Subject: Re: A few questions about Skylark sessions
Posted by dolik.rce on Fri, 28 Dec 2012 08:33:48 GMT
View Forum Message <> Reply to Message

Hi Peter

Peter wrote on Thu, 27 December 2012 20:421. When does a new session actually start? Suppose I have a basic Skylark application with just one handler which does nothing (empty definition) and go to the corresponding URL. Will this start a new session if no session is active? Or would I need an explicit call to SessionSet() or NewSessionId()?
The session cookie is first sent to client (and the session record on the server is first created) when it is first needed. That is when you first use SessionSet or NewSessionId or when you use $post_identity() in witz or similar function that calls SessionSet internally. Empty handler doesn't require any session information, so no session is created.

Peter wrote on Thu, 27 December 2012 20:422. What's the exact difference between ClearSession() and NewSessionId()? The former is supposed to clear current session id and variables, but according to manual it retains values in shared variable space. Does it mean that if I have a session variable .VAR set to 1 and call ClearSession() then http[".VAR"] (cast to int) will still be 1?

Basically: what does ClearSession() exactly do (what does it clear and what does it preserve)? What's the difference between:

- calling ClearSession() alone?
- calling NewSessionId() alone?
- calling ClearSession() and then NewSessionId()?
NewSessionId() keeps all the information associated with the session, only changes the key under which it is stored. So it clears nothing, only changes the id. The most common reason for this is to change the id (and cookie value on client side) after log in and log out to prevent session hijacking.

ClearSession() deletes both the session variables and the session id, only thing that is kept is values of the session variables in shared variable space. So you are right about the behavior: if you have session variable .VAR and call ClearSession(), you can still use its value from http[".VAR"], but only for the same request when you called ClearSession. Any subsequent requests will not have the session variables stored, so they won't appear in the shared variable space anymore.

Calling both, ClearSession() and then NewSessionId() is probably not useful in any way. As soon as you ClearSession(), next call to SessionSet() would create new session id anyways.

Peter wrote on Thu, 27 December 2012 20:423. What's the real meaning of SkylarkSessionConfig::expire? When I set it to 600 (10 minutes), it should clear the current session after 10 minutes since... when? The moment a session file is saved on server? What's supposed to happen after 10 minutes? A new session is started automatically or not? Is it supposed to work the same in all browsers? In another thread someone said it shouldn't be set to less than 10 minutes. Why is that?

The true meaning of SkylarkSessionConfig::expire is "The minimal time between requests (in seconds), that the session information is guaranteed to persist". The expiration time is always counted from last request that modified the session. Because expiring the sessions is costly operation, it is highly optimized so it happens in batches from time to time. The exact mechanism (in pseudo code) is:

```
Every 1000 requests do {
    if (now() - last_expiration_check > 10 min) {
        last_expiration_check = now();
        expire_time = now() - SkylarkSessionConfig::expire
        for each session {
            if (session.last_write_time < expire_time){
                delete(session);
            }
        }
    }
}
```

So as you can see the expiration is performed after every 1000th request is served (counted per thread), but only if it is longer then 10minutes after last expiration. That is why I said it is useless to set the expire value to less than 10 minutes.

After the session is expired, the server will not recognize it, and will behave as if it is a new, unknown user coming. If you try to set any session variables for him, new session will be created.

It should work the same way across browsers, only difference might be in how the browsers are set up to handle cookies. The session cookie is sent as a session cookie, meaning that it should be deleted when the browser is closed. It is possible that some browsers misbehave a bit in this behavior...

I hope I explained it better this time If there is still something unclear, don't hesitate to ask more. You can also try to reach me on irc/gtalk/icq to ask me in real time.

Best regards,
Honza