدار کوکی را name & Value در دسترس ما می‌گذارد:

```
var d = new Date();

                $.jCookies({
                    name: 'dotnettips.info',
                    value: { Title: 'ساخت کوکی با jcookie', Author: 'علی یگانه مقدم', Seen:
d.getDate(), Favorite: true }
                });
```

همانطور که می‌بینید ذخیره اطلاعات توسط jcookie بسیار ساده و راحت بوده و هر نوع داده ای در آن به راحتی قابل ذخیره سازیست. برای مثال می‌توانید اطلاعات یک کلاس را خیلی راحت و سریع با آن ذخیره کنید. به طور پیش فرض تاریخ انقضای کوکی 27 روز بعد از ایجاد آن می‌باشد. در صورتی که تمایل دارید این تاریخ را تغییر دهید یکی از خاصیت‌های seconds,minutes,hours و days در دسترس شماست و مقادیری که جلوی آن‌ها به کارگرفته می‌شود باید نوع صحیح بوده و در صورتی که مقدار نامعتبر وارد شود خاصیت مورد نظر نادیده گرفته می‌شود.

```
$.jCookies ({ name : 'User', value : { username : 'Bob' , level : 5 }, minutes : 60 });
```

برای تغییر پیش فرض‌های ساخت کوکی مانند انقضای 27 روز به عدد پیش فرض خودتان فایل jcookies.js را باز کرده و تنظیمات پیش فرض آن را تغییر بدهید. برای تغییر دنبال کد زیر بگردید:

```
$.jCookies.defaults =
{
name : '',
value : '',
days : 27
}
```

بازیابی کوکی
برای بازیابی کوکی مجددا از $.jCookies استفاده می‌شود ولی تنها باید یک خاصیت get که نام کوکی هست را بنویسید:

```
var values = $.jCookies ({ get: 'dotnettips.info' });
```

در صورتی که نام کوکی‌ای که درخواست کرده اید وجود نداشته باشد یا اینکه تاریخ انقضای آن سر رسیده است و از سیستم کلاینت حذف شده است یا اینکه هنگام درخواست کوکی با خطایی مواجه شده باشد، مقدا برگشتی false خواهد بود و اگر نیاز دارید که بدانید آیا نوع برگشتی false به خاطر خطا بوده است یا خیر یک خاصیت نوع بول به نام error هم اضافه می‌شود:

```
var values = $.jCookies({ get: 'Rutabaga', error: true });
```

در صورتی که خطایی داده شود response مقدار values در مرورگر کروم به شکل زیر خواهد بود. در هر مرورگر نحوه نمایش خطا می‌تواند متفاوت باشد.

```
Error : {
            arguments : undefined,
            message : "Invalid base64 data",
            stack : "—",
```

```
            type : undefined
        }
```

بازیابی همه کوکی ها

در صورتی که به خاصیت get مقدار * را بدهید تمامی کوکی‌ها برگشت داده خواهند شد و به صورت آرایه ای از نام کوکی‌ها در دسترسی خواهند بود:

```
var values = $.jCookies({ get: '*' });
                alert(values["dotnettips.info"].Title);
                alert(values["data2"].Title);
```

حذف کوکی

نحوه کدنویسی حذف کوکی هم دقیقا مشابه خواندن کوکی است؛ با این تفاوت که به جای استفاده از خاصیت get از خاصیت erase استفاده می‌کنیم و با دادن نام کوکی به این خاصیت، کوکی حذف خواهد شد:

```
var value = $.jCookies({ erase: 'dotnettips.info' });
```

در صورتی که کوکی وجود داشته باشد، آن را حذف کرده و مقدار true را برگشت خواهد زد و در صورتی که کوکی وجود نداشته باشد مقدار false را بر میگرداند.

بازیابی کوکی در سمت سرور با سی شارپ

در این روش ما ابتدا با همان دستور معمولی دات نت یعنی page.request.cookie درخواست دریافت کوکی را می‌دهیم ولی از آنجا که در jcookie دو عمل روی داده‌ها صورت گرفته است باید دو کار اضافه‌تر را انجام داد:

برگشت داده‌ها از حالت رمزگذاری base64

داده‌ها در فرمت json هستند و باید به حالتی قابل استفاده در محیط شی گرا تبدیل شوند.

برای بازگردانی از حالت base64 از کلاس و متد Convert.FromBase64String در فضای نام system.convert استفاده می‌کنیم که آرایه ای از نوع بایت را بر میگرداند و از Encoding.UTF8.GetString هم برای decode کردن آرایه به نوع رشته استفاده می‌کنیم. تا به اینجای کار داده‌های ما به صورت یک json خوانا با فرمت string درآمده است. برای دسترسی به داده‌های موجود در این فرمت باید آن‌ها را Deserialize کنیم که این کار را از طریق کلاس JavaScriptSerializer در فضای نام System.Web.Script.Serialization انجام می‌دهیم و از کلاس دیکشنری برای ذخیره داده‌های برگشتی استفاده می‌کنیم که نوع string برای نام خاصیت و نوع آبجکت برای ذخیره مقدار خاصیت خواهد بود. یعنی برای بازگردانی اولین مثال بالا باید داده‌های در نوع دیکشنری به صورت زیر لیست شوند

Title	ایجاد کوکی با jcookie
Author	علی یگانه مقدم
Seen	2015/1/14
Favorite	true

```
byte[] from64 = Convert.FromBase64String(Page.Request.Cookies["dotnettips.info"].Value);
            string json = Encoding.UTF8.GetString(from64);
            Dictionary<string, object> article =new
JavaScriptSerializer().Deserialize<Dictionary<string, object>>(json);
            Page.Response.Write("Title: "+ (string)article["Title"]);
```

پشتیبانی از یونیکد

موقعی که من اولین مثال بالا را نوشتم و مقادیر را به صورت فارسی وارد کردم متوجه شدم که رشته‌های یونیکد را انکود نمی‌کند و در نتیجه زبان فارسی در آن پشتیبانی نمی‌شود. برای همین تغییراتی در فایل js ایجاد کرده و عبارت value قبل از تبدیل به base64 را به صورت utf-16 انکود کردم و در هنگام خواندن کوکی هم به صورت utf-16 دیکود کردم و مشکل زبان فارسی هم در این حالت حل شد. البته کدی که اضافه کردم قابلیت‌های انکودینگ بیشتری هم دارد.

فقط تنها مورد این هست که برای خواندن کوکی در سمت سرور باید یک تغییر کوچک یک کلمه ای بدهیم؛ باید کلمه UTF8 را به

Unicode که می‌شود همان UTF-16 در کد تغییر دهیم، که به کد زیر تغییر خواهد یافت:

```
byte[] from64 = Convert.FromBase64String(Page.Request.Cookies["dotnettips.info"].Value);
            string json = Encoding.Unicode.GetString(from64);
            Dictionary<string, object> article =new
JavaScriptSerializer().Deserialize<Dictionary<string, object>>(json);
            Page.Response.Write("Title: "+ (string)article["Title"]);
```

برای دریافت jcookie با پشتیبانی از زبان فارسی به اینجا مراجعه کنید.
کدهای بالا در فایل زیر قرار گرفته اند. jcookie.zip

عنوان: **WebStorage**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۰۸ ۱۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Cookie, IndexedDB, Webstorage, HTML 5

webstorage تقریبا فناوری جدیدی است که برای نگهداری ثابت داده‌ها بر روی سیستم کاربر استفاده می‌شود. webstorage مزایای زیادی برای برنامه‌های تحت وب دارد. برای مثال با استفاده از آن میتوان فعالیت‌های کاربر را رصد کرد، بدون اینکه کد و دیتابیس سمت سرور را دخالت دهیم. حتی اگر سیستم کاربر آفلاین هم بشود برنامه می‌تواند همچنان به فعالیتش ادامه دهد. در این مقاله به مزایای این روش می‌پردازیم.

WebStorage در برابر کوکی ها

یکی از روش‌های سنتی ذخیره اطلاعات در سیستم کاربر، کوکی‌ها در بستر Http هستند. تفاوت‌های زیادی بین این دو وجود دارد که تعدادی از آن‌ها را در زیر بررسی می‌کنیم:

مکانیزم ذخیره سازی:

کوکی‌ها داده‌های ساخت یافته‌ای هستند که از وب سرور به سمت مرورگر کاربر به عنوان پاسخی در ازای درخواستی ارسال می‌شوند. درخواست و پاسخ کوکی‌ها شامل بخش هدر بوده که اطلاعات آن باعث شناسایی کوکی برای مدیریت و شناسایی آن‌ها می‌گردد تا هر موقع درخواستی صورت بگیرد، به سمت سرور برگشت خواهد خورد.

به طور خلاصه اینکه کوکی‌ها توسط درخواست‌ها و پاسخ‌ها ایجاد یا به روز می‌شوند. در نتیجه داده‌ها چه تغییر کرده باشند چه تغییر نکرده باشند، همیشه بخشی از هدر Http هستند. در سوی دیگر webstorage به طور کامل به صورت کلاینتی پیاده سازی گشته است و وب سرور را درگیر کار خودشان نمی‌کنند و کارایی بهتری را ارائه می‌دهند. از آنجا که همه چیز در خود سیستم کاربر اتفاق می‌افتد، در صورت از دست دادن کانکشن شبکه، کاربر می‌تواند همچنان به فعالیت‌های به روزرسانی و تغییر ادامه دهد.

چند نسخه از مرورگر

کاربری که با وب سایت مدنظر کار میکند میتواند توسط چند مرورگر مختلف یا چند تب و پنجره مختلف به طور همزمان کار کند و اطلاعات آن در همه‌ی مرورگرها و دیگر پنجره‌های آن مرورگر قابل دسترس می‌باشد.

محدودیت حجمی

محدودیت کوکی‌ها و webstorage از نظر حجم ذخیره سازی بین مروگرهای مختلف، متفاوت است. ولی در حالت کلی در بیشتر مرورگرها محدودیت حجم 4 کیلوبایت برای کوکی‌ها موجود است. (این ابزار می تواند نهایت حجمی را که که مرورگر شما از کوکی پشتیبانی می‌کند، نشان دهد.)

در مورد webstorage استاندارد W3C محدودیتی اعلام نکرده است و تصمیم گیری بر سر این موضوع را به سازندگان مرورگرها واگذار کرده است. ولی در حالت کلی حجم بیشتری از کوکی را ذخیره میکند و عموما تا 5 مگابایت توانایی ذخیره سازی وجود دارد. بدین ترتیب حجم آن 124,527% بیشتر از کوکی‌ها است. (این ابزار می‌تواند نهایت حجمی را که مرورگر شما از webstorage پشتیبانی می‌کند، ببینید).

انواع Webstorage

دو نوع متد ذخیره سازی در webstorage داریم:

session storage

local storage

Web Storage type	Lifetime of stored data	Data structure	Data type
sessionStorage	Session only	Key/value pairs	String
localStorage	Persistent	Key/value pairs	String

SessionStorage

داده‌هایی که بدین صورت ذخیره می‌شوند تنها تا زمانی دوام آورده و پایدار هستند که session مرورگر فعال است؛ یعنی تا زمانیکه کاربر در سایت فعلی حضور دارد.
استفاده از این روش برای ذخیره سازی‌های موقت عالی است. برای نمونه مقادیر ورودی فرمی که کاربر در حال کار با آن است، میتواند به طور موقت ذخیره شوند تا زمانی که کاربر بتواند تمامی مراحل را طی کرده، بدون اینکه ارجاعی به دیتابیس سمت سرور داشته باشد. همچنین ذخیره موقت داده‌ها می‌تواند به کاربر کمک کند تا در صورت rereshهای ناگهانی یا بسته شدن‌های ناگهانی مرورگرها، نیازی به ورود مجدد داده‌ها نداشته باشد.

LocalStorage

در این روش داده‌ها با از دست رفتن session مرورگر جاری از بین نمی‌رود و برای بازدیدهای آتی کاربر از سایت، داده‌ها همچنان در دسترس هستند. **پشتیبانی مرورگرها**
وب سایت caniuse گزارش می‌دهد که اکثر مرورگرها پشتیبانی خوبی از webstorage دارند.

Browser	Version
Internet Explorer	**IE 8** and above
Mozilla Firefox	**Firefox 3.5** and above
Google Chrome	**Chrome 4** and above
Apple Safari	**Safari 4** and above
Opera	**Opera 11.5** and above

با اینکه توصیه نامه W3C از پایان کار پیاده سازی این قابلیت خبر میدهد ولی در حال حاضر که این مقاله تدوین شده است هنوز نهایی اعلام نشده است. برای پشتیبانی مرورگرهای قدیمی از webstorage می‌توان از فایل جاوااسکریپتی Store.js کمک گرفت.

مفاهیم امنیتی و محافظت از داده ها

محدودیت‌های حمایتی و حفاظتی webstorage دقیقا همانند کوکی هاست. به این معنی که وب سایت‌های دیگر توانایی اتصال به webstorage سایت دیگری را ندارند. البته این مورد ممکن است برای وب سایت هایی که بر ساب دومین تکیه کرده‌اند ایجاد مشکل کند. برای حل این مسائل می‌توانید از کتابخانه‌های سورس بازی چون Cross Storage که توسط Zendesk ارائه شده است، استفاده کرد.

همانند هر مکانیزم ذخیره سازی سمت کلاینت، مواردی توصیه میگردد که رعایت آن‌ها از لحاظ امنیتی پر اهمیت است. به عنوان نمونه ذخیره‌ی اطلاعات شخصی و موارد حساس توصیه نمی‌گردد؛ چرا که احتمال دسترسی آسان نفوذگران به داده‌های محلی و خواندن آن‌ها وجود دارد.

Data Integrity یا یکپارچگی داده‌ها نیز در نظر گرفته شده است. باید حفاظتی در برابر عدم موفقیت ذخیره سازی داده‌ها نیز وجود داشته باشد. این عدم موفقیت‌ها میتواند به دلایل زیر رخ دهد:
اگر کاربر قابلیت webstorage را غیرفعال کرده باشد.

اگر فضایی برای کاربر باقی نمانده باشد.

با محدودیت حجمی webstorage مواجه شده است.

با مواجه شدن با خطاها یک استثنا صادر می‌شود که می‌توانید آن را دریافت و کنترلی را روی برنامه تحت وب داشته باشید. یک نمونه استثنا QuotaExceededError

IndexedDB

یکی از فرایندهای ذخیره سازی داده‌ها که همان مزایای webstorage را ارائه میدهد indexed Database API است. این قابلیت از HTML 5 اضافه شده است و قسمتی از مشخصات webstorage شناخته نمی‌شود. برای همین مستنداتی در حوزه‌ی webstorage برای آن پیدا نخواهید کرد ولی قابلیت‌هایی فراتر از webstorage دارد.

این قابلیت پیچیدگی بیشتری را نسبت به خود webstorage ایجاد می‌کند، ولی فرصت‌های بسیاری را برای ذخیره سازی داده‌هایی با معماری‌های پیچیده‌تر و رابطه‌ها را می‌دهد. با استفاده از IndexedDB داده‌ها به شکل دیتابیس‌های سمت سرور RDMS ذخیره می‌شوند و این قابلیت را دارید که به سمت آن کوئری هایی مشابه بانک‌های اطلاعاتی سمت سرور را ارسال کنید.

در قسمت آتی نحوه کدنویسی آن را فرا خواهیم گرفت.

نظرات خوانندگان

نویسنده: احمد نواصری
تاریخ: ۱۳۹۴/۰۴/۱۴ ۲:۵۹

آیا روش‌های ذکر شده (Session & Local Storage) برای طراحی یک سبد خرید (در یک پروژه فروشگاه اینترنتی) مناسب هستند؟ اگر مناسب هستند، بهتر از Session معمولی کار میکنند؟

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۱۴ ۳:۷

اینجا را بخوانید

عنوان: **WebStorage: قسمت دوم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۴/۰۹ ۱:۰
آدرس: www.dotnettips.info
گروه‌ها: Cookie, IndexedDB, Webstorage, HTML 5

در این مقاله قصد داریم نحوه‌ی کدنویسی webstorage را با کتابخانه‌هایی که در مقاله قبل معرفی کردیم بررسی کنیم.
ابتدا روش ذخیره سازی و بازیابی متداول آن را بررسی میکنیم که تنها توسط دو تابع صورت می‌گیرد. مطلب زیر برگرفته از w3Schools است:
دسترسی به شیء webstorage به صورت زیر امکان پذیر است:

```
window.localStorage
window.sessionStorage
```

ولی بهتر است قبل از ذخیره و بازیابی، از پشتیبانی مرورگر از webstorage اطمینان حاصل نمایید:

```
if(typeof(Storage) !== "undefined") {
    // Code for localStorage/sessionStorage.
} else {
    // Sorry! No Web Storage support..
}
```

برای ذخیره سازی و سپس خواندن به شکل زیر عمل می‌کنیم:

```
localStorage.setItem("lastname", "Smith");

//========================
localStorage.getItem("lastname");
```

خواندن می‌تواند حتی به شکل زیر هم صورت بگیرد:

```
var a=localStorage.lastname;
```

استفاده از store.js برای مرورگرهایی که از webstorage پشتیبانی نمی‌کنند به شکل زیر است:

```
//ذخیره مقدار
store.set('username', 'marcus')

//بازیابی مقدار
store.get('username')

//حذف مقدار
store.remove('username')

//حذف تمامی مقادیر ذخیره شده
store.clear()

//ذخیره ساختار
store.set('user', { name: 'marcus', likes: 'javascript' })

//بازیابی ساختار به شکل قبلی
var user = store.get('user')
alert(user.name + ' likes ' + user.likes)

//تغییر مستقیم مقدار قبلی
store.getAll().user.name == 'marcus'

//بازخوانی تمام مقادیر ذخیر شده توسط یک حلقه
store.forEach(function(key, val) {
    console.log(key, '==', val)
})
```

همچنین بهتر هست از یک فلگ برای بررسی فعال بودن storage استفاده نمایید. به این دلیل که گاهی کاربرها از پنجره‌های

private استفاده می‌کنند که ردگیری اطلاعات آن ممکن نیست و موجب خطا می‌شود.

```
<script src="store.min.js"></script>
<script>
    init()
    function init() {
        if (!store.enabled) {
            alert('Local storage is not supported by your browser. Please disable "Private Mode", or
upgrade to a modern browser.')
            return
        }
        var user = store.get('user')
        // ... and so on ...
    }
</script>
```

در صورتیکه بخشی از داده‌ها را توسط localstorage ذخیره نمایید و بخواهید از طریق storage به آن دسترسی داشته باشید، خروجی string خواهد بود؛ صرف نظر از اینکه شما عدد، شیء یا آرایه‌ای را ذخیره کرده‌اید.
در صورتیکه ساختار JSON را ذخیره کرده باشید، می‌توانید رشته برگردانده شده را با json.stringify و json.parse بازیابی و به روز رسانی کنید.
در حالت cross browser تهیه‌ی یک sessionStorage امکان پذیر نیست. ولی می‌توان به روش ذیل و تعیین یک زمان انقضاء آن را محدود کرد:

```
var storeWithExpiration = {
// دریافت کلید و مقدار و زمان انقضا به میلی ثانیه
    set: function(key, val, exp) {
//ایجاد زمان فعلی جهت ثبت تاریخ ایجاد
        store.set(key, { val:val, exp:exp, time:new Date().getTime() })
    },
    get: function(key) {
        var info = store.get(key)
//در صورتی که کلید داده شده مقداری نداشته باشد نال را بر میگردانیم
        if (!info) { return null }
//تاریخ فعلی را منهای تاریخ ثبت شده کرده و در صورتی که
//از مقدار میلی ثاینه بیشتر باشد یعنی منقضی شده و نال بر میگرداند
        if (new Date().getTime() - info.time > info.exp) { return null }
        return info.val
    }
}

// استفاده عملی از کد بالا
// استفاده از تایمر جهت نمایش واکشی داده‌ها قبل از نقضا و بعد از انقضا
storeWithExpiration.set('foo', 'bar', 1000)
setTimeout(function() { console.log(storeWithExpiration.get('foo')) }, 500) // -> "bar"
setTimeout(function() { console.log(storeWithExpiration.get('foo')) }, 1500) // -> null
```

مورد بعدی استفاده از سورس cross-storage است. اگر به یاد داشته باشید گفتیم یکی از احتمالاتی که برای ما ایجاد مشکل می‌کند، ساب دومین هاست که ممکن است دسترسی ما به یک webstorage را از ساب دومین دیگر از ما بگیرد.
این کتابخانه به دو جز تقسیم شده است یکی هاب Hub و دیگری Client .

ابتدا نیاز است که هاب را آماده سازی و با ارائه یک الگو از آدرس وب، مجوز عملیات را دریافت کنیم. در صورتیکه این مرحله به فراموشی سپرده شود، انجام هر نوع عمل روی دیتاها در نظر گرفته نخواهد شد.

```
CrossStorageHub.init([
  {origin: /\.example.com$/,           allow: ['get']},
  {origin: /:\/\/(www\.)?example.com$/, allow: ['get', 'set', 'del']}
]);
```

حرف $ در انتهای عبارت باعث مشود که دامنه‌ها با دقت بیشتری در Regex بررسی شوند و دامنه زیر را معتبر اعلام کند:

```
valid.example.com
```

ولی دامنه زیر را نامعتبر اعلام می‌کند:

```
invalid.example.com.malicious.com
```

همچنین می‌توانید تنظیماتی را جهت هدرهای CSP و CORS، نیز اعمال نمایید:

```
{
  'Access-Control-Allow-Origin':  '*',
  'Access-Control-Allow-Methods': 'GET,PUT,POST,DELETE',
  'Access-Control-Allow-Headers': 'X-Requested-With',
  'Content-Security-Policy':      "default-src 'unsafe-inline' *",
  'X-Content-Security-Policy':    "default-src 'unsafe-inline' *",
  'X-WebKit-CSP':                 "default-src 'unsafe-inline' *",
}
```

پس کار را بدین صورت آغاز می‌کنیم، یک فایل به نام hub.htm درست کنید و هاب را آماده سازید: **hub.htm**

```
<script type="text/javascript" src="~/Scripts/cross-storage/hub.js"></script>
<script>
 CrossStorageHub.init([
      {origin: /.*localhost:300\d$/, allow: ['get', 'set', 'del']}
    ]);
  </script>
```

کد بالا فقط درخواست‌های هاست لوکال را از پورتی که ابتدای آن با 300 آغاز می‌شود، پاسخ می‌دهد و مابقی درخواست‌ها را رد می‌کند. متدهای ایجاد، ویرایش و حذف برای این آدرس معتبر اعلام شده است.
در فایل دیگر که کلاینت شناخته می‌شود باید فایل hub معرفی شود تا تنظیمات هاب خوانده شود:

```
var storage = new CrossStorageClient('http://localhost:3000/example/hub.html');

var setKeys = function () {

return storage.set('key1', 'foo').then(function() {
return storage.set('key2', 'bar');
      });
};
```

در خط اول، فایل هاب معرفی شده و تنظیمات روی این صفحه اعمال می‌شود. سپس در خطوط بعدی داده‌ها ذخیره می‌شوند. از آنجا که با هر یکبار ذخیره، return صورت می‌گیرد و تنها اجازه‌ی ورود یک داده را داریم، برای حل این مشکل متد then پیاده سازی شده است. متغیر setKeys شامل یک آرایه از کلیدها خواهد بود.
نحوه‌ی ذخیره سازی بدین شکل هم طبق مستندات صحیح است:

```
storage.onConnect().then(function() {

  return storage.set('key', {foo: 'bar'});

}).then(function() {

  return storage.set('expiringKey', 'foobar', 10000);

});
```

در کد بالا ابتدا یک داده دائم ذخیره شده است و در کد بعد یک داده موقت که بعد از 10 ثانیه اعتبار خود را از دست می‌دهد.
برای خواندن داده‌های ذخیره شده به نحوه زیر عمل می‌کنیم:

```
storage.onConnect().then(function() {
  return storage.get('key1');
}).then(function(res) {
  return storage.get('key1', 'key2', 'key3');
}).then(function(res) {
  // ...
});
```

کد بالا نحوه‌ی خواندن مقادیر را به شکل‌های مختلفی نشان میدهد و مقدار بازگشتی آن‌ها یک آرایه از مقادیر است؛ مگر اینکه تنها یک مقدار برگشت داده شود. مقدار بازگشتی در تابع بعدی به عنوان یک آرگومان در دسترس است. در صورتی که خطایی رخ دهد، قابلیت هندل آن نیز وجود دارد:

```
storage.onConnect()
    .then(function() {
      return storage.get('key1', 'key2');
    })

.then(function(res) {
      console.log(res); // ['foo', 'bar']
    })['catch'](function(err) {
      console.log(err);
    });
```

برای باقی مسائل چون به دست آوردن لیست کلیدهای ذخیره شده، حذف کلیدهای مشخص شده، پاکسازی کامل داده‌ها و ... به مستندات رجوع کنید.

در اینجا جهت سازگاری با مرورگرهای قدیمی خط زیر را به صفحه اضافه کنید:

```
<script src="https://s3.amazonaws.com/es6-promises/promise-1.0.0.min.js"></script>
```

ذخیره‌ی اطلاعات به شکل یونیکد، فضایی دو برابر کدهای اسکی میبرد و با توجه به محدود بودن حجم webstorage به 5 مگابایت ممکن است با کمبود فضا مواجه شوید. در صورتیکه قصد فشرده سازی اطلاعات را دارید می‌توانید از کتابخانه lz-string استفاده کنید. ولی توجه به این نکته ضروری است که در صورت نیاز، عمل فشرده سازی را انجام دهید و همینطوری از آن استفاده نکنید.

IndexedDB API

آخرین موردی که بررسی می‌شود استفاده از IndexedDB API است که با استفاده از آن میتوان با webstorage همانند یک دیتابیس رفتار کرد و به سمت آن کوئری ارسال کرد.

```
var request = indexedDB.open("library");

request.onupgradeneeded = function() {
  // The database did not previously exist, so create object stores and indexes.
  var db = request.result;
  var store = db.createObjectStore("books", {keyPath: "isbn"});
  var titleIndex = store.createIndex("by_title", "title", {unique: true});
  var authorIndex = store.createIndex("by_author", "author");

  // Populate with initial data.
  store.put({title: "Quarry Memories", author: "Fred", isbn: 123456});
  store.put({title: "Water Buffaloes", author: "Fred", isbn: 234567});
  store.put({title: "Bedrock Nights", author: "Barney", isbn: 345678});
};

request.onsuccess = function() {
  db = request.result;
};
```

کد بالا ابتدا به دیتابیس library متصل می‌شود و اگر وجود نداشته باشد، آن را می‌سازد. رویداد onupgradeneeded برای اولین بار اجرا شده و در آن می‌توانید به ایجاد جداول و اضافه کردن داده‌های اولیه بپردازید؛ یا اینکه از آن جهت به ارتقاء ورژن دیتابیس استفاده کنید. خصوصیت result، دیتابیس باز شده یا ایجاد شده را باز می‌گرداند. در خط بعدی جدولی با کلید کد ISBN کتاب تعریف شده است. در ادامه هم دو ستون اندیس شده برای عنوان کتاب و نویسنده معرفی شده است که عنوان کتاب را یکتا و بدون تکرار در نظر گرفته است. سپس در جدولی که متغیر store به آن اشاره می‌کند، با استفاده از متد put، رکوردها داخل آن درج می‌شوند. در صورتیکه کار با موفقیت انجام شود رویداد onSuccess فراخوانی می‌گردد.

برای انجام عملیات خواندن و نوشتن باید از تراکنش‌ها استفاده کرد:

```
var tx = db.transaction("books", "readwrite");
var store = tx.objectStore("books");

store.put({title: "Quarry Memories", author: "Fred", isbn: 123456});
store.put({title: "Water Buffaloes", author: "Fred", isbn: 234567});
```

```
store.put({title: "Bedrock Nights", author: "Barney", isbn: 345678});

tx.oncomplete = function() {
  // All requests have succeeded and the transaction has committed.
};
```

در خط اول ابتدا یک خط تراکنشی بین ما و جدول books با مجوز خواندن و نوشتن باز می‌شود و در خط بعدی جدول books را در اختیار می‌گیریم و همانند کد قبلی به درج داده‌ها می‌پردازیم. در صورتیکه عملیات با موفقیت به اتمام برسد، متغیر تراکنش رویدادی به نام oncomplete فراخوانی می‌گردد. در صورتیکه قصد دارید تنها مجوز خواندن داشته باشید، عبارت readonly را به کار ببرید.

```
var tx = db.transaction("books", "readonly");
var store = tx.objectStore("books");
var index = store.index("by_author");

var request = index.openCursor(IDBKeyRange.only("Fred"));
request.onsuccess = function() {
  var cursor = request.result;
  if (cursor) {
    // Called for each matching record.
    report(cursor.value.isbn, cursor.value.title, cursor.value.author);
    cursor.continue();
  } else {
    // No more matching records.
    report(null);
  }
};
```

در دو خط اول مثل قبل، تراکنش را دریافت می‌کنیم و از آنجا که می‌خواهیم داده را بر اساس نام نویسنده واکشی کنیم، ستون اندیس شده نام نویسنده را دریافت کرده و با استفاده از متد opencursor درخواست خود را مبنی بر واکشی رکوردهایی که نام نویسنده **fred** است، ارسال میداریم. در صورتیکه عملیات با موفقیت انجام گردد و خطایی دریافت نکنیم رویداد onsuccess فراخوانی می‌گردد. در این رویداد با دو حالت برخورد خواهیم داشت؛ یا داده‌ها یافت می‌شوند و رکوردها برگشت داده می‌شوند یا هیچ رکوردی یافت نشده و مقدار نال برگشت خواهد خورد. با استفاده از cursor.continue می‌توان داده‌ها را به ترتیب واکشی کرده و مقادیر رکورد را با استفاده خصوصیت value به سمت تابع report ارسال کرد.

کدهای بالا همه در مستندات معرفی شده وجود دارند و ما پیشتر توضیح ابتدایی در مورد آن دادیم و برای کسب اطلاعات بیشتر می‌توانید به همان مستندات معرفی شده رجوع کنید. برای idexedDB هم می‌توانید از این منابع + + + استفاده کنید که خود w3c منبع فوق العاده‌تری است.