Authentication and authorization

What is a Cookie?

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds **on the user's computer**.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With PHP, you can both create and retrieve cookie values.

Working with Cookies using PHP

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

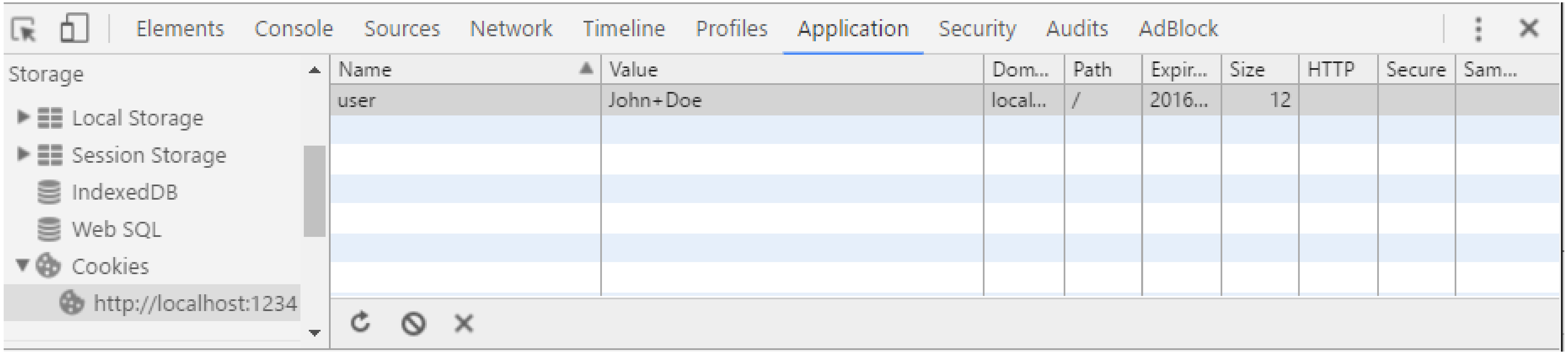
- name – the name of the cookie
- value - the value of the cookie. Do not store sensitive info!
- expire – the time the cookie expires (Unix timestamp – number of seconds since the epoch): time()+60*60*24*30 (30 days)
- path - The path on the server in which the cookie will be available on (‘/’ for the entire domain)
- domain - The (sub)domain that the cookie is available to
- secure – Boolean: Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client
- Httponly – When true the cookie will be made accessible only through the HTTP protocol

Create/Retrieve a Cookie

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is not set! <br />";
    echo "Value is " . $_COOKIE[$cookie_name];
}
?>
</body>
</html>
```

View cookies in browser (Chrome)

Press F12 -> Application -> Cookies



Name	Value	Dom...	Path	Expir...	Size	HTTP	Secure	Sam...
user	John+Doe	local...	/	2016...	12			

Modify a cookie

- Use setcookie with the same name ☺

Delete a cookie

- use the setcookie() function with an expiration date in the past

```php
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
```

Session

- Stored on the server
- Session variables last until the user closes the browser
- Session variables hold information about one single user, and are available to all pages in one application.

Start session

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Set session variables
$_SESSION["favcar"] = "Audi";
echo "Session variables are set.";
?>
</body>
</html>
```

Get session variable values

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
<?php
// Echo session variable that was set on previous page
echo "Favorite car is " . $_SESSION["favcar"];
?>
</body>
</html>
```

Modify session variable

```php
<?php
// To change a session variable just overwrite it
$_SESSION["favcar"] = "BMW";
print_r($_SESSION);
?>
```

Destroy session

```php
<?php
// Remove all session variables
session_unset();

// Destroy the session
session_destroy();
?>
```

Authentication

- User authenticates his identity
- How?
  - Password
  - Signature
  - Biometric data
  - External login (Facebook, Google)

Our implementation: Password authentication using cookies

Create DB Users table

```
CREATE TABLE USERS (
    user_id INT NOT NULL AUTO_INCREMENT,
    email VARCHAR(20) NOT NULL,
    password VARCHAR(20) NOT NULL,
    PRIMARY KEY (user_id),
    UNIQUE INDEX (email)
) ENGINE=INNODB;
```

Registration form

```
<form action="register.php" method="post">
  Email: <br />
  <input type="text" name="email" value="<?php echo $email;?>"><br />
  Password: <br />
  <input type="password" name="password"> <br />
  Confirm password: <br />
  <input type="password" name="confirm"> <br />
  <br />
  <input type="submit" value="Submit">
</form>
```

Validation – always validate!

- Email – valid mail
- Password – minimum 6 symbols, at least one uppercase, etc…
- Password & confirm password – are they equal?

Store passwords in db

- Never store them in pure text!
- HASH them!

```
$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

$isCorrect = password_verify($password, $hashedPassword);
```

Register user -> Insert into db

```
if (count($errors) == 0) {
    $conn = new PDO("mysql:host=localhost:3306;dbname=register", "root", "");
    $sql = "INSERT INTO Users (email, password) VALUES(?, ?)";
    $stmt = $conn->prepare($sql);
    $result = $stmt->execute([$email, password_hash($password)]);

    if($result) {
        echo '<h1>Successful registration!</h1>';
        echo '<a href="login.php">Go to Login</a>';
        die;
    }
    else {
        $error = $stmt->errorInfo();
        if ($error[1] == 1062) {
            array_push($errors,"Email taken");
        }
    }
}
```

Login form

```
<form action="login.php" method="post">
  Email:<br />
  <input type="text" name="email" value="<?php echo $email;?>"> <br />
  Password: <br />
  <input type="password" name="password"><br />
  <br />
  <input type="submit" value="Submit">
</form>
```

Authorization

```
if (count($errors) == 0) {
  $conn = new PDO("mysql:host=localhost:3306;dbname=register", "root", "");
  $sql = "SELECT * FROM Users WHERE email=? and password=?";
  $stmt = $conn->prepare($sql);
  $result = $stmt->execute([$email, password_hash($password, PASSWORD_DEFAULT)]);

  if($result && $stmt->rowCount() == 1) {
    setcookie('user', $email, time() + (86400 * 30), "/");
    header('Location: content.php');
    die;
  }
  else {
      array_push($errors,"Wrong email or password");
  }
}
```

Or even better

```
$sql = "SELECT password FROM Users WHERE email = ?";

$stmt = $conn->prepare($sql);

$result = $stmt->execute([$email]);

if (password_verify($password, $result['password'])) {

        session_start(); // password is valid
        $_SESSION['logged'] = true;
}
```

Simple authorization (not best, but interesting)

Check if the user is logged in

```
<?php
if(!isset($_COOKIE['user'])) {
    header('Location: login.php');
}
?>

<h1>Welcome <?php echo $_COOKIE['user'] ?>!</h1>
<a href="logout.php">Log out</a>
```

or:

```
if (!$_SESSION['logged']) {
    header('Location: login.php');
}
```

Log out

- Just remove the cookie!

```php
<?php
    setcookie('user', '', time() - 3600, '/');
    header('Location: login.php');
?>
```

Thank you!

Q&A