9.4 Browser Sessions vs. Server Sessions

By default, session-tracking is based on cookies that are stored in the browser's memory, not written to disk. Thus, unless the servlet explicitly reads the incoming `JSESSIONID` cookie, sets the maximum age and path, and sends it back out, quitting the browser results in the session being broken: the client will not be able to access the session again. The problem, however, is that the server does not know that the browser was closed and thus the server has to maintain the session in memory until the inactive interval has been exceeded.

Consider a physical shopping trip to a Wal-Mart store. You browse around and put some items in a physical shopping cart, then leave that shopping cart at the end of an aisle while you look for another item. A clerk walks up and sees the shopping cart. Can he reshelve the items in it? No—you are probably still shopping and will come back for the cart soon. What if you realize that you have lost your wallet, so you get in your car and drive home? Can the clerk reshelve the items in your shopping cart now? Again, no—the clerk presumably does not *know* that you have left the store. So, what can the clerk do? He can keep an eye on the cart, and if nobody has touched it for some period of time, he can then conclude that it is abandoned and take the items out of it. The only exception is if you brought the cart to him and said "I'm sorry, I left my wallet at home, so I have to leave."

The analogous situation in the servlet world is one in which the server is trying to decide if it can throw away your `HttpSession` object. Just because you are not currently using the session does not mean the server can throw it away. Maybe you will be back (submit a new request) soon? If you quit your browser, thus causing the browser-session-level cookies to be lost, the session is effectively broken. But, as with the case of getting in your car and leaving Wal-Mart, the server does not *know* that you quit your browser. So, the server still has to wait for a period of time to see if the session has been abandoned. Sessions automatically become inactive when the amount of time between client accesses exceeds the interval specified by `getMaxInactiveInterval`. When this happens, objects stored in the `HttpSession` object are removed (unbound). Then, if those objects implement the `HttpSessionBindingListener` interface, they are automatically notified. The one exception to the "the server waits until sessions time out" rule is if `invalidate` or `logout` is called. This is akin to your explicitly telling the Wal-Mart clerk that you are leaving, so the server can immediately remove all the items from the session and destroy the session object.