Sessions and Cookies

The way sessions work in the software has changed over the years in the software. To better understand why it works like it does now, it helps to look at how things worked back In the Old Days.

How Sessions Work

When you view any of the normal pages, if you don't have a classified_session cookie set, it will try to set one, and will create a new session for you. This session is created for every page load if there is not already a session.

This is a list of what sessions are used for in the software[1)]:

- Keep track of data submitted during the registration progress.
- In Anonymous Listings Addon, sessions are used to keep track of listing data set while placing an anonymous listing.
- If the user logs in, the session keeps track of the logged in user's data, to keep them logged in, and keep track of progress made when placing a listing or other similar activities.
- "real time" general statistics information displayed, such as the number of people currently viewing the site.
- List of users currently logged in (displayed in admin panel)
- Keep track of the "state" of certain user interface elements, for instance to show the last tab selected or show the last browsing view selected.
- If using the social connect addon, and user logs in with Facebook, used to keep track of Facebook ID and related information.

This is how a new registration currently works:

1. User views the registration form page. Like all pages in the software, it checks to see if a session is already created. If not, it creates a new session, and sets a cookie, but does not verify that the cookie is saved correctly by re-loading the page, so there is no way for it to know if cookies work on the first page of registration.
2. User clicks "submit" button, and it submits to an "in between" step that says "Verifying Registration Data" or something similar. However it is not actually verifying the registration data, this step's sole purpose is to verify, without a doubt, whether or not cookies work or not. We'll get into why and how it works in a little bit. On this page it also checks to see if a session is already created, if not it creates one and attempts to set the cookie. This page has a "hidden form" that contains all the information just submitted. After a few seconds, that form is then re-submitted (using JavaScript) to proceed to the next step. For users with Javascript disabled, the hidden form can also be submitted by clicking on the loading animated image.
3. This is the page where the registration data is actually processed, where any errors are displayed, etc. On this page one of the first things it checks is to see if a session is currently set. If there is no session or no cookie set, because of the "in between" step before, the software knows there is definitely a problem with cookies, so it displays an error to the user informing them.

When the user logs in, the process is very similar to above, except that the data being submitted is the login data, not the registration data.

Reason For In-Between Step

In the scenario given in the section above this one, imagine what would happen without the "in between" step. On the page where it processes the form data, and it also verifies that cookies are working. Sure it will work just fine if the user has a classified_session cookie and the session has not expired. But what would happen if the user waited to submit the registration page, so that the session has expired? Without the in-between step, the software would give an error saying that cookies are not working. But they are working just fine, it's just that the cookie or the session has expired!

This is where the in-between step comes into play. Say the user waits a while to submit the registration form, and the session has expired already. On the in-between step, it creates a brand new session because the old one has expired. On the next page where it verifies that a session exists to make sure cookies are working, the only time a session would not exist, is if cookies are not working. Now when it displays the error message about cookies not working, that error message is accurate and not the result of the user just waiting a long time to submit the form.

In the Old Days

Back in the old days, sessions worked OK as long as you had cookies, but if you didn't have cookies, all bets were off for whether you got to see the site or not. Below is a brief history of how session handling has progressed through the different versions, written partially from memory and partially from release notes, so the accuracy is not guaranteed. It is meant as more to give you an idea of where the software has come from and get a glimpse as to why it works like it does now.

Pre 2.0.6

This period of session handling is what I like to call the dark ages of session handling.

Back then, cookies either worked, or you didn't get to see the website. If you did not have cookies turned on, the page would continuously re-load until the browser figured it out and gave up. Or it might just be a "white screen of death" in some cases. This is obviously very bad, specifically it made the software virtually invisible to search engines. Well on Enterprise editions of ClassAuctions software this is how it was. It is a little hazy on how it worked in Premier, Basic, or Lite editions pre-2.0.10 because the code was that hard to follow[2)]. I believe though that it was opposite of this, it did not really *care* if cookies worked or not so the page would display just fine and search engines could see the pages. But then if cookies did not work, when you try to log in or register or anything like that it would just keep refreshing the page like nothing happened[3)]. That is obviously not good behavior either, but a little better than the not working at all and the software not displaying any pages.

Typical Scenario for GeoClassAuctions Enterprise

1. User[4)] views the page. The software sees if the "classified_session" cookie is set. If the cookie is not set, go to #2. If the cookie is set already, go to step 3.
2. Cookie is not set yet, so the software attempts to set the cookie. The only way to see if the cookie is actually saved or not however, is to re-load the page, so the page is re-loaded. Go to

step #1.
3. Cookie is found and working, so display the page.

Now imagine what would happen if cookies were turned off in the above scenario. You would keep looping between steps 1 and 2 over and over, and this is why it made things "invisible" to search engines or anyone without cookies.

This is around the time a new developer joined the ranks at Geodesic Solutions LLC. and saw all the problems with how sessions worked (or didn't work), and decided to try to make them better.[5)]

2.0.6 - 3.0.2

This marked an "in between" stage in session handling, where it was getting better, but still had its flaws.

During this period, in order to verify cookies were working, the page would re-load 3 times and after that would just accept that cookies did not work, and would display the page. There was still no explanation to the end user however when they tried to log in. Here is the new scenario:

Typical Session Scenario for this period

1. User views the page. The software sees if the "classified_session" cookie is set. If it is not set, go to #2. If it is set, or the redirect_count is 3 or more, go to step 3.
2. Cookie is not set yet, so the software attempts to set the cookie. The page is re-loaded, but with an extra "parameter" added saying the "redirect count". If the redirect count is not already set, it would be set to 1, if it is set it would be incremented. Go to step #1.
3. Cookie is set, or the "redirect count" is 3 or more, so display the page.

So now, if the cookie is not set, it would redirect 3 times, then display the page. There is a technical reason for trying 3 times but I won't bore you with the details. This seemed to work OK but still had its flaws. For instance, every time the page re-directed, it created a new session, so that's 3 new sessions every time someone views the page with cookies turned off.

There is still not an accurate message displayed when the user tries to log in or register when cookies are turned off, it just re-loads the page with no explanation of why things don't work. There was also issues with some of the newer session security added, like it would force the session to have the same IP number to help make *session hijacking* harder. That turned out to be a bad idea for certain ISP's, specifically AOL and a few others where the IP would change.

So the developer decided to hit the drawing board again to come up with an even better solution, that would show **accurate** error messages if cookies were not working, would not require re-directing 3 times just to view the page without cookies, etc. The result is the current way sessions are handled, which you can see in the section How Sessions Work.

1)
This list will not include any functionality added by 3rd party addons.
2)
Or more accurately, the developer writing this never looked very closely at how it worked in those packages, but did get a feel for how it seemed to work.
3)
This is from memory so it may have worked differently in those much older versions
4)

By user, we mean either a real user, or a search engine spider
[5)]

This is a huge over-simplification of the version numbers and when that new developer actually came in by the way, as back then there were actually different software for the different editions, and quite a bit of over-lapped version numbers causing all sorts of confusion. The developer came in around 2.0.0b but that is actually a higher version than 2.0.5.2, actually 1.0.5b is newer than version 2.0.5.2. See why we skipped over all that to version 3.0 once we moved all the different software to use the same base code?