Introduction to Cookies

Cookies are a popular and cross-platform mechanism by which website authors can store user-specific content for later use. For example, a user’s display settings for a stock tracker might be stored in a cookie –due to the way HttpRequest objects are brought into an ASP.NET (or Padarn) web application, the cookie information can be read easily and parsed like a dictionary data structure. Cookies can be optionally persisted to the user’s hard drive; generally cookies are stored in the user’s browser-defined location. For Internet Explorer users, this is generally in the Temporary Internet Files folder. For Firefox users, this is stored in the cache folder under the profile main folder.

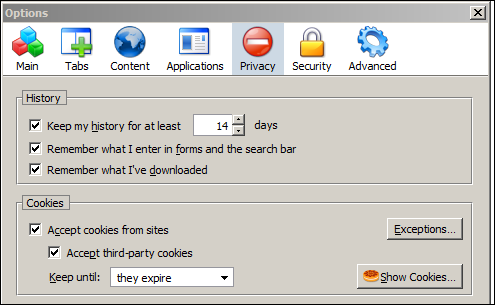
The cookie itself is a small text file that gets attached to any HttpRequest posted to a web server, so long as the cookie’s domain has been established appropriate for the web server. If your website is configured as [www.hostname.com](http://www.hostname.com), the domain of the cookie would have to be established as [www.hostname.com](http://www.hostname.com). Cookies also support application path specificity. So if you have a cookie that should only be used for applications stored in [www.hostname.com/application](http://www.hostname.com/application), your cookie will need to have a Path property value set.

In addition to user-specific preferences, many web developers use cookies to determine if an authentication session remains valid. Instead of forcing the user to login every time a request to a particular page occurs, a cookie can be used to store the last visit date and time. If the last visit happened recently, and the cookie contains a valid authentication token (Guid, for example), then the user won’t be asked to login. The demonstration code in this HOL illustrates how this can be used in a Padarn web application.

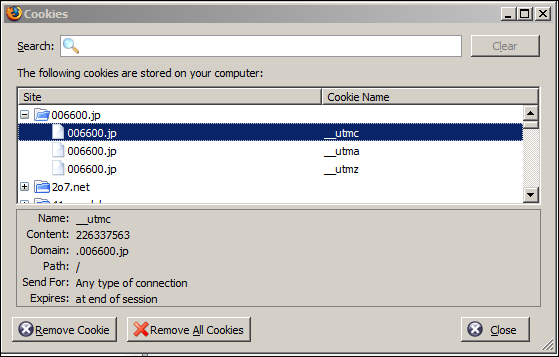
Another well-advertised feature of cookies is they can be disabled in situations where only SSL connections should be supported. Unfortunately, this feature doesn’t help all that much in minimizing security breaches since information is stored in an unencrypted (by default) format so if sensitive data is exchanged over a secured connection and that data is persisted to disk, if the user’s machine is hacked, that information can be read from the cookie.

As powerful as cookies can be in developing compelling web applications, they do have some other weaknesses. Cookies only support up to 4096 bytes, so it’s best to think carefully about streamlining all cookie needs for an entire site into one file as opposed to having different cookies for different user particulars. Often, cookie information is keyed off of a particular user ID. Cookie support is dictated by the web browser used to access the site. Some browsers only support 20 or so cookies per website. Some browsers have a hard limit on total cookies across all sites. That number is usually around 300. Some users opt to disable cookies completely and so cookie exchanges will fail. Our code ensures browser cookie support is enabled. Of course, the user can always delete the cookie associated with your website, so it’s critical to check for its existence before attempting to read from it.

In Firefox, the Privacy tab houses the settings for cookies across your profile:



Within this tab, it’s possible to view individual cookie contents.



In the “Show Cookies” dialog, you can inspect the contents of individual cookies for a site. For the website [www.006600.jp](http://www.006600.jp), you can see there are three cookies created. The first one doesn’t ever get persisted to disk and will be deleted after the web response is returned to the browser. Cookies are never deleted explicitly by a website using a cookie; instead, websites can alter the expiration date of a cookie to alter the persistence behavior. A cookie can be set to expire in the past so that the cookie will be deleted once the HttpResponse is sent to the browser. Otherwise, a cookie can be set far into the future so that only when the user manually deletes the cookie or invokes other state management behavior (like logging out of a website) will the cookie be deleted.

* Writing Cookies –

The web browser handles all of the plumbing required to persist cookies to a user’s profile folder. When building a Padarn website, we use the HttpResponse object to access the Cookies collection (implemented as a NameObjectCollectionBase) to modify the contents of a cookie file. Remember: the cookie that we have access to is applicable only to the domain and path specified by the cookie. Since the Cookies collection is treated as a name/value pair, it’s important to use unique names for anything stored in the cookie, as this will be set as the key. If uniqueness is not enforced, then the last name/value pair entered will overwrite any prior entries.

If you fail to specify the expiration date (using the DateTime Expires property), then the content stored will be treated as session information and the cookie will not be persisted to the user’s hard disk. When the browser window is closed (or tab), the content is erased permanently. Often, websites ask the user if they are using a publicly accessible workstation to determine if cookies should be stored.