

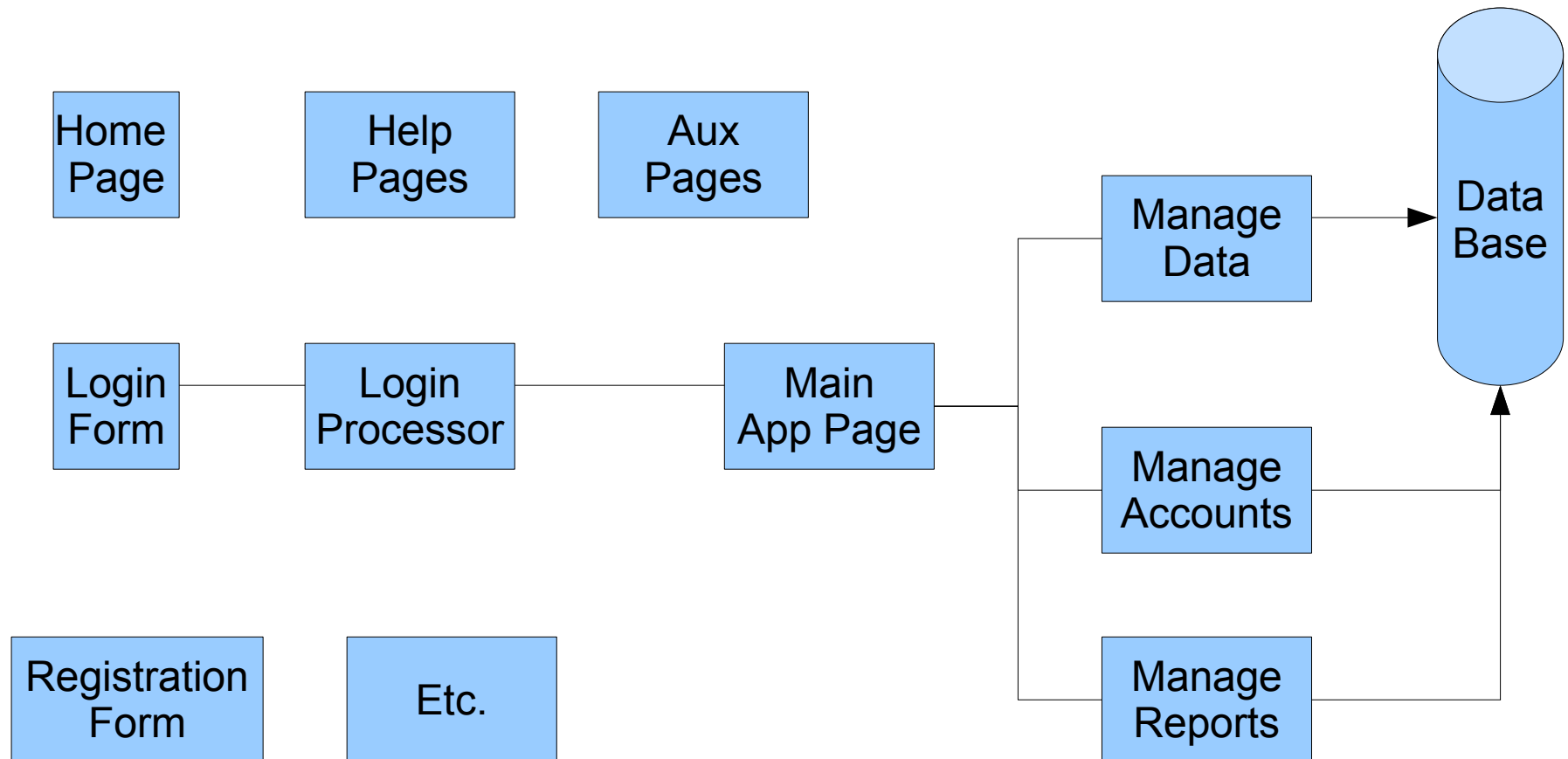
Cookies and Sessions

Session Management in Web Applications

Overview

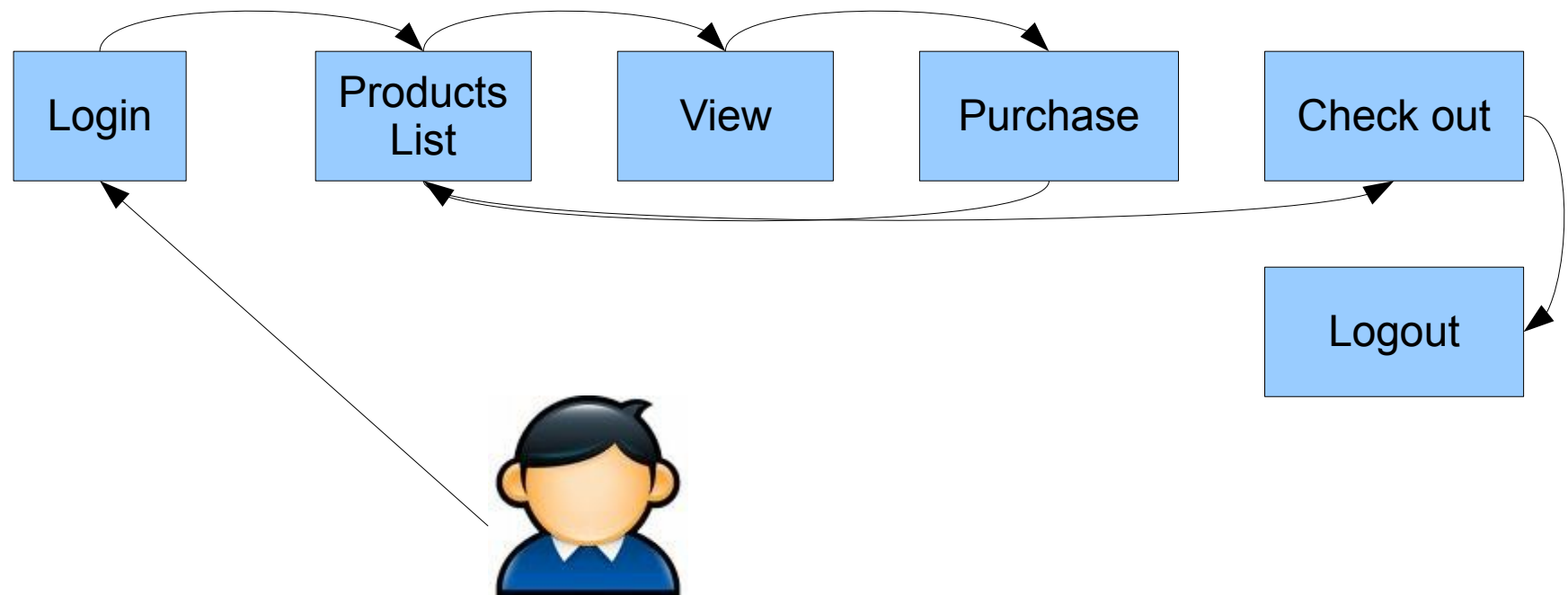
- A web application is composed of multiple web pages:
 - Forms
 - “Processing” scripts/servlets
 - Reports
 - Information pages (i.e. static pages, help pages, etc).
 - Etc.
- Most web applications require a database management system for the data they use/process.

Overview:



Session Management

- Keeping track of user's activity across sessions of interaction with a system.



Session Management

- A **session** is the “dialogue” between a user and the system, where important information may arise and be reused as the dialogue continues.
 - i.e. whatever activity does from login to and logout from an application (inclusive) can be considered a session.

Session Management

- Consider the following:
 - User uses login form. Application must remember user name.
 - User views products.
 - User clicks on “buy”, application must remember what the item is and how many.
 - User goes back to window shopping.
 - User does something else on site.
 - User checks out his online “shopping cart”.
 - User logs out.

Session Management

- The main challenge is:
 - How your application remembers pertinent data about your user or data used in processing routines.

HTTP and Session Management

- HTTP is a stateless/sessionless protocol.
 - Every HTTP request (and response) is independent from any previous requests coming from the same client.
 - HTTP does not “remember” what the last request from the same user was.

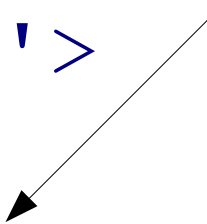
Strategies in Storing Session Data

- Hidden form element technique.
 - Use of the input type='hidden' HTML widget to store data used in processing forms.
- Cookies (Client based)
 - Server generated, client maintained session information
- Server-based Session Management
 - Server maintained session information

Hidden Form Element

```
<form name='myform'
      action='process.php'>
  <input type='hidden'
        name='proc_value'
        Value='125544576' />
  ... other form elements
</form>
```

Use the hidden type to include “nonvisible” data to be used in processing.

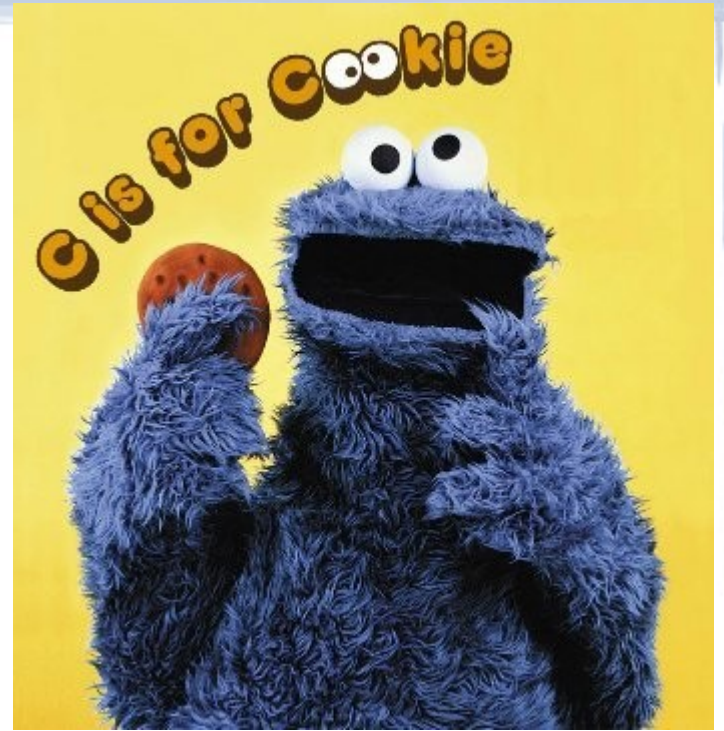


Hidden Form Element

- Easy to implement.
- Process the “additional” information just like any other submitted POST or GET data.
- Works only with forms.

Cookies

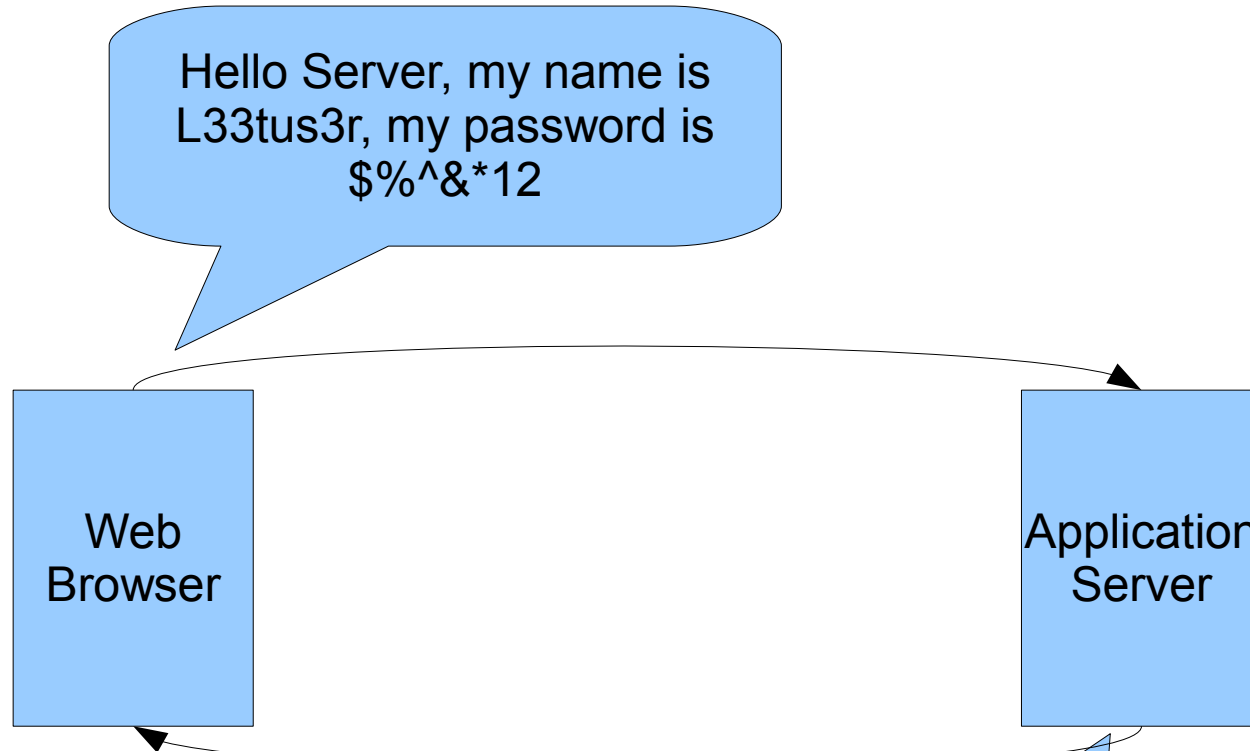
- Aka Tracking cookie,
- Aka browser cookie
- Aka HTTP cookie
- Information stored on a user's computer by a web browser FROM a web server (web application)



Cookies

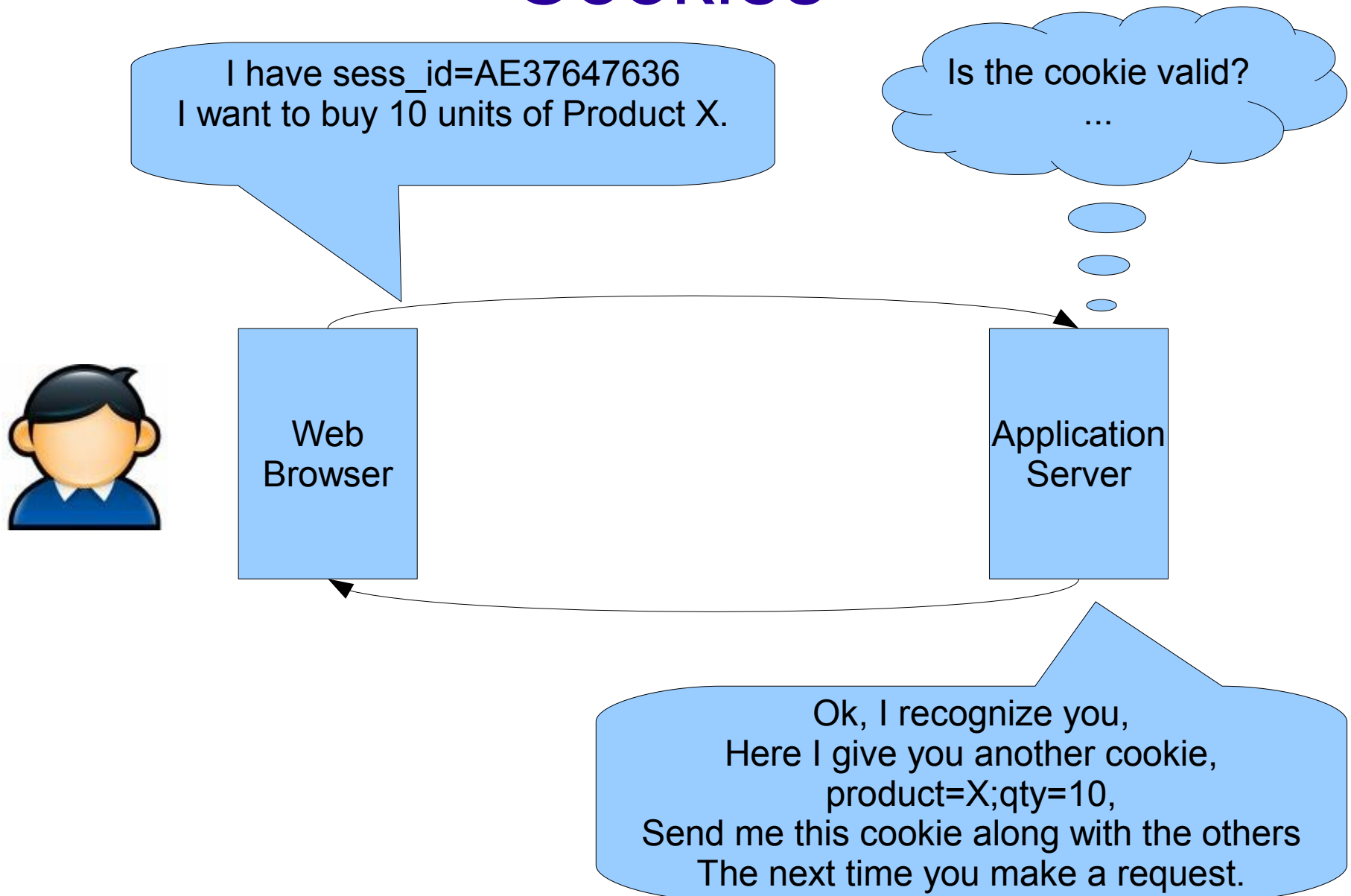
- Name-value pairs containing application-specific information.
 - Ever wondered how the “Remember Me On This Computer” in login forms work?
 - Can contain any information the server wants the client to remember*

Cookies



Hi L33tusr, I give you
sess_id=AE37647636
Cookie, please send
Me this everytime you
Make a request so I know its you.

Cookies



Cookies: HTTP Response

- The server sends the cookie via the HTTP Response with the Set-Cookie header:

```
HTTP/1.1 200 Ok
```

```
Content-type: text/html
```

```
Set-Cookie: cookienname=cookievalue
```

```
(content of page)
```


Cookies: HTTP Request

- From then on, the client sends the cookies it receives from the server in subsequent requests.

```
GET /process.php HTTP/1.1
```

```
Host: www.example.com
```

```
Cookie: cookienam=cookievalue
```

```
Accept: */*
```

Cookie attributes:

- Name – identifier for the cookie.
- Expiration Date – as to when the cookie is valid (cookie is considered “stale” past this date).
- Path and domain name – path within the domain and the domain where the cookie is valid.
- Secure – if cookie is passed only through secure connection (HTTPS)

Cookie Attributes:

- Example:

```
Set-Cookie: sess_id=A34511; expires=Fri, 31-  
Dec-2010 23:59:59;  
path=/;domain=.example.com; secure=true;  
HttpOnly
```

//Web script/CGI sets this up automatically
//via standard API calls.

Cookies: Don'ts

- Some US laws on auditing cover things like not saving “sensitive” information in cookies (such as SSNs).
- Do not save passwords in cookies.
- Do not save credit card information in cookies.
- General good practice is to send session identifier to the client as cookie.
 - Identifier must be random.
 - Server must keep track of this identifier.

Cookies in PHP

- **setcookie(name,value,expire,path, secure, httponly)**
 - `setcookie('UserName','L33tus3r', 0, '/',
'example.com', false,false);`
 - Cookies must be sent before any output is generated from your script.
 - This means that `setcookie` must be called before any text (even a single space) is sent to the client.

Cookies in PHP

```
<?php  
//do it here  
setcookie(....);  
echo "output";  
?>
```

Important!



Cookies in PHP

- In subsequent pages, cookie values from requests can be retrieved using the `$_COOKIE` variable.
 - `$_COOKIE['id']` where id is the name of the cookie set.
- You may call `setcookie` multiple times to set multiple cookies.

Misconceptions about Cookies

- Cookies are viruses.
 - Cookies are not executables, they are text files saved somewhere where the browser can access them.
- Cookies generate pop-ups
 - Javascript does that.
- Cookies used for spamming.
- Cookies are used for advertising.

Cookies

- Cookies may be disabled in the browser.
- Previous cookies may be deleted via browser preferences menu.

Security Issues with Cookies

- Cookies is not an accurate tool for identification.
- Cookie hijacking.
 - Cookies can be stolen via packet sniffing, cross site scripting (XSS).
- Cookie theft
- Cookie poisoning
- Cross-site Cooking

Server Based Session Maintenance

- How it works:
 - Makes use of cookies! But only a hard to guess, unique session identifier is sent to the client (usually a long integer).
 - Everytime a request is made, the session ID is sent as a cookie in the request.
 - The server maintains information regarding that session ID. i.e. instead of sending information about a purchase, that info is saved on the server instead and made accessible via the user's session ID @ the server side.

Using Server Based Session Management in PHP

```
<?php  
session_name('Private'); //--  
session_start(); //must call  
//we can use $_SESSION var to  
//access or set session values  
$_SESSION['username']='L33tus3r';  
//you can now use the above other  
// pages as long as session is  
//active  
?>
```

PHP Session functions

- `session_id()`
 - Gets or sets the session id
- `session_destroy()`
 - Invalidates session, erases saved data.
 - Used in logouts see PHP manual for more details.
- You can use `unset($_SESSION['name'])` to remove session values stored.

Notes:

- Default behavior for PHP is to send session id to client as cookie.
- If cookies are not available at the browser, it is encoded with the URL instead.

Security Issues

- Session ID hijacking
 - Same as cookie hijacking.