Cookies and Session Management

Chapter Contents

- What are cookies.
- Using cookies with PHP.
- Session management.

Navigation Map

- What are cookies.
- Using cookies with PHP.
- Session management.

The Problem

- HTTP is a **stateless** protocol.
- This feature prevents the web server from keeping state on the clients, meaning that:
  - No contextual information is maintained about the client.
  - There is no way to associate requests from a particular client.
- In other words, the protocol has no memory of historic events.

The Problem

- This presents a problem to web application developers.
- A solution that will sit on top of the HTTP protocol, and will help maintain state information, was needed.

Navigation Map

- What are cookies.
- Using cookies with JSP.
- Session management.

What are Cookies?

- Cookies, by definition, are information bits that a server puts on a client computer so it can "remember" something about the client at a later time.
- Cookies are defined as a part of the "**HTTP state management mechanism**" (*RFC 2109*), and are commonly used for various tasks.

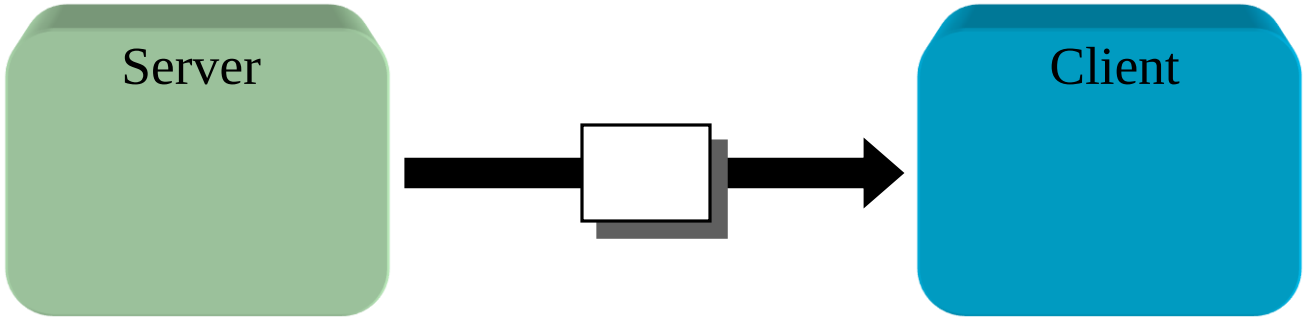
Cookies Mechanism

How does the cookies mechanism works:

1. On client request, the server creates the information and various associated attributes.

Cookies Mechanism

2. The server uses an HTTP response header named "**Set-Cookie**" to send the information to the client .



Cookies Mechanism

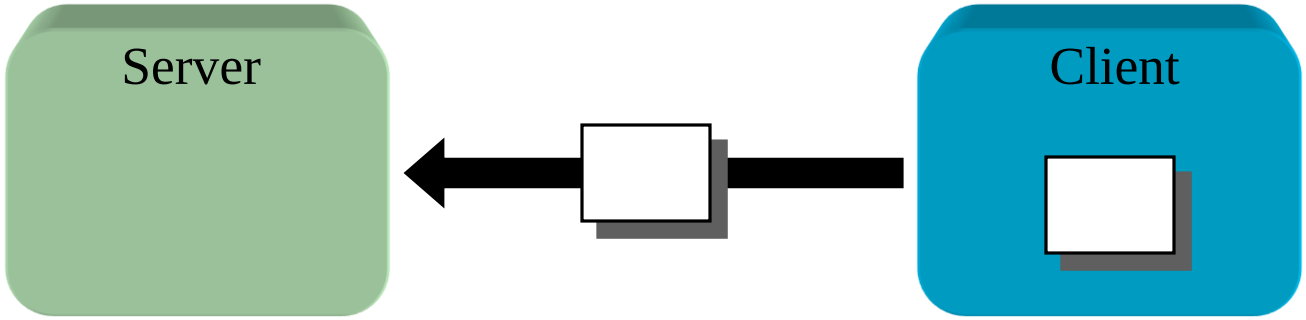
3. The client stores the information.

Cookies Mechanism

4. On every request, the client checks for applicable cookies.

Cookies Mechanism

5. If the client finds applicable cookies, it sends the information back using the "**Cookie**" HTTP request header.



Cookies Mechanism

6. The server gets the information from the client's request, and use it as needed.

Cookies Attributes

- Cookies information includes attributes, which allow better control over the cookie.

These attributes include:

- **Name** – The name of the cookie.
- **Value** – The value of the cookie.
- **Comment** – An associated comment, usually the cookie's purpose.
- **Domain** – Specify the domain for which the cookie is valid.
- **Path** – Specify the subset of URLs this cookie corresponds to.
- **Max age** – Defines the lifetime of the cookie in seconds.
- **Secure** – Security level for the cookie (yes / no).
- **Version** – Version of state management specification.

Using Cookies with PHP

Functions enable cookies manipulation:

setcookie(*name*, *value*, *expire*, *path*, *domain*, *secure*, *httponly*);

- Only the *name* parameter is required. All other parameters are optional.
- The setcookie() function must appear BEFORE the <html> tag.

Using Cookies with PHP

Retrieving a cookie value:

$_COOKIE[“cookie_name”]

Using Cookies with PHP

Example:

```
<?php
$cookie_name = "user"; $cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html><body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
} ?>
```

Cookies Problems

- Although cookies are useful, there are some limitations when using them:
  - Cookies require a-lot of work from the programmer.
  - The data stored is only textual.
  - Information travels back and forth, client to server and back.
  - Clients are expected to support only 20 cookies per web server and a total of 300.
  - Cookies might be limited to 4 KB each.

Cookies Uses

- In spite of the limitations, cookies are still used for various actions, such as:
  - Remembering users preferences over time.
  - Store advertising related material.
  - Automatic login to applications.
  - More …

Nevertheless, state management is better achieved by using **sessions**.

Navigation Map

- What are cookies.
- Using cookies with JSP.
- Session management.

What is a Session?

- A **session** is the presence of a single user inside the application.
- Every user that enters the application opens a new **session**, which is maintained until the user concludes the working period.
- In a given time period, a single **session** exists per single user.

Common Uses

- Some common uses for session management:
  - Applications that require a login procedure.
  - Application that serves clients differently upon authentication (example: web based mail applications).
  - A shopping-cart-based application.

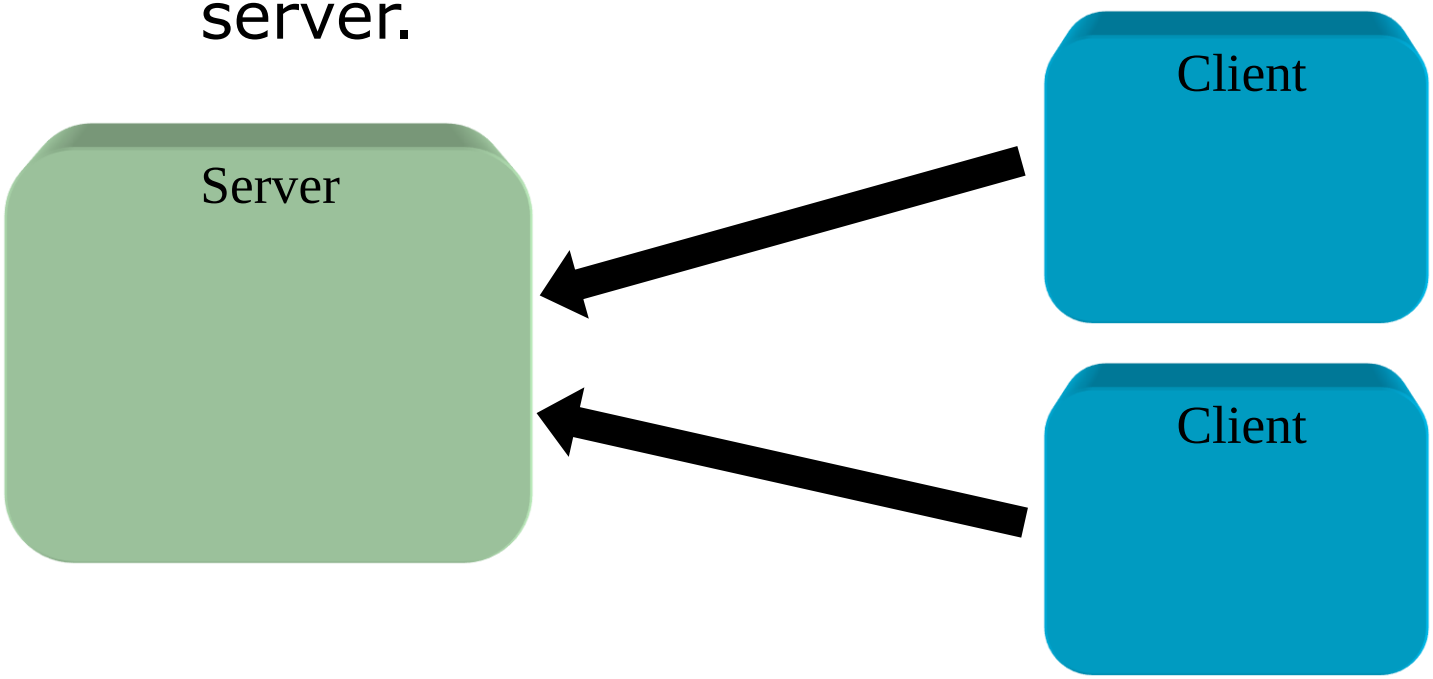
Session Management

- Session management is a technique used to keep track of each client's server connection.
- Finds a way around the HTTP stateless limitations.

Session Management

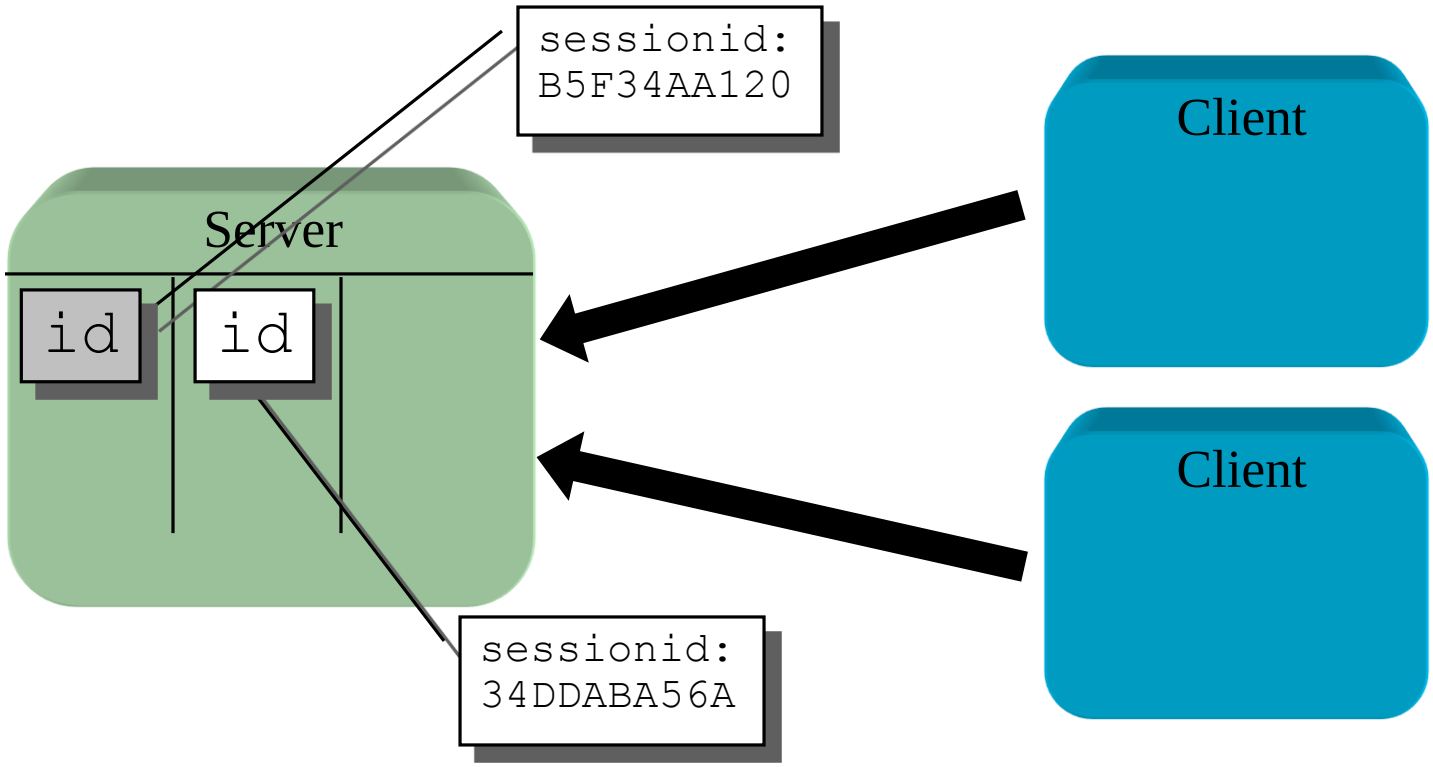
How does the sessions mechanism works:

The client sends a primary request to the server.



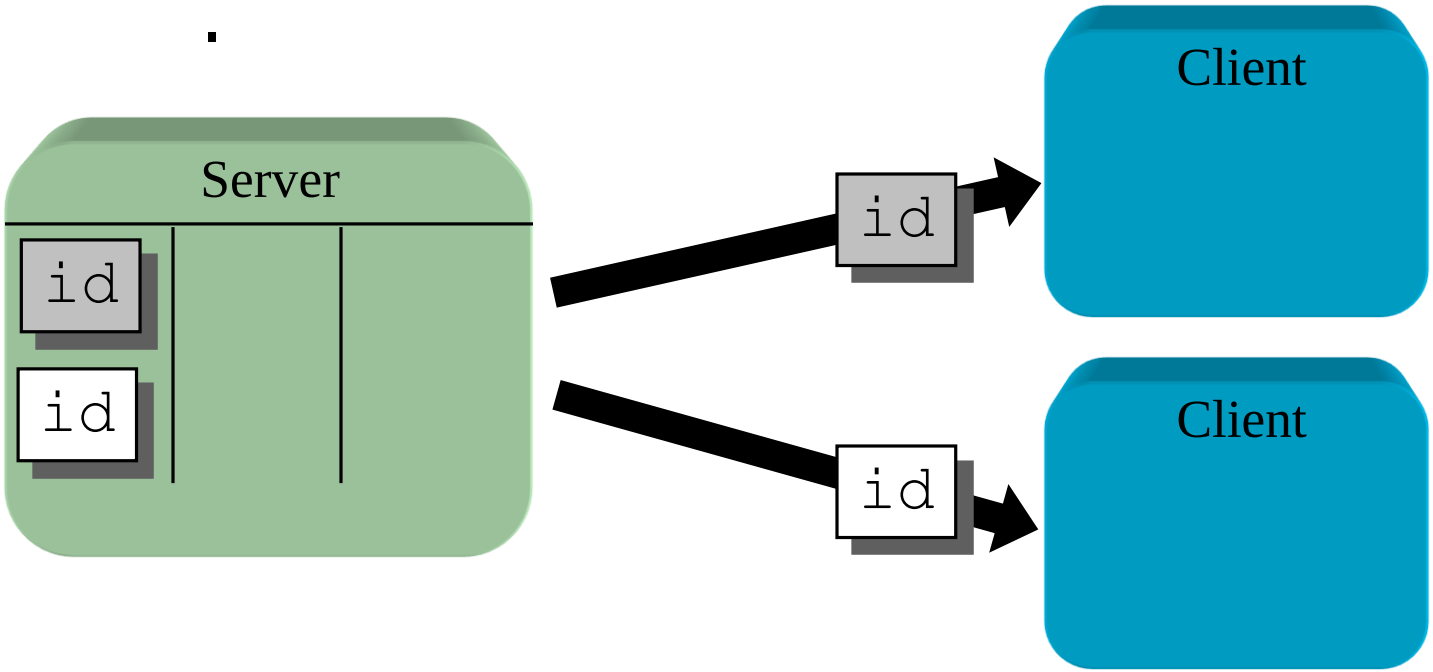
Session Management

2. Each connection is assigned a unique ID (usually called ***session ID***) when the session is created.



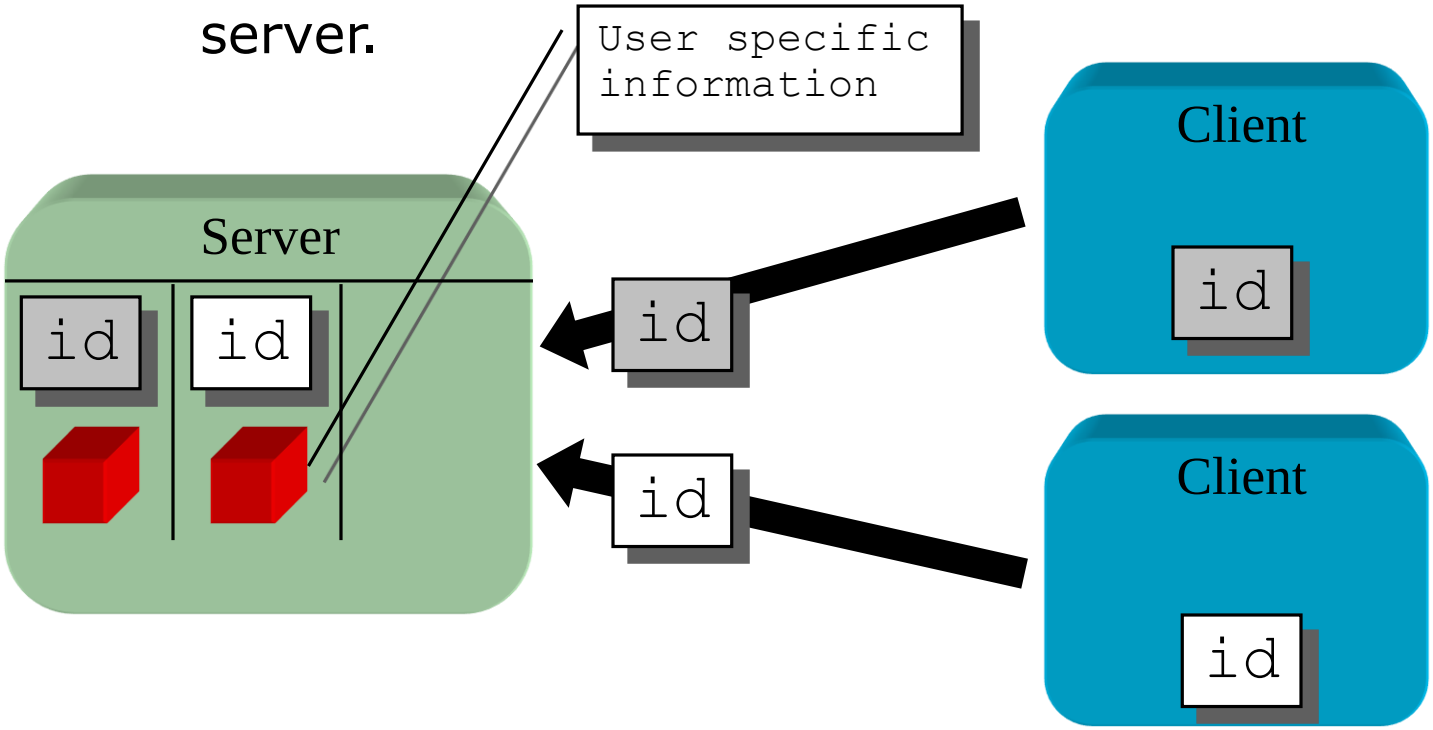
Session Management

3. The ID is transferred to the client, and passed back in all subsequent requests .



Session Management

4. The server can use the ID to track requests and associate user specific information stored on the server.



Session Tracking Using Cookies

- This is the most common technique.
- A cookie named "**PHPSESSID**" is created and sent to the client with the created session ID.
- The cookie is re-sent with every request and allows the server to track the session.
- This technique doesn't require any effort from the programmer.

Session Creation

- A session is being created for a client upon first request to a page with code:

```
session_start();
```

- The server assumes it is a new client if the request doesn't contain session ID information, and creates a new session for that client.

Session Use

- From the second request and on, the client becomes part of the existing session, as a long as the client has a valid session ID.
- To preserve a data in session use superglobal array $_SESSION:

```
session_start();
$_SESSION["user"]="Dana";
```

- The data will be accessible in any PHP page!!!

Session Destruction

- The answer to the question when to create a new session is quite straightforward – when the client enters the application.
- The problem is when to destroy it.
- In HTTP, there is no explicit termination indicator to the time in which the client in no longer active.

Session Destruction

Two methods of session destruction:

1. Explicit destruction in the code.
2. Destruction using a timeout.

Session Destruction

Explicit destruction in the code:

```
session_start();
session_unset(); // unset $_SESSION variable for the run-time
session_destroy(); // destroy session data in storage
```

Session Destruction

Destruction using a timeout:

```
 if (isset($_SESSION['LAST_ACTIVITY']) &&
(time() - $_SESSION['LAST_ACTIVITY'] > 1800)) {
    // last request was more than 30 minutes ago
    session_unset();
    session_destroy();
}
$_SESSION['LAST_ACTIVITY'] = time();
    // update last activity time stamp
```

Key Points

- Cookies and Session are used to keep state information.
- Cookies are simple, basic and limited, but keep persistent state.
- Sessions overcome cookie problems.
- Session management is used with all web based applications.