Sessions

This chapter documents sessions and server -side variables.

- **Overview** describes how sessions operate and how sessions can be used
- **Session Methods** describes the methods which can be used to create, manipulate, and delete sessions
- **Session Tasks** shows various tasks one might perform when using sessions
- **Session Example** this self contained example illustrates how to use sessions to store site preferences

Overview

Sessions allow variables to be created which are persistent from page to page within a web site. Rather than passing data using HTML forms or URLs, visitor-specific data can be stored in Lasso variables which are automatically saved and retrieved by Lasso for each page a visitor loads.

Sessions can be utilized for a variety of purposes, including:

- **Current state** - Sessions can store the current state of a Web site for a given visitor. They can determine what the last search they performed was, how the data on a results page was sorted, or in what format the data should be presented.
- **Store references to database data** - Key column values can be stored in a session for quick access. These might include records in a user database or a shopping cart database.
- **Store authentication information** - After a visitor has authenticated themselves using a username and password, that authentication information can be stored in a session and then checked to ensure that the same visitor is accessing data from page to page.
- **Store data without using a database** - Complex data types such as arrays and maps can be stored in session variables. In a web site with multiple forms the data from each form can be stored in a session and only placed in the database once the final form is submitted. Or, a shopping cart can be stored in a session and only placed in an orders database on checkout.

How Sessions Work

Table of Contents

Language Guide

A session has three characteristics: a name, a list of variables that should be stored, and an ID string that identifies a particular site visitor.

- **Name** - The session name is defined when the session is created by the `session_start` method. The same session name must be used for each request that which wants to load the session. The name usually represents what type of data is being stored in the session, e.g. `Shopping_Cart` or `Site_Preferences`.
- **Variables** - Each session maintains a list of variables which are being stored. Variables can be added to the session using `session_addVar`. The values for all variables in the session are remembered at the end of each request which loads the session. The value for each saved variable is restored when that session is next loaded. Only *thread variables* can be added to a session.
- **ID** - Lasso automatically creates an ID string for each site visitor when a session is created. The ID string is either stored in a cookie or passed from page to page using the `-lassosession` GET/POST parameter. When a session is loaded, the ID of the current visitor is combined with the name of the session to locate and load the particular set of variables for that session and the current visitor.

Sessions are created and loaded using the `session_start` method. This method should be used early for each request that needs access to the session variables. The `session_start` method either creates a new session or loads an existing session depending on whether there are existing variables currently stored for the site visitor.

Sessions can be set to expire after a specified amount of idle time. The default is 15 minutes. If the visitor has not loaded a page which starts the session within the idle time limit, then the session will be deleted automatically. Note that the idle timeout restarts every time a request loads the session.

Once a variable has been added to a session using the `session_addVar` method, its stored value will be set each time the `session_start` method is called. The variable does not need to be added to the session on each request, though it is safe to do so. A variable can be removed from a session using the `session_removeVar` method. This method does not alter a variable's current value, but does prevents the value of the variable from being saved for the session.

Session Methods

Each of the session methods is described in the table below.

Method	Description
`session_start`	Starts a new session or loads an existing session. Parameters are described in the next table.
`session_id(sessionName)`	Returns the current session ID. Accepts a single parameter: the name of the session for which the session ID should be returned.

`session_addVar(`
`  sessionName,`
`  varName`
`)`
	Adds a variable to a specified session. Accepts two parameters: the name of the session and the name of the variable.

`session_removeVar(`
`  sessionName,`
`  varName`
`)`
	Removes a variable from a specified session. Accepts two parameters: the name of the session and the name of the variable.

`session_end(`
`  sessionName,`
`  -secure=false`
`)`
	Deletes the stored information about a named session for the current visitor. Accepts a required parameter: the name of the session to be deleted, and an optional keyword parameter: `-secure`. The `-secure` keyword should be true if the `-secure` keyword was true when session_start was called.

`session_abort(sessionName)`	Prevents the session from being stored at the end of the current request. This allows graceful recovery from an error that would otherwise corrupt data stored in the session. Accepts a single parameter: the name of the session to be aborted.
`session_result(sessionName)`	When called immediately after the session_start method, session_result returns "new", "load", or "expire" depending on whether a new session was created, an existing session loaded, or an expired session forced a new session to be created, respectively. If session_start is called with the optional `-rotate` keyword parameter, then the word "rotate" may also be returned from this method.
`session_deleteExpired`	This method is used internally by the session manager and does not normally need to be called directly. It trigers a cleanup routine which deletes expired sessions from the current session storage location.

Session_Start Parameters

Parameter	Type	Default	Description
name	string	*none*	The name of the session. This is the only required parameter. All other parameters are optional and have default values which cover the majority of usages.
-expires	integer	15	The idle expiration time for the session in minutes
-id	string	*none*	Optionally sets the ID for the current visitor. This permits the ID to be supplied explicitly by the developer. If no ID is specified then Lasso will automatically create an ID.
-useCookie	boolean	true	If true, then sessions will be tracked by cookie. -useCookie defaults to true unless -useLink , -useAuto , or -useNone are specified.
-useLink	boolean	false	If true, then sessions will be tracked by modifying all the absolute and relative links in the outgoing response data.
-useNone	boolean	false	If specified, no links on the current page will be modified and a cookie will not be set. -useNone allows custom session tracking to be used, bypassing the automated methods provided by Lasso.
-useAuto	boolean	true	This option automatically uses -useCookie if cookies are available on the visitor's browser or -useLink if they are not. Since Lasso has no way of knowing if cookies are enabled when a session is first started, -useLink is implicitly true on that first request and links will be adjusted to carry the session. If the session cookie is present on subsequent requests, -useLink will be implicitly false and links will not be adjusted.
-cookieExpires	integer	*none*	Optionally sets the expiration in minutes for the session cookie. This permits the cookie expiration to be set, regardless of the overall expiration for the session itself.
-domain	string	*none*	Optionally sets the domain for the session cookie.
-path	string	*none*	Optionally sets the path for the session cookie.
-secure	boolean	false	If true, the session cookie will only be sent back to the web server on requests for HTTPS secure web pages. session_end should also be specified with -secure if this option is desired.
-rotate	boolean	false	If true, the session will have a new ID generated for it on each request.

Note: *`-useCookie` is the default for `session_start` unless `-useLink` or `-useNone` are specified. Use `-useLink` to track a session using only links. Use both `-useLink` and `-useCookie` to track a session using both links and a cookie.*

Starting a Session

The `session_start` method is used to start a new session or to load an existing session. When the `session_start` method is called with a given `name` parameter it first checks to see whether an ID is defined for the current visitor. The ID is searched for in the following three locations:

- **Parameter** - If the `session_start` method has an `-id` keyword parameter then it is used as the ID for the current visitor.
- **Cookie** - If a session tracker cookie is found for the name of the session then the ID stored in the cookie is used.
- **-lassosession** - If a `-lassosession` parameter for the name of the session was specified as a GET or POST parameter then that value is used as the session ID.

The name of the session and the ID are used to check whether a session has already been created for the current visitor. If it has then the variables in the session are loaded replacing the values for any variables of the same name that are already active on the current page.

If no ID can be found, the specified ID is invalid, or if the session identified by the name and ID has expired then a new session is created.

After the `session_start` method has been called, the `session_id` method can be used to retrieve the ID of the current session. It is guaranteed that either a valid session will be loaded or a new session will be created when `session_start` is called.

Note: *The `session_start` method must be used once for each request that will access session variables.*

Session Tracking

The session ID for the current visitor can be tracked using two different methods or a custom tracking system can be devised. The tracking system to be used depends on which parameters are specified when the `session_start` method is called.

- **Cookie** - The default session tracking method is to use a browser cookie. If no other method is specified when creating a session then the `-useCookie` method is used by default. The cookie will be inspected automatically when the visitor makes another request which includes a call to the `session_start` method. No additional programming is required.
- The session tracking cookie is of the following form. The name of the cookie includes the words `_Session_Tracker_` followed by the name given to the session in `session_start`. The value for the cookie is the session ID as returned by `session_id`.

```
_LassoSessionTracker_SessionName=1234567890abcdefg
```

- **Links** - If the `-useLink` parameter is specified in the `session_start` method then Lasso will automatically modify links contained on the current page. No additional programming beyond specifying the `-useLink` parameter is required.
- By default, links contained in the `href` parameter of `... (http://lassosoft.wiki.zoho.com/...)` and in the `action` parameter of `tags will be modified.`
- Links are only modified if they reference a file on the same machine as the current Web site. Any links which start with any of the following strings are not modified.
  file:// ftp:// http:// https:// javascript: mailto: telnet:// #
- Links are modified by adding a `-lassosession` parameter to the end of the link. The value of the parameter is the session ID, as returned by the `session_id` method, followed by an underscore and then the session name. For example, an anchor tag referencing the current file would appear as follows after the `-lassosession` parameter was added.
- **Auto** - If the `-useAuto` parameter is specified in the `session_start` method then Lasso will check for a cookie with an appropriate name for the current session. If the cookie is found then `-useCookie` will be used to propagate the session. If the cookie cannot be found then `-useLink` will be used to propagate the session. This allows a site to preferentially use cookies to propagate the session, but to fall back on links if cookies are disabled in the visitor's browser.
- **None** - If the `-useNone` parameter is specified in the `session_start` method then Lasso will not attempt to propagate the session. The techniques described later in this chapter for manually propagating the session must be used.

Session Tasks

To start a session:

The following example starts a session named `Site_Preferences` with an idle expiration of `24` hours (`1440` minutes). The session will be tracked using both cookies and links.

```
session_start('Site_Preferences', -expires=1440, -useLink, -useCookie)
```

To add variables to a session:

Use the session_addVar method to add a variable to a session. Once a variable has been added to a session its value will be restored when session_start is next called. In the following example a variable RealName is added to a session named Site_Preferences.

```
session_addVar('Site_Preferences', 'Real_Name')
```

To remove variables from a session:

Use the session_removeVar method to remove a variable from a session. The variable will no longer be stored with the session and its value will not be restored in subsequent requests. The value of the variable in the current page will not be affected. In the following example a variable RealName is removed from a session named Site_Preferences.

```
session_removeVar('Site_Preferences', 'Real_Name')
```

To delete a session:

A session can be deleted using the session_end method with the name of the session. The session will be ended immediately. None of the variables in the session will be affected in the current request, but their values will not be restored in subsequent requests. Sessions can also end automatically if the timeout specified by the -expires keyword is reached. In the following example the session Site_Preferences is ended.

```
session_end('Site_Preferences')
```

To pass a session in an HTML form:

Sessions can be added to URLs automatically using the -useLink keyword in the session_start method. In order to pass a session using a form, a hidden input must be added explicitly. The hidden input should have the name -lassosession and a value of Session_Name:session_id. In the following example, the ID for a session Site_Preferences is returned using session_id and passed explicitly in an HTML form.

To track a session using link decoration only if cookies are disabled:

The following example shows how to start a session using links if cookies are disabled. The -useAuto parameter will first try setting a cookie and decorate the links on the current page. If the session cookie is found on subsequent page loads, then it will be used and the links on the page will not be decorated. If the cookie cannot be found then links will be used to propagate the session.

```
session_start('Site_Preferences', -useAuto)
```

Session Example

This example demonstrates how to use sessions to store user-specific values which are persisted from page to page. It displays a form which the user can manipulate. The user's selections are saved from one request to the next.

Sessions will be used to track the visitor's name, email address, favorite color, and favorite forms of FTL travel in session variables.

```
[
  local(wr = web_request,
    sessionName = 'sessions_example')
  // start the session
  session_start(#sessionName)
  if (session_result(#sessionName) != 'load')
    // the session did not already exist,
    // so set the variables we want to be saved
    session_addVar(#sessionName, 'realName')
    session_addVar(#sessionName, 'emailAddress')
    session_addVar(#sessionName, 'favoriteColor')
    session_addVar(#sessionName, 'hyperDrive')
    session_addVar(#sessionName, 'warpDrive')
    session_addVar(#sessionName, 'wormHole')
    session_addVar(#sessionName, 'improbabilityDrive')
    session_addVar(#sessionName, 'spaceFold')
    session_addVar(#sessionName, 'jumpGate')
    // initialize our vars to empty values
    var(realName, emailAddress, favoriteColor,
      hyperDrive, warpDrive, wormHole,
      improbabilityDrive,  spaceFold, jumpGate)
  else (#wr->param('submit'))
    // the session existed
    var(realName = #wr->param('realName'))
    var(emailAddress = #wr->param('emailAddress'))
    var(favoriteColor = #wr->param('favoriteColor'))
    var(hyperDrive = #wr->param('hyperdrive'))
    var(warpDrive = #wr->param('warpdrive'))
    var(wormHole = #wr->param('wormhole'))
    var(improbabilityDrive = #wr->param('improbabilitydrive'))
    var(spaceFold = #wr->param('spacefold'))
    var(jumpGate = #wr->param('jumpgate'))
  /if
]
```

```
<html>
<body>
  <form action="<?= include_currentPath ?>" method="POST">
    Your Name:
    <input type="text" name="realName" value="<?= $realName ?>" />
    <br />
    Your Email Address:
    <input type="text" name="emailAddress"
      value="<?= $emailAddress ?>" />
    <br />
    Your Favorite Color:
    <select name="favoriteColor">
      <option value="blue"<?=
        $favoriteColor == 'blue'?
          ' selected="yes"'
        ?>> Blue </option>
      <option value="red"<?=
        $favoriteColor == 'red'?
          ' selected="yes"'
        ?>> Red </option>
      <option value="green"<?=
        $favoriteColor == 'green'?
          ' selected="yes"'
        ?>> Green </option>
    </select>
    <br />
    Your Favorite Forms of Superluminal Travel:<br />

    <input type="checkbox" name="hyperdrive" value="hyperdrive"
      <?= $hyperDrive? ' checked="yes"' ?> /> Hyper Drive<br />
    <input type="checkbox" name="warpdrive" value="warpdrive"
      <?= $warpDrive? ' checked="yes"' ?> /> Warp Drive<br />
    <input type="checkbox" name="wormhole" value="wormhole"
      <?= $wormHole? ' checked="yes"' ?> /> Worm Hole<br />
    <input type="checkbox" name="improbabilitydrive"
      value="improbabilitydrive"
      <?= $improbabilityDrive? ' checked="yes"' ?>
      /> Improbability Drive<br />
    <input type="checkbox" name="spacefold" value="spacefold"
      <?= $spaceFold? ' checked="yes"' ?> /> Space Fold<br />
    <input type="checkbox" name="jumpgate" value="jumpgate"
      <?= $jumpGate? ' checked="yes"' ?> /> Jump Gate<br />
    <br />
    <input type="submit" name="submit" value="Submit" />
    <a href="<?= include_currentPath ?>">Reload This Page</a>
  </form>
</body>
</html>
```