

8.8 Basic Cookie Utilities

This section presents some simple but useful utilities for dealing with cookies.

Finding Cookies with Specified Names

[Listing 8.3](#) shows two static methods in the `CookieUtilities` class that simplify the retrieval of a cookie or cookie value, given a cookie name. The `getCookieValue` method loops through the array of available `Cookie` objects, returning the value of any `Cookie` whose name matches the input. If there is no match, the designated default value is returned. So, for example, our typical approach for dealing with cookies is as follows:

```
String color =
    CookieUtilities.getCookieValue(request, "color", "black");

String font =
    CookieUtilities.getCookieValue(request, "font", "Arial");
```

The `getCookie` method also loops through the array comparing names but returns the actual `Cookie` object instead of just the value. That method is for cases when you want to do something with the `Cookie` other than just read its value. The `getCookieValue` method is more commonly used, but, for example, you might use `getCookie` in lieu of `getCookieValue` if you wanted to put a new value into the cookie and send it back out again. Just don't forget that if you use this approach, you have to respecify any cookie attributes such as date, path, and domain: these attributes are not set for incoming cookies.

Listing 8.3 CookieUtilities.java

```
package coreservlets;

import javax.servlet.*;
import javax.servlet.http.*;

/** Two static methods for use in cookie handling. */

public class CookieUtilities {

    /** Given the request object, a name, and a default value,
     *  this method tries to find the value of the cookie with
     *  the given name. If no cookie matches the name,
     *  the default value is returned.
     */

    public static String getCookieValue
        (HttpServletRequest request,
         String cookieName,
         String defaultValue) {
        Cookie[] cookies = request.getCookies();
        if (cookies != null) {
            for(int i=0; i<cookies.length; i++) {
                Cookie cookie = cookies[i];
                if (cookieName.equals(cookie.getName())) {
                    return(cookie.getValue());
                }
            }
        }
        return(defaultValue);
    }
}
```

```

}

/** Given the request object and a name, this method tries
 * to find and return the cookie that has the given name.
 * If no cookie matches the name, null is returned.
 */

public static Cookie getCookie(HttpServletRequest request,
                               String cookieName) {
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for(int i=0; i<cookies.length; i++) {
            Cookie cookie = cookies[i];
            if (cookieName.equals(cookie.getName())) {
                return(cookie);
            }
        }
    }
    return(null);
}
}

```

Creating Long-Lived Cookies

[Listing 8.4](#) shows a small class that you can use instead of `Cookie` if you want your cookie to automatically persist for a year when the client quits the browser. This class (`LongLivedCookie`) merely extends `Cookie` and calls `setMaxAge` automatically.

Listing 8.4 LongLivedCookie.java

```

package coreservlets;

import javax.servlet.http.*;

/** Cookie that persists 1 year. Default Cookie doesn't
 * persist past current browsing session.
 */

public class LongLivedCookie extends Cookie {
    public static final int SECONDS_PER_YEAR = 60*60*24*365;

    public LongLivedCookie(String name, String value) {
        super(name, value);
        setMaxAge(SECONDS_PER_YEAR);
    }
}

```

[\[Team LiB \]](#)

[◀ PREVIOUS](#) [NEXT ▶](#)