Forms: client-side

```
<html>
    <head>
        <title> A Form Example </title>
    </head><body>
<form action="welcome.php" method="post">
Name: <br /> <input type="text" name="name" /><br />
Phone Number: <br /> <input type="text" name="phone" /><br />
<input type="submit" value="Submit">
</form>
</body>
</html>
```

- form action – where to send the form data
- method – how to send the data (GET or POST)
- Name attributes become the keys used to access the corresponding fields in the $_GET or $_POST arrays

Forms: server-side

```
<html>
<head><title>This is welcome.php</title></head>
<body>
The name that was submitted was: &nbsp;
<?php echo $_POST['name']; ?><br />
The phone number that was submitted was: &nbsp;
<?php echo $_POST['phone']; ?><br />
</body>
</html>
```

- A simple PHP file that displays what was entered into the form
  - Can do many other things server-side depending on the situation
- Note the use of $_POST

PHP Cookies

- Cookies are stored on the user's browser, and are sent to the server on every relevant request
- The $_COOKIE superglobal makes a cookie a key-value pairing
  - Store user information as a value with a known key
  - Never assume a cookie has been set. Always check with `isset($_COOKIE[$cookie_name])` before trying to use the cookie's value

The setcookie() function

- To set a cookie in PHP:

  ```
  setcookie(name, value, expire, path, domain);
  ```

- Name and value correspond to `$_COOKIE[$name] = $value`
- Expiration – cookie will no longer be read after the expiration
  - Useful to use time in seconds relative to the present:
    - `time()` + time in seconds until expiration
- Path and domain refer to where on the site the cookie is valid
  - Usually '/' for path and the top-level domain (yoursitename.com)
- To delete a cookie, set a new cookie with same arguments but expiration in the past

Setting cookies

- Cookies are set via the HTTP header
    - Must be sent before the body – before any HTML, CSS, JS, etc.
- This code will not work:

```
if(isset($_COOKIE["6470"])) {
       $value  =  $_COOKIE['6470'];
       echo "Cookie is set to $value";
}
else {
       $value = 0;
}
// after echo statement: will not work!
setcookie("6470", $value+1, time()+60*60);?>
```

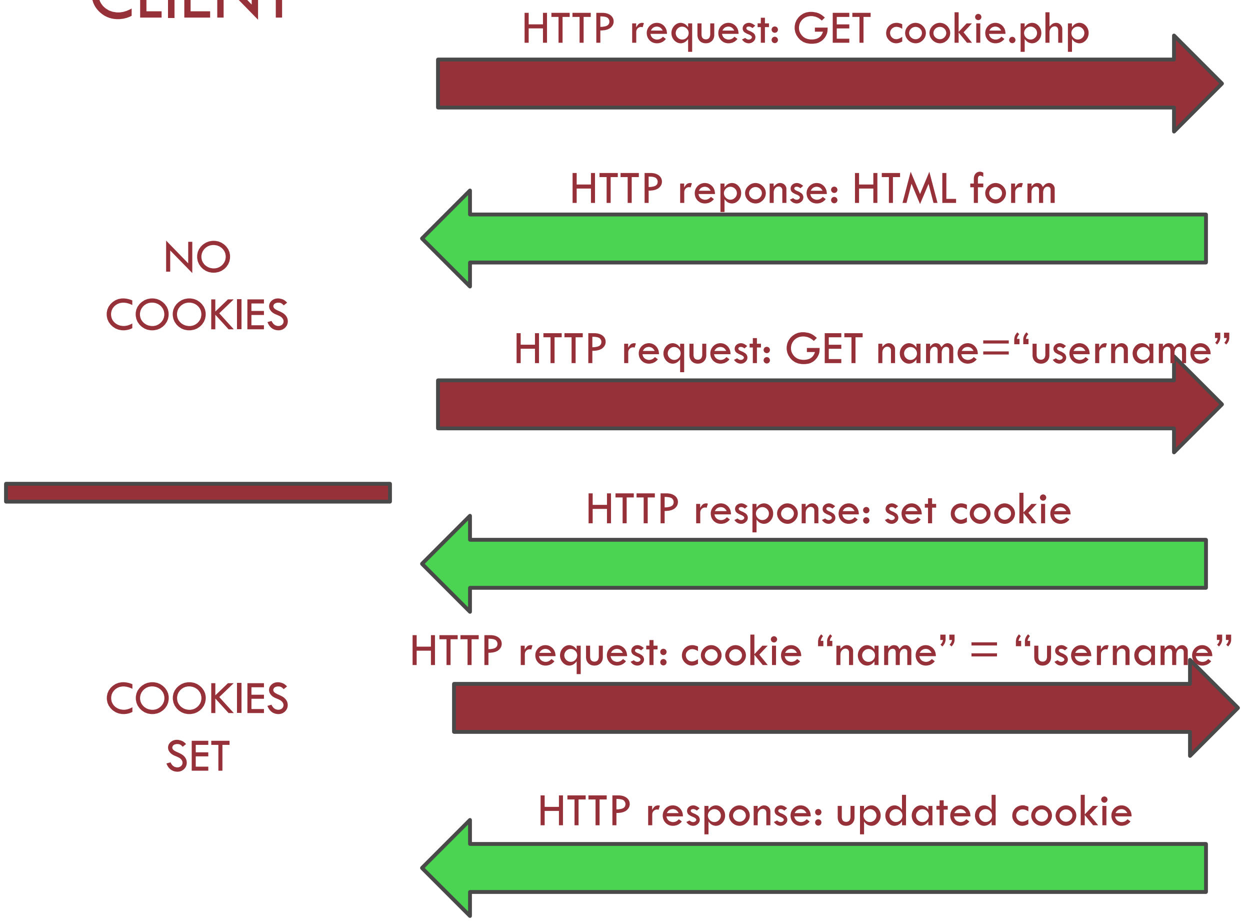
Example of cookie usage

- First visit: form with a text field for user's name
- Subsequent visits: Welcome message with the name
- Store the name field in a cookie:
  - Key: "name"; value: the user's name input into the form
- Remember: when a cookie is set (the setcookie function call is made), the cookie can only be accessed on the **next** request

Contents of the HTTP request/response

CLIENT

SERVER



HTTP request: GET cookie.php

HTTP reponse: HTML form

NO COOKIES

HTTP request: GET name="username"

HTTP response: set cookie

HTTP request: cookie "name" = "username"

COOKIES SET

HTTP response: updated cookie

isset($_COOKIE["name"])? NO
isset($_GET["name"])? NO
respond with HTML form

isset($_COOKIE["name"])? NO
isset($_GET["name"])? YES
set cookie on client
welcome message based on user input

isset($_COOKIE["name"])? YES
isset($_GET["name"])? NO
update cookie on client
welcome message based on cookie

Case 1: cookies already set

```
if(isset($_COOKIE["name"])) {
        $cookie_exp = time()+60*60; // one hour
        $name = $_COOKIE["name"];
        setcookie("name", $name, $cookie_exp);
        if (isset($_COOKIE["visits"])) {
                $num_visits = $_COOKIE["visits"]+1;
                setcookie("visits", $num_visits, $cookie_exp);
        }
        echo "Welcome $name! ";
        if (isset($_COOKIE["visits"])) {
                echo "You've visited $num_visits times";
        }
}
```

Cases 2&3: first and second visits

```
// case 2: upon submission of form
else if (isset($_GET["name"])) {
    $name = $_GET["name"];
    setcookie("name", $name, $cookie_exp);
    setcookie("visits", 2, $cookie_exp);
    echo "Welcome $name! This is your second visit.";
}

// case 3: first visit: need to show form
else {
    <form action="<?php $_SERVER["PHP_SELF"] ?>" method="get">
    Enter your name here: <input type="text" name="name" />
    <br /><input type="submit" />
    </form>
}
```

A registration-login example

- Login page
  - Check username and password
  - If already logged in (use sessions!), welcome the user by name
  - Link to register page
- Register page
  - Form for registration
  - If registration is successful, confirm the username
  - Link back to login page

A shared database resource

- Both login and register pages use the same database connection
- Put database connection, select database code into the same file
- Reference the connection resource ($db) in other files

```
<?php
$db = mysql_connect("localhost", "6470user", "6470") or
        die(mysql_error());
mysql_query("CREATE DATABASE IF NOT EXISTS 6470example") or
        die(mysql_error());
mysql_select_db("6470example", $db) or die(mysql_error());
mysql_query("CREATE TABLE IF NOT EXISTS users (USERNAME
        VARCHAR(2000), PASSWORD VARCHAR(2000))") or
        die(mysql_error());
?>
```

The login page – handle login request

```
if (isset($_POST["username"]) && isset($_POST["password"])) {
    require("db.php"); // establish DB connection
    $user = $_POST["username"];
    $pass = $_POST["password"];
    $query = "SELECT PASSWORD from users WHERE USERNAME='" .
        mysql_real_escape_string($user) . "'";
    $result = mysql_query($query, $db) or die(mysql_error());
    $row = mysql_fetch_assoc($result);
    if ($pass == $row["PASSWORD"]) {
        $_SESSION["username"] = $user;
    }
    else {
        echo "Invalid username or password <br />";
    }
}
```

The register page

```
if (isset($_POST["username"]) && isset($_POST["password"])) {
    require("db.php");
    $user = mysql_real_escape_string($_POST["username"]);
    $pass = mysql_real_escape_string($_POST["password"]);
    $query = "INSERT INTO users VALUES ('$user', '$pass')";
    mysql_query($query, $db) or die(mysql_error());
    echo "Registration for $user was successful <br /><br />";
    // HTML login <a href> tag
} else {
    // HTML form
}
```

MySQL recap

- Connecting to database
  - $db= mysql_connect(location, username, password)
  - mysql_select_db(db_name, $db)
- Making a query
  - $result = mysql_query(query_string, $db)
- Getting results of query
  - while($row = mysql_fetch_assoc($result))
- Sanitizing user input
  - $username = mysql_real_escape_string($_POST[“username”])