8.6 Using Cookie Attributes

Before adding the cookie to the outgoing headers, you can set various characteristics of the cookie by using the following `set`*Xxx* methods, where *Xxx* is the name of the attribute you want to specify.

Although each `set`*Xxx* method has a corresponding `get`*Xxx* method to retrieve the attribute value, note that the attributes are part of the header sent from the server to the browser; they are *not* part of the header returned by the browser to the server. Thus, except for name and value, the cookie attributes apply only to *outgoing* cookies from the server to the client; they aren't set on cookies that come *from* the browser to the server. So, don't expect these attributes to be available in the cookies you get by means of `request.getCookies`. This means that you can't implement continually changing cookie values merely by setting a maximum age on a cookie once, sending it out, finding the appropriate cookie in the incoming array on the next request, reading the value, modifying it, and storing it back in the `Cookie`. You have to call `setMaxAge` again each time (and, of course, pass the `Cookie` to `response.addCookie`).

Here are the methods that set the cookie attributes.

public void setComment(String comment)

public String getComment()

These methods specify or look up a comment associated with the cookie. With version 0 cookies (see the upcoming entry on `setVersion` and `getVersion`), the comment is used purely for informational purposes on the server; it is not sent to the client.

public void setDomain(String domainPattern)

public String getDomain()

These methods set or retrieve the domain to which the cookie applies. Normally, the browser returns cookies only to the exact same hostname that sent the cookies. For instance, cookies sent from a servlet at bali.vacations.com would not normally get returned by the browser to pages at queensland.vacations.com. If the site wanted this to happen, the servlets could specify `cookie.setDomain(".vacations.com")`. To prevent servers from setting cookies that apply to hosts outside their domain, the specified domain must meet the following requirements: it must start with a dot (e.g., `.coreservlets.com`); it must contain two dots for noncountry domains like `.com`, `.edu`, and `.gov`; and it must contain three dots for country domains like `.co.uk` and `.edu.es`.

public void setMaxAge(int lifetime)

public int getMaxAge()

These methods tell how much time (in seconds) should elapse before the cookie expires. A negative value, which is the default, indicates that the cookie will last only for the current browsing session (i.e., until the user quits the browser) and will not be stored on disk. See the `LongLivedCookie` class (Listing 8.4), which defines a subclass of `Cookie` with a maximum age automatically set one year in the future. Specifying a value of 0 instructs the browser to delete the cookie.

public String getName()

The `getName` method retrieves the name of the cookie. The name and the value are the tw pieces you virtually *always* care about. However, since the name is supplied to the `Cookie` constructor, there is *no* `setName` method; you cannot change the name once the cookie is created. On the other hand, `getName` is used on almost every cookie received by the serve Since the `getCookies` method of `HttpServletRequest` returns an array of `Cookie` objects, common practice is to loop down the array, calling `getName` until you have a particular nan then to check the value with `getValue`. For an encapsulation of this process, see the `getCookieValue` method shown in Listing 8.3.

public void setPath(String path)

public String getPath()

These methods set or get the path to which the cookie applies. If you don't specify a path, browser returns the cookie only to URLs in or below the directory containing the page that the cookie. For example, if the server sent the cookie from `http://ecommerce.site.com/toys/specials.html`, the browser would send the cookie b when connecting to `http://ecommerce.site.com/toys/bikes/beginners.html`, but not t `http://ecommerce.site.com/cds/classical.html`. The `setPath` method can specify something more general. For example, `cookie.setPath("/")` specifies that *all* pages on tl server should receive the cookie. The path specified must include the current page; that is may specify a more general path than the default, but not a more specific one. So, for exa a servlet at `http://host/store/cust-service/request` could specify a path of `/store/` (since `/store/` includes `/store/cust-service/`) but not a path of `/store/cust-service/returns/` (since this directory does not include `/store/cust-service/`).

Core Approach

To specify that a cookie apply to all *URLs on your site, use `cookie.setPath ("/")`.*

public void setSecure(boolean secureFlag)

public boolean getSecure()

This pair of methods sets or gets the boolean value indicating whether the cookie should o be sent over encrypted (i.e., SSL) connections. The default is `false`; the cookie should ap all connections.

public void setValue(String cookieValue)

public String getValue()

The `setValue` method specifies the value associated with the cookie; `getValue` looks it up Again, the name and the value are the two parts of a cookie that you *almost always* care a although in a few cases, a name is used as a boolean flag and its value is ignored (i.e., the existence of a cookie with the designated name is all that matters). However, since the co value is supplied to the `Cookie` constructor, `setValue` is typically reserved for cases when change the values of incoming cookies and then send them back out. For an example, see Section 8.10 (Modifying Cookie Values: Tracking User Access Counts).

public void setVersion(int version)

public int getVersion()

These methods set and get the cookie protocol version with which the cookie complies. Version 0, the default, follows the original Netscape specification (http://wp.netscape.com/newsref/std/cookie_spec.html). Version 1, not yet widely supported, adheres to RFC 2109 (retrieve RFCs from the archive sites listed at http://www.rfc-editor.org/).

[Team LiB]

PREVIOUS NEXT