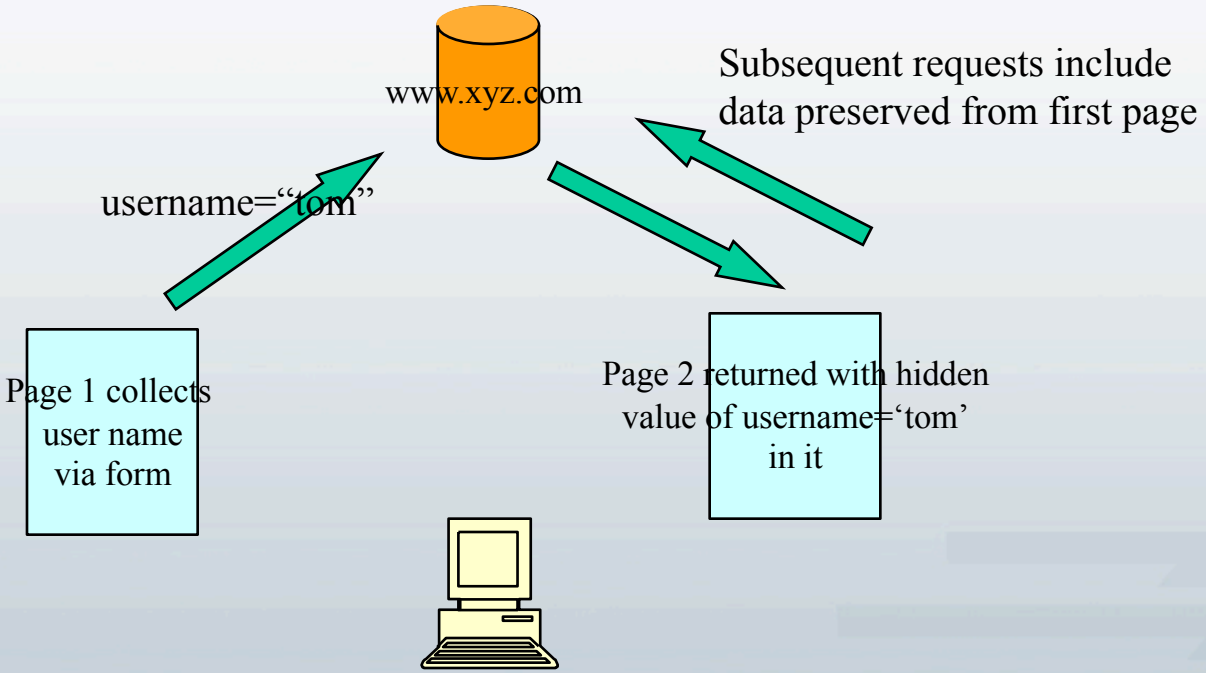
State and Session Management

Addressing HTTP Statelessness

- HTTP being stateless, in other words having no memory from page view to page view, can make Web programming a hassle. A variety of techniques to address this are used including:
  - Hidden Form fields
  - Dirty URLs
  - Cookies
    - Memory or session cookies
    - Persistent cookies
  - Wrongly - client tech
- Fortunately, most modern programming environments such as PHP, ColdFusion, ASP.NET, JSP, etc. provide a session concept which will abstract away many of the details of state preservation.
  - Sessions are most often implemented as cookies which store a unique ID referencing data stored on the server

Hidden Form Fields

- Forms contain <input type="hidden" name="cartid" value="78Ccad786"> this data is passed along with submission
- Subsequent page loads are rewritten with hidden fields in them



Hidden Form Field Notes

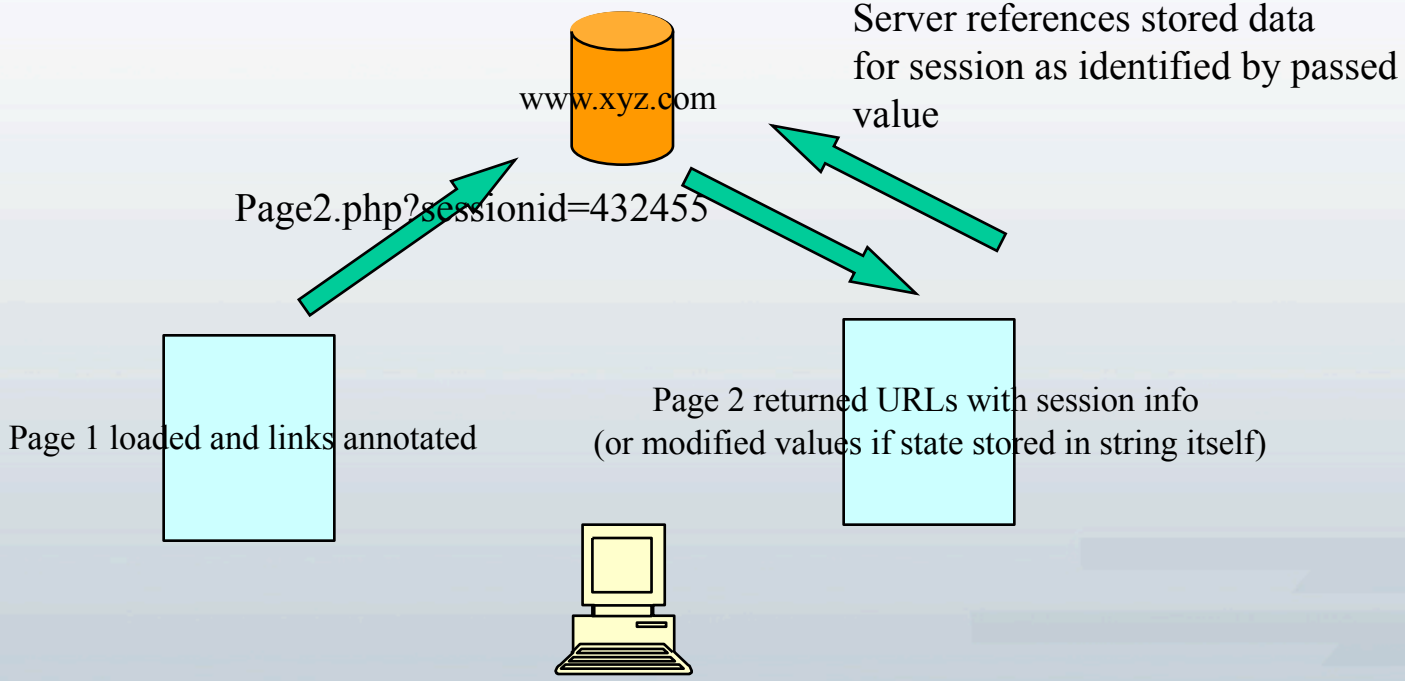
- Open for easy parameter tampering
  - Countermeasure: integrity checks
- Does not preserve state information across a user visit
- Requires that a form be used to trigger the pass back of the hidden data
  - .NET uses hidden form fields (a viewstate value) and wrap the whole page in a single giant form

State Preservation via "Dirty URLs"

- All links are rewritten with query strings in them

```
<a href="page1.php?sessionid=432455">Page 1</a>
<a href="page2.php?sessionid=432455">Page 2</a>
```

- All forms pass the value via the query string as well (either direct action change or via hidden field)

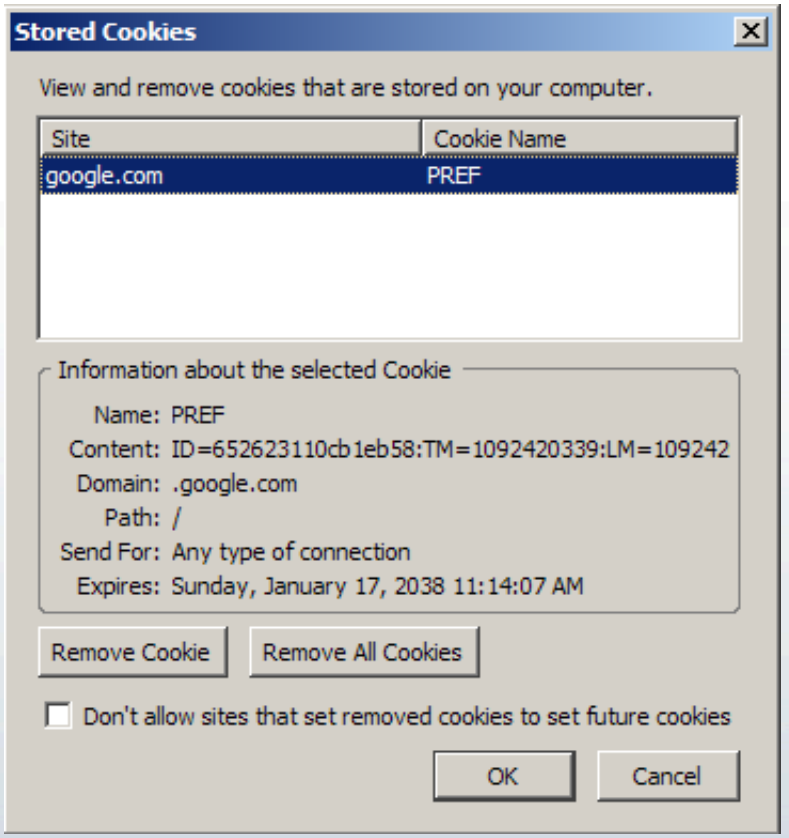


Dirty URL state management

- Open for easy tampering
  - Countermeasure: integrity checks
- Does not preserve state information across a user visit
- Does not promote URL usability, marketing, etc.
- Without dereferencing to server stored data has significant limit to the amount of data that can be saved (URL limit around 2K)

Cookies

- Cookies are simple short pieces of data stored on a user's system often in a file (cookies.txt) or a directory C:\Documents and Settings\userid\Cookies with individual cookie files
  - Example contents of a Google cookie
    - `PREFID=6b476a56502ce137:TM=1088552548:LM=1088552548:S=FUe7V3k3CtQYjEeHgoogle.com/1536261887833632111634405477774429646386*`
  - Notice that often cookies do not store the actual data to be tracked but an id that references data stored on a server

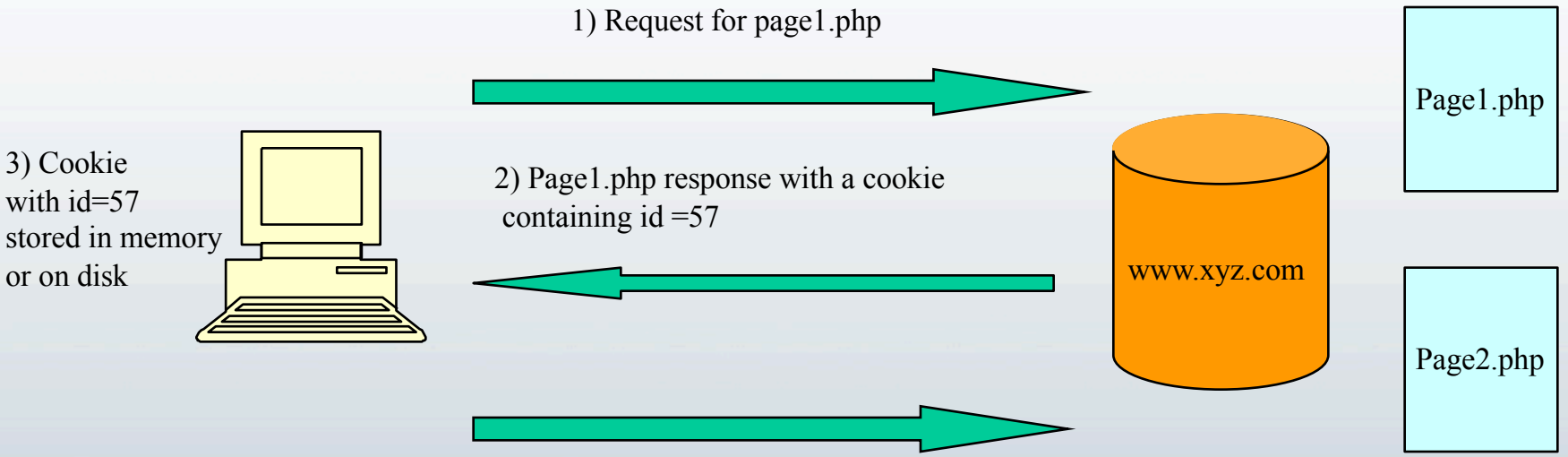


Cookies

- A cookie is typically issued to a browser via the HTTP header
  - Set-Cookie: *NAME*=*VALUE*; expires=*DATE*; path=*PATH*; domain=*DOMAIN_NAME*; secure
- Most of the time we would not set cookies manually in an HTTP response (though you could), instead the Web programming environment you employ will provide some command.
  - To issue a cookie in PHP use `setcookie(name, value);`
  - Example: `setcookie('leadstooge','moe');`
- However, not that JavaScript can also easily set cookies client-side via the `document.cookie` object

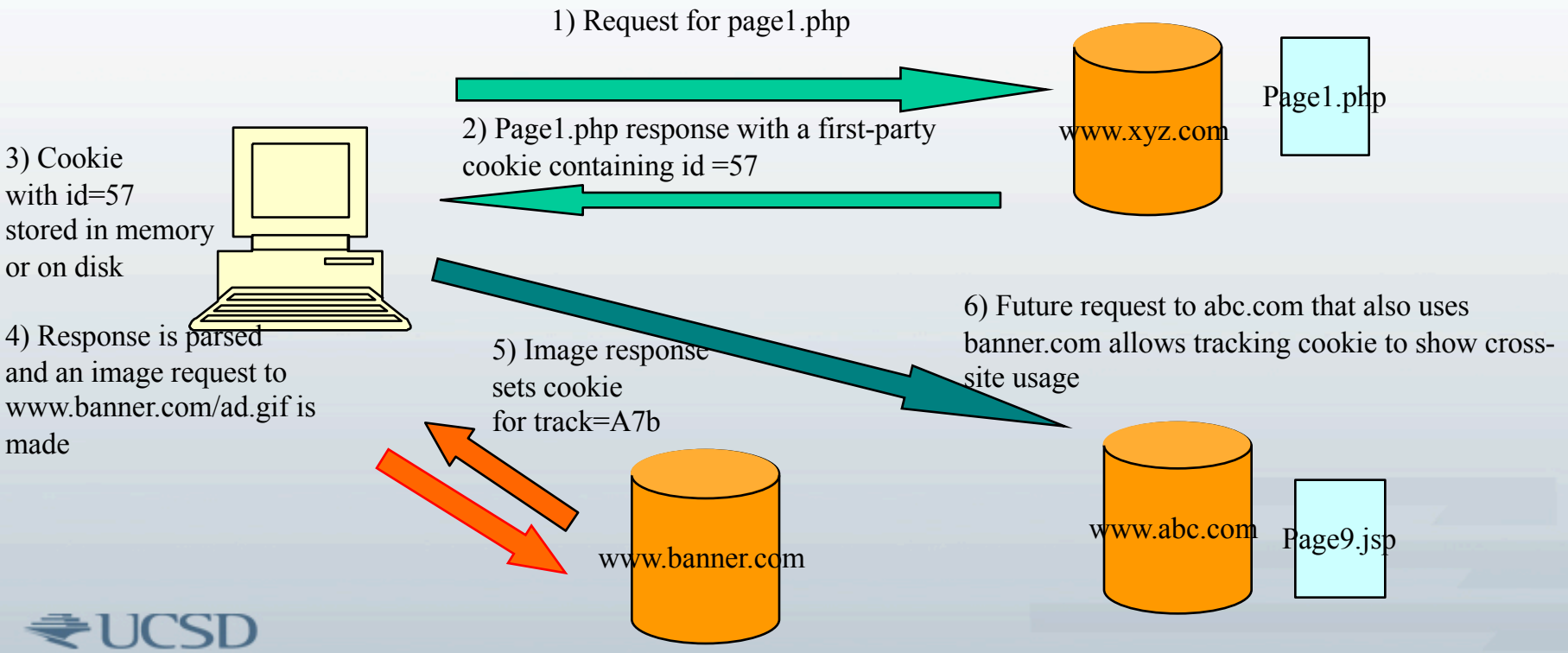
First Party Cookie Example

- Once the cookie is set it is transferred back and forth from the browser to the site for every request within the domain and path that it is set for.



Third Party Cookie Example

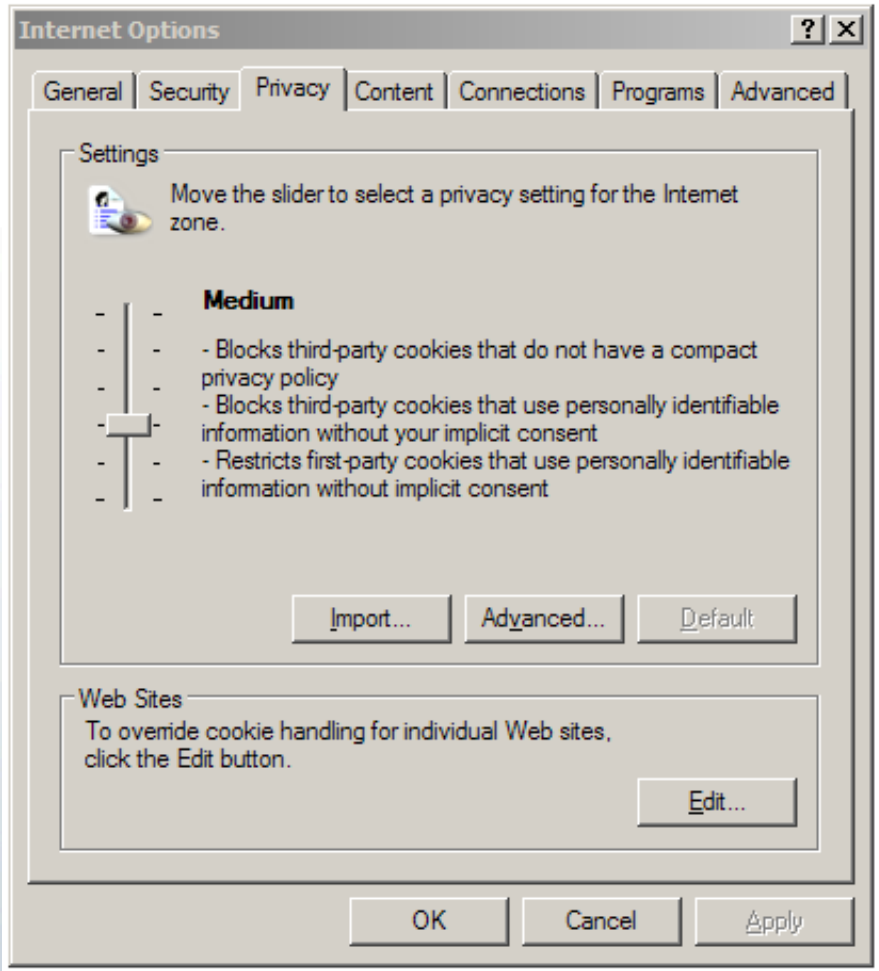
- Cookies can be set by any HTTP request including images, scripts, CSS, etc.
  - This is often used in the case of Third-party cookie issued from a banner ad or tracking site

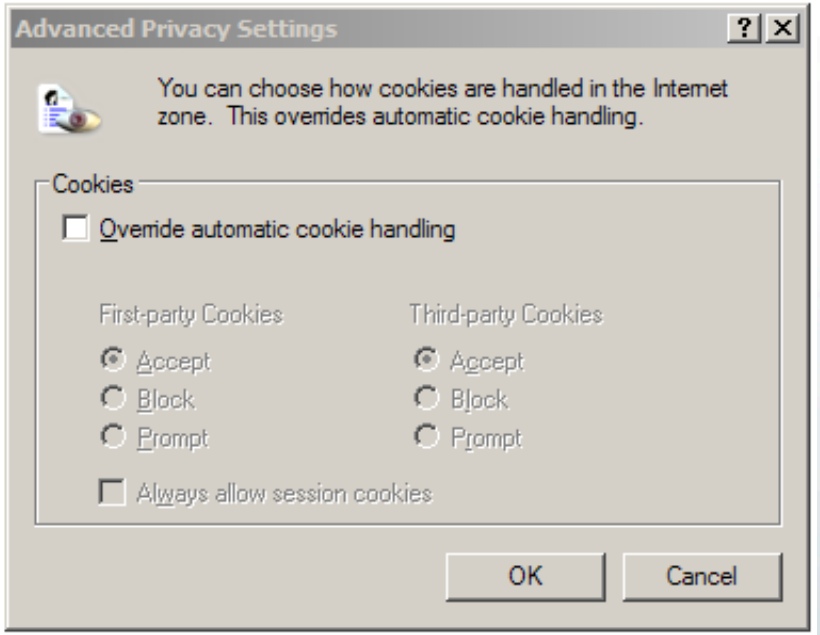


Privacy Concern with Cookies

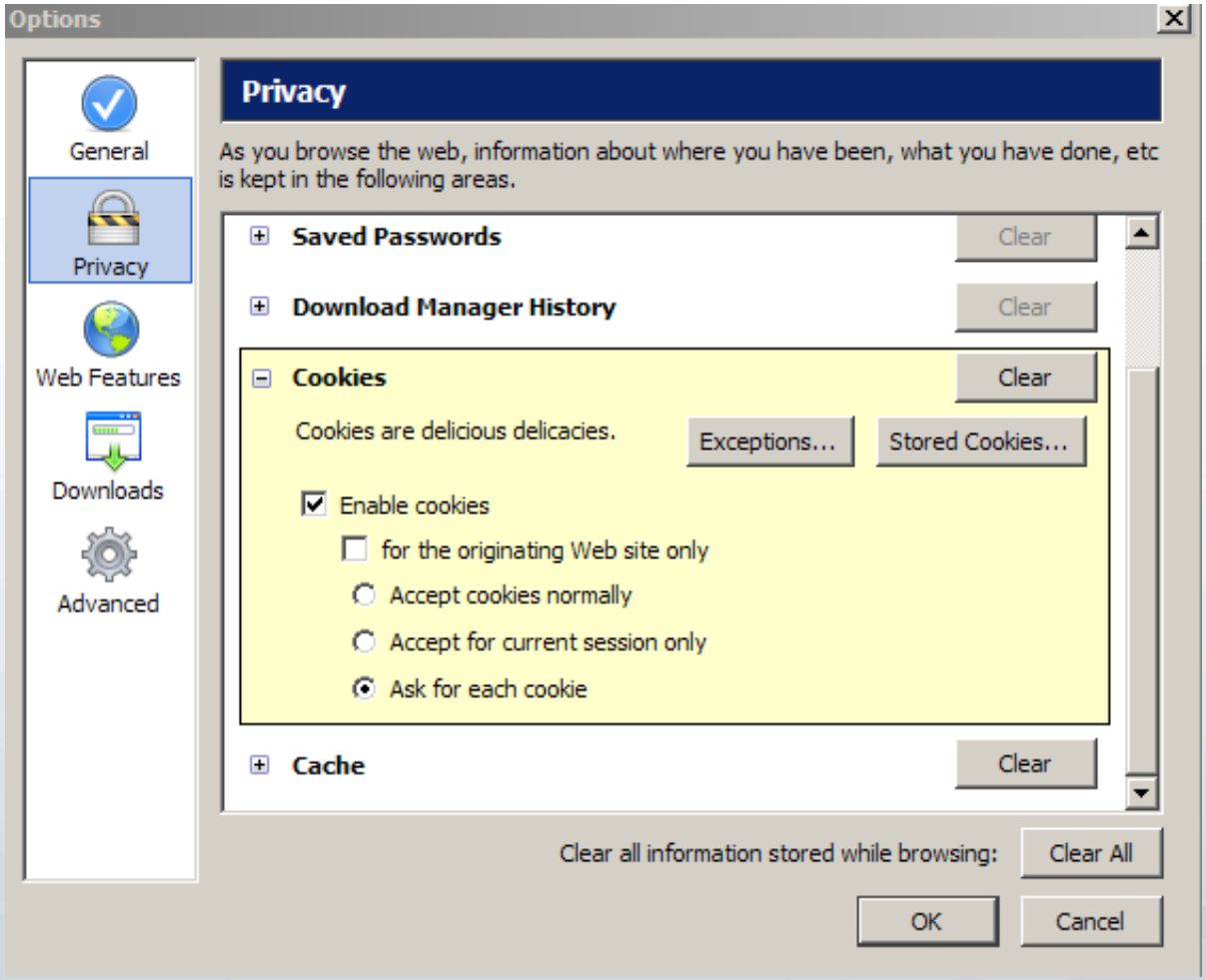
- Third party cookies represent a possibility to track user activity across a range of sites
  - Typically these sites are part of some “network” – like an ad network
- The cookies are also set often via a clear pixel GIF so the user may not see what they should be trying to block
  - These cookie setting clear pixels are dubbed “Web bugs” or just “bugs”
  - Typically they are used in a hosted Web analytics services like Hitbox from WebSideStory
- The privacy concern with cookies come from the fact that you may register at a particular site like the last example xyz.com and they then associate your personal information with the cookie-id. If the personal information is then shared to other network participating sites they can start building a detailed profile on your usage habits
  - Because of this many users want to block third party cookies, but the real problem is giving out your private information and knowing what happens to it afterwards
  - We lack “informational self-determination” online which is unfortunate
  - Browsers, tools, and technology like P3P help but fundamentally once again we see a social issue and not a technical one.

IE Cookie Related Settings



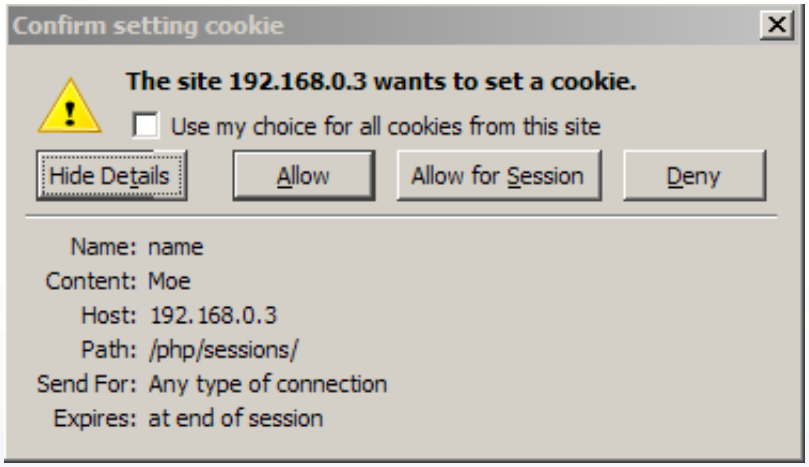


Firefox Cookie Related Settings



Cookie Example – cookie1.php

```
<?php

 setcookie('name','Moe');
 setcookie('numstooges','3');

?>
<html>
<head><title>Simple Cookie</title></head>
<body>
<h1>I just "baked" a cookie</h1>


<a href="cookie2.php">See what's in the cookie</a>


<form>
  <input type="button"
   value="JS Cookie Check"
   onclick="alert(document.cookie);">
</form>

</body>
</html>
```



Cookie Example – cookie2.php

```
<html>
<head><title>Show Cookie</title></head>
<body>
<?php

print ("There are ". $_COOKIE['numstooges'] ." stooges and the
   main one is named " . $_COOKIE['name']);
?>
</body>
</html>
```

- The cookie set in this particular case is a *session cookie* or *memory cookie*
  - It will be destroyed once the browser session is closed down
  - It is not shared between browsers, but it is likely shared between browser windows
  - These types of cookies are not as troubling from a privacy point of view and are necessary to make a typical modern Web site (unfortunately some users still deny them)

Cookie Limits

- The browser and server may put some limits on cookies
  - Commonly held that browser:
    - Can store a limited number of cookies (300?)
      - Cookies received after browser limit are deleted starting the least used
    - A single cookie can be no more than 4K in size
      - Cookies are trimmed down to 4K if they exceed that size
  - Commonly held that a browser store –or- server will accept no more than 20 cookies for a particular host.domain combo
      - Though web.xyz.com and store.xyz.com are different and thus could have 20 each
  - The reality is these limits may not in fact be true as they are implementation dependent

- Commonly held development practice suggests using as few cookies per domain and only storing a sessionid or related value with the bulk of tracked information stored server side.

Setting Persistent Cookies

- Specify some expiration date
  - In PHP `setcookie('name','value'expiration,'path', 'domain',secure);`
  - Expiration stored in number of seconds since Epoch
  - Examples

```
setcookie('color','green',time()+3600);
  // expires in 1 hour

setcookie('userid','admin',time()+3600, '/admin');
  // limit to admin directory path

setcookie('userid','admin',time()+3600,'/admin',
            '', 1);
  // use security in transmit
```

Setting Persistent Cookies

```
<?php

 setcookie('favoritestooge','Moe',time()+3600);

?>
<html>
<head><title>Simple Cookie</title></head>
<body>
<h1>I just "baked" a persistent cookie so check your disk</h1>

</body>
</html>
```

```
C:\Documents and Settings\Thomas Powell\Application Data\Mozilla\Firefox\Profile
s\default.xab>more cookies.txt
# HTTP Cookie File
# http://www.netscape.com/newsref/std/cookie_spec.html
# This is a generated file!  Do not edit.
# To delete cookies, use the Cookie Manager.

192.168.0.3     FALSE   /php/sessions/  FALSE   1092815507      favoritestooge
Moe
.google.com     TRUE    /       FALSE   2147368447      PREF    ID=652623110cb1e
b58:TM=1092420339:LM=1092420339:S=vUIiM9kTm-wGkcsK
```

Deleting Cookies

- Session cookies go away on browser close and persistent cookies are blasted upon expiration you can do it early by setting a value to blank or modifying the expiration or even better both!

```
setcookie('favoritestooge','');
setcookie('favoritestooge','',time()-3600);
```

- You do want to encourage a user to logout of a Web application rather a click away because you want to improve security
  - Consider the problem of a public use system with still active cookies from banking sites, secure extranets, etc.

Sessions

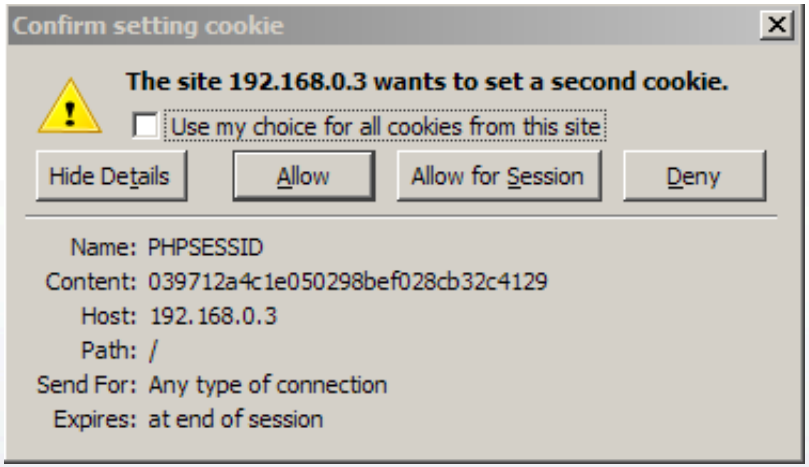
- While it is possible to do all the management of most sites using cookies (or even hidden fields or dirty URLs) it is much better to use sessions

- A well implemented session management system abstracts away how data is preserved from page to page (typically in a cookie) and stores the saved information server side only referencing via a SESSIONID value in the cookie
  - You could of course build this easily enough by storing some database-id in a cookie yourself and every page load doing a query against the database reading or modifying data in regards to the user's state

- A significant benefit of sessions not storing the data in the cookie is that it helps guard against session hijacking
  - Assuming that session ids are not easily guessable
  - Even better to "brand" the session to the user somehow

Sessions in PHP

- Use `session_start()` and PHP will issue a cookie with name PHPSESSID and some value
- You will use `session_start()` most likely on every page in your app



- It is possible to set a php.ini setting of session.auto_start so that you don't need to use `session_start()` all the time

- You can specify a different value other than PHPSESSID before by calling `session_name('MYID');` where MYID is your value and then calling `session_start();`

- To create a session variable use `$_SESSION['var'] = 'value';`

  Example: `$_SESSION['stooge'] = 'moe';`

Session Example in PHP

```
<?php
    session_start();
    $_SESSION['fav_stooge']='Moe';
    $_SESSION['num_stooges']=3;
?>
<html
<head>
<title>Session Fun</title>
</head>
<body>
<h1>Just set some session values</h1>

<a href="session2.html">Next page</a>
   <!-- eventually read this later on session3.php -->

</body>
</html>
```

Session Example in PHP Contd.

```
<html
<head>
<title>Session Fun</title>
</head>
<body>
<h1>Reading session variables</h1>
<?php
  session_start();
  print "There are ".$_SESSION['num_stooges'] ." and
  my favorite is ". $_SESSION['fav_stooge'];

?>
</body>
</html>
```

Deleting Session Info from PHP

- To delete an individual session variable in PHP use `unset($_SESSION['thevar']);` just as you would unset an arbitrary value in PHP

- To delete all session variables then just blow out the entire array like so `$_SESSION = array();`

- Removing all session data from the server can be accomplished with `session_destroy();`

- *Note: That to do anything with session values including destroying them you still have to call `session_start()` first.*

- *Just like cookies you really ought to clean up sessions upon "logout"*

PHP Session Details

- By default sessions uses memory cookies that expire upon browser close. You can use the `session_set_cookie_params(`*`expiration`*`,'path','domain',`*`secure`*`)` function to set the typical cookie values you might want to do

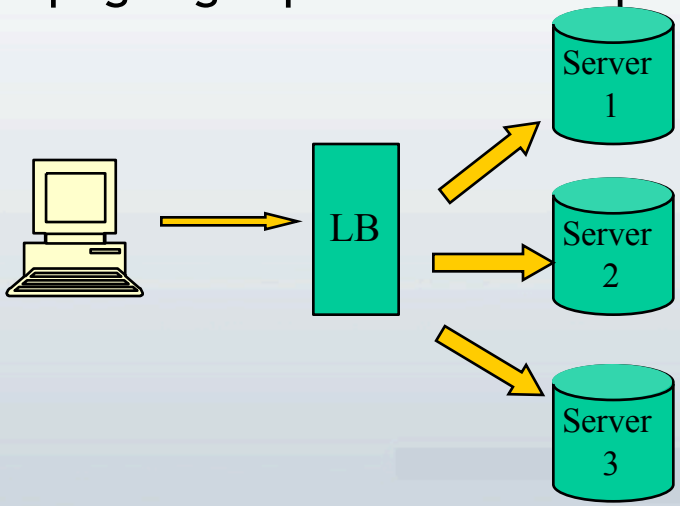
  - Example:

```
session_name('mysid');
session_set_cookie_params(3600,'/whoismgr','www.xyz.com');
session_start();
```

- Note that you do not have to manually calculate the time from EPOCH here you specify the number of seconds into the future before the cookie expires.

Session Considerations

- Session Hijack
  - Make the IDs unpredictable and tie to a browser
  - HttpOnly Cookies
- Cookies off
  - Default to a dirty that puts the SessionID into every link dynamically as a query string
  - Careful: not automatic and assume all pages get passed via script engine unlike a cookie
- Sessions in a Server Farm
  - How do you share the server stored data?
  - Don't
    - Sticky Server
  - Share it using
    - A backend db
    - Letting the load balancer hold it



Some other important ideas

- Reassociating cookies
  - Why your blank Vons club card doesn't save you as much as you think
- Using Flash persistence, browser specific or HTML5 persistence – we can abstract this - careful here
- Tricks using if-modified-science values for Web bugs
- Tracking with JavaScript, fingerprinting and simple or esoteric ideas like frames or window.name
- If you are going to worry – be consistent