

# Cookies and Sessions

# Cookies

- A cookie is a name/value pair that is stored in a browser.
- On the server, a web application creates a cookie and sends it to the browser. On the client, the browser saves it back to the server every time it accesses a page from that server.
- By default, cookies only last until the user closes the web browser. However, cookies can be set to persist in the user's browser for up to three years.
- Uses for cookies
  - To allow users to skip login and registration forms.
  - To customize page.
  - To focus advertising like banner ads that target the user's interest.

# Creating cookies

- The `setcookie()` function is used to set a cookie. Note: The `setcookie()` function must appear BEFORE the `<html>` tag.
  - `setcookie(name, value, expire, path, domain);`
    - `name` – the name of the cookie
    - `$value` – the value of the cookie. The default is the empty string.
    - `$expire` – the expiration date of the cookie as a timestamp. If set to 0, the cookie expires when the user closes the browser. The default is 0.
    - `$path` – if set to `'/'`, the cookie is available to all directories within the entire domain. The default is the directory of the PHP file that's setting the cookie.
    - `$domain` - to make the cookie available on all subdomains of `example.com` then you'd set it to `".example.com"`. Setting it to `www.example.com` will make the cookie only available in the `www` subdomain .
    - `$secure` – if true, the cookie is only available if it is sent using HTTPS. The default is false.
    - `$httponly` – if true, the cookie is only available through the HTTP protocol and not through client-side scripting language such as JavaScript. The default is false

# PHP Sessions

- A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.
- When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.
- A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.
- Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

# session\_start()

- Starts a new session or resume a previous section. Returns true if successful and false otherwise. This function must be called before the page sends any HTML output to your application.
- By default, a session uses a per-session cookie to associate a browser with the data for its session. However, you can use the `session_set_cookie_params` function to customize the cookie for the session.
- the `session_set_cookie_params` must be called before the `session_start` function.
  - `$duration = 60 * 60 * 24 * 365; // 1 year in seconds`
  - `session_set_cookie_params($duration , '/');`
  - `session_start();`

# session\_set\_cookie\_params

- session\_set\_cookie\_params — Set the session cookie parameters
- session\_set\_cookie\_params(\$lifetime, \$path, \$domain, \$secure, \$httponly);
  - lifetime
    - Lifetime of the session cookie, defined in seconds.
  - path
    - Path on the domain where the cookie will work. Use a single slash ('/') for all paths on the domain.
  - domain
    - Cookie domain, for example 'www.php.net'. To make cookies visible on all subdomains then the domain must be prefixed with a dot like '.php.net'.
  - secure
    - If TRUE cookie will only be sent over secure connections.
  - httponly
    - If set to TRUE then PHP will attempt to send the httponly flag when setting the session cookie.

# Using the SESSION variable

- Once you start a session, you can use the autoglobal `$_SESSION` variable to set and get the user's data for a session. This variable is an associative array.
  - `$_SESSION ['customer_name'] = 'Peter';`
  - `$customer_name = $_SESSION ['customer_name'] ;`
  - `$_SESSION ['cart'] = array();`
  - `$_SESSION ['cart']['book'] = 'C++';`
  - `$_SESSION ['cart']['computer'] = 'Apple';`
- You can use the `unset` function to remove an item from the `$_SESSION` array.
  - `unset( $_SESSION['cart']);`
- You can remove all session variables:
  - `$_SESSION = array();`

# Ending a session

- A session ends when the user closes the browser, when a specified amount of time elapses without a request, or when the code calls the `session_destroy` function.
- To remove all data associated with the session from the client and the server, you can clear the session data from memory, call the `session_destroy` function, and use the `setcookie` function to delete the session cookie.
- The `session_name()` gets the name of the session cookie, by default is "PHPSESSID".
- To end a session.
  - `$_SESSION = array();` // clear session data from memory
  - `session_destroy();` // clean up the session ID
- To delete the session cookie from the browser
  - `$name = session_name();`
  - `setcookie($name, "", time()-3600);` // set the expiration date to one hour ago



# Other session functions

- Get the name of the session cookie
  - `$name = session_name();`
- Get the value of the session ID
  - `$id = session_id();`
- Set the session ID
  - `session_id('myid1234');`
- Creates a new session ID for the current session. This function can be used to help session hijacking.
  - `session_regenerate_id();`
- Ends the current session and saves session data in case you need to redirect to other website.
  - `session_write_close();`

# Case Study: Bookstore using Cookies and session



**My Bookstore**

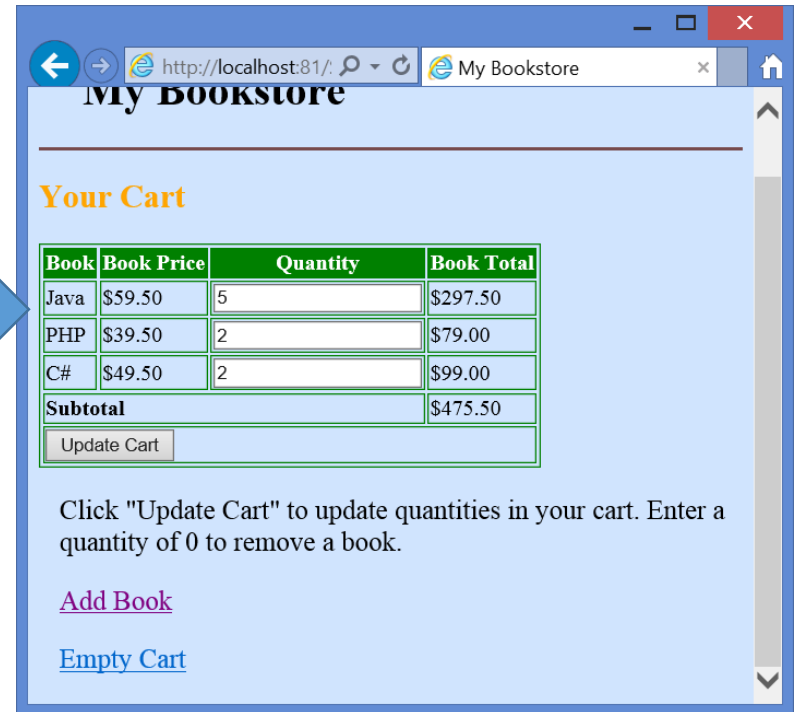
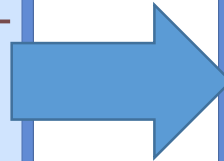
---

**Add Book**

Title: C# (\$49.50) ▾

Quantity: 1 ▾

[View Cart](#)



**My Bookstore**

---

**Your Cart**

Book	Book Price	Quantity	Book Total
Java	\$59.50	5	\$297.50
PHP	\$39.50	2	\$79.00
C#	\$49.50	2	\$99.00
<b>Subtotal</b>			\$475.50

Click "Update Cart" to update quantities in your cart. Enter a quantity of 0 to remove a book.

[Add Book](#)

[Empty Cart](#)

# The MVC for Bookstore App

- Controller (index.php)
  - Actions
    - show\_add\_book
    - show\_cart
    - add
    - update
    - empty\_cart
  - Views
    - add\_book\_view.php
    - cart\_view.php
  - Model
    - \$books for array of book info
    - \$\_SESSION['shop\_cart'] for a shopping cart of book orders