School of Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

Web Application Development

Selected Topics: Passing Control and Data + User Authentication

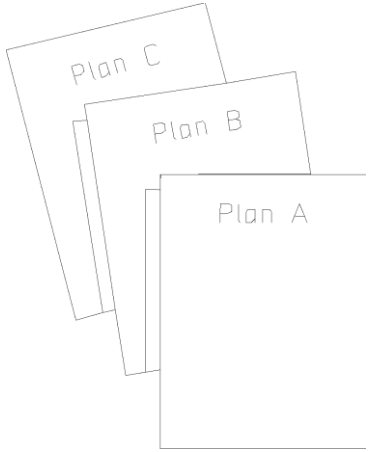
If everyone is moving forward together, then success takes care of itself.

Henry Ford

Overview

- Objective
  - Learn about some additional PHP concepts

- Content
  - Passing control across pages
  - Passing data across pages
  - Authenticating users



- After this module, you should be able to
  - Write code that pass control from a page to another
  - Write code that pass data from a page to another
  - Write code that authenticate users

I. Passing Control Across Pages

- Method 1: Form Submission

```
<html>
  <body>
    <form action='another.php'>
       <input type="submit“/>
    </form>
  </body>
</html>
```

first.php

- Method 2: Hyperlink

```
<html>
  <body>
    <a href='another.php'>Go to another page</a>
  </body>
</html>
```

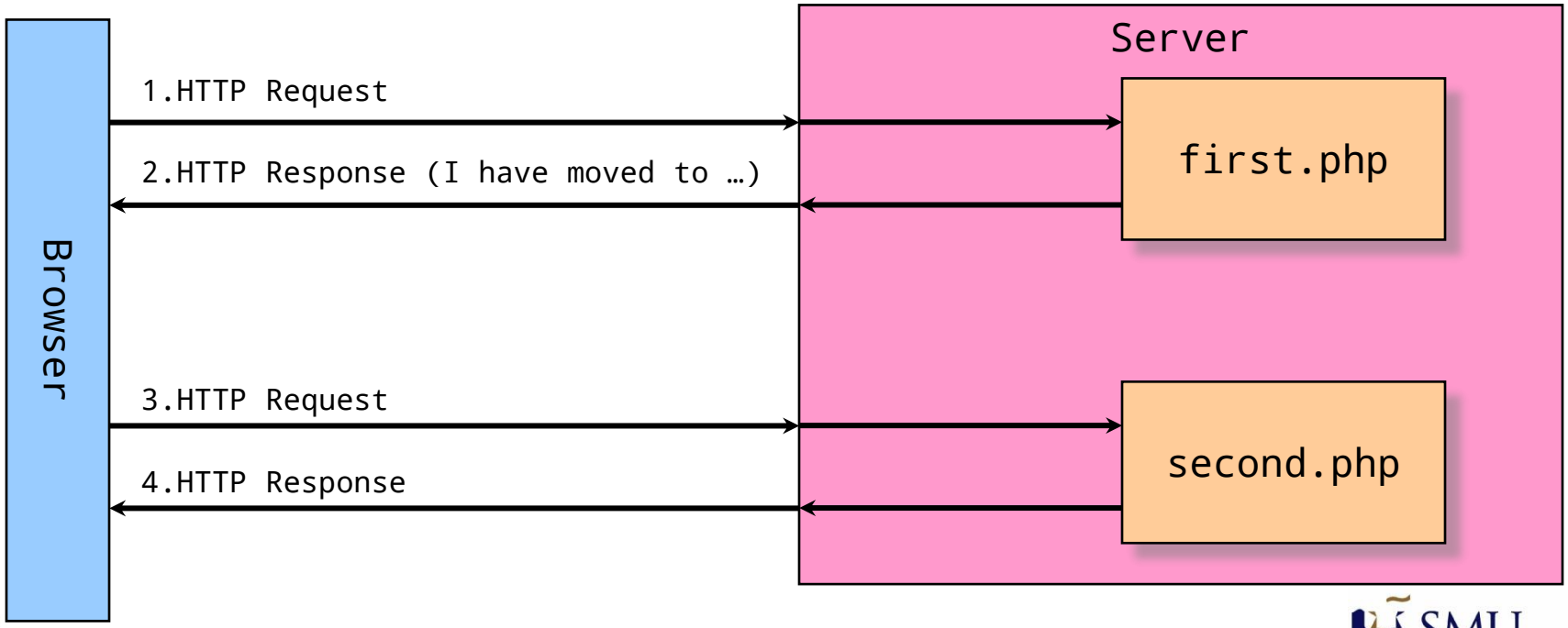
first.php

I. Passing Control Across Pages

- Method 3: **Automatic** page redirection

```
header("Location: second.php");
exit;
```

first.php

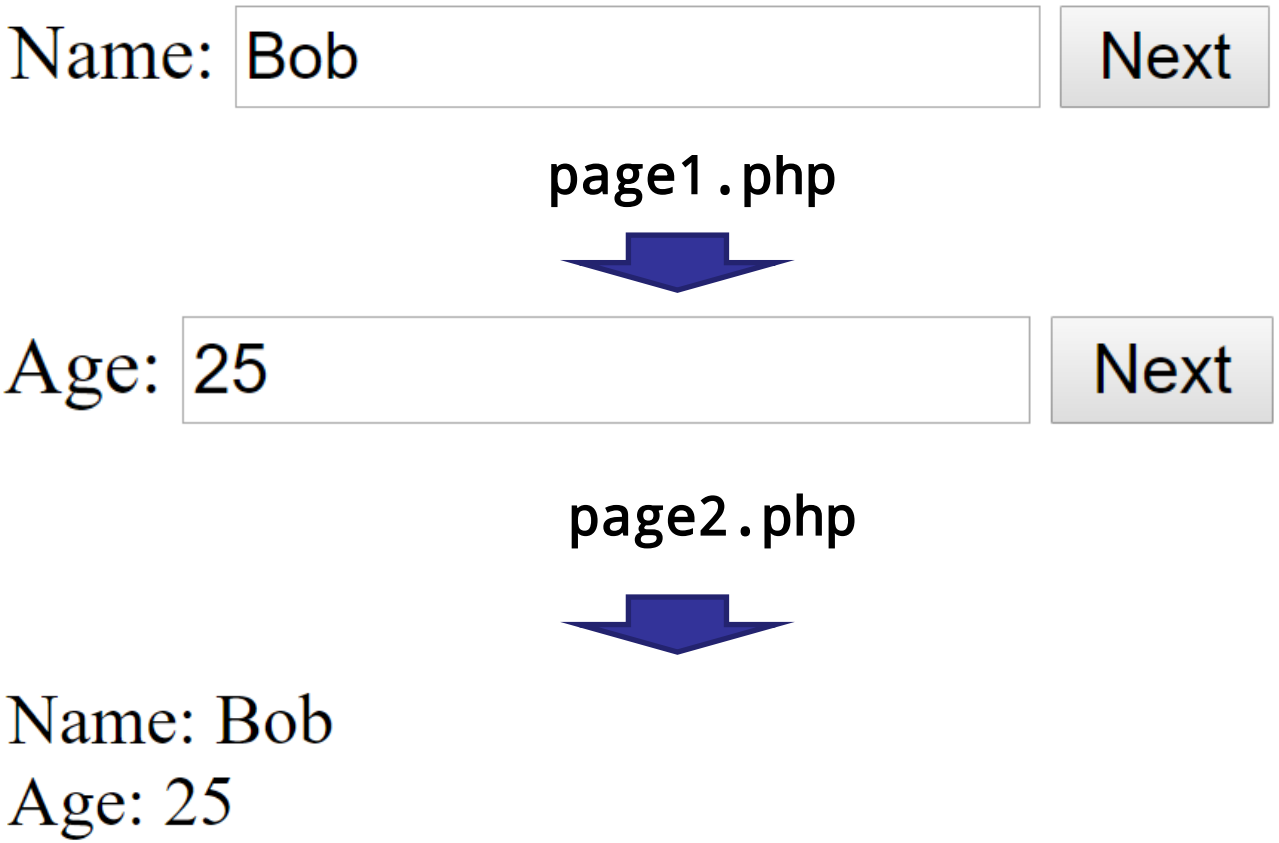


Exercise 1: Passing Control

- Modify the code in the previous slide so that first.php **redirects** to
  - http://www.google.com
- Put the code in your WampServer
- Check if it works

II. Passing Data Across Pages

- Method 1: Through form fields, including **hidden** fields



Hidden Fields

```
<html><body>
    <form method="post" action="page2.php">
        Name: <input type="text" name="name"/>
        <input type="submit" value="Next"/>
    </form>
</body></html>
```

page1.php

```
<html><body>
    <form method="post" action="summary.php">
        Age: <input type="text" name="age"/>
        <?php
            echo "<input type='hidden' name='name'
                        value='". $_POST['name'] . "'/>";
        ?>
        <input type="submit" value="Next"/>
    </form>
</body></html>
```

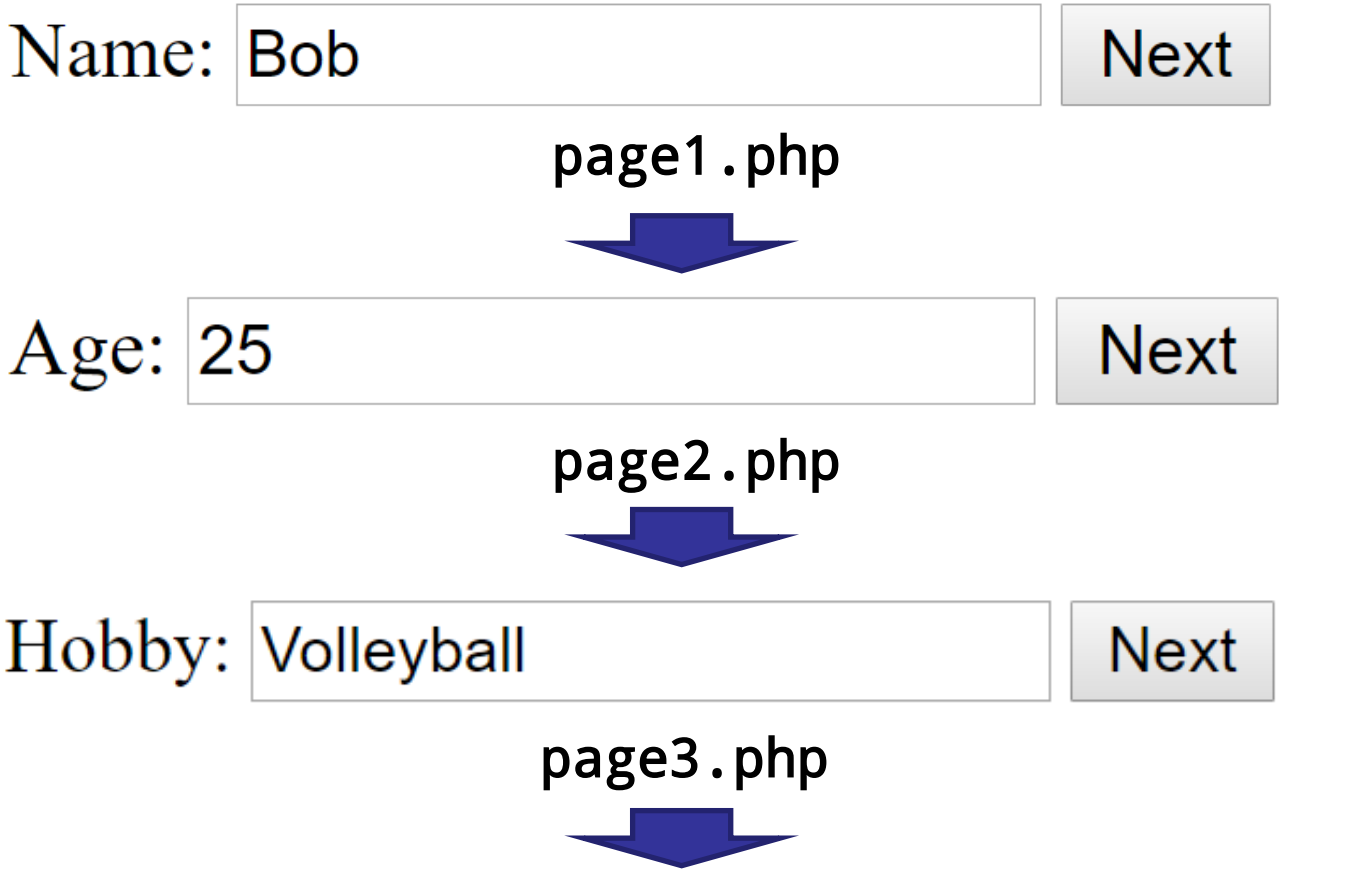
page2.php

Hidden Fields

```
<?php
    echo "Name: " . $_POST["name"];
    echo "<br>";
    echo "Age: " . $_POST["age"];
?>
```

summary.php

Exercise 2: Hidden Fields



II. Passing Data Across Pages

- Method 2: Through URL

```
<html>
  <body>
    <a href="view_object.php?src=a.jpeg&width=500">
    View Object
    </a>
  </body>
</html>
```

main.php

```
<?php
    echo "<img src=$_GET['src'] width=$_GET['width‘]/>”;
?>
```

view_object.php

II. Passing Data Across Pages

- Method 3: Using HTTP Session
- What is HTTP Session?
  - Stores data shared between a user and a website (e.g., eLearn)
  - Data available **across multiple pages** (e.g., IS112, IS113, and other pages in eLearn) or multiple instances of the same page
  - Data will automatically be reset after a period of time

II. Passing Data Across Pages

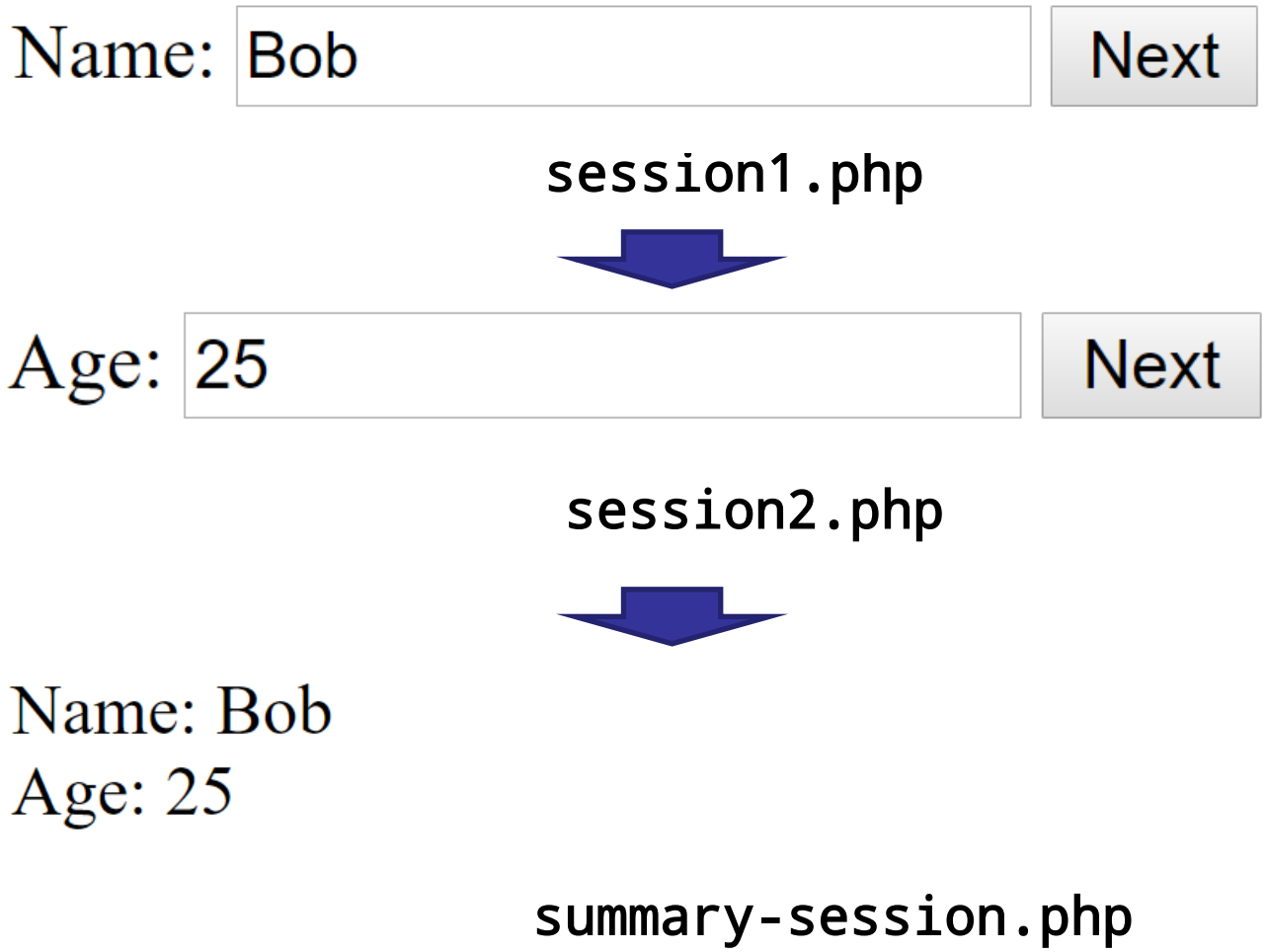
- Why we need it?
  - Identify a user across more than a page in a site
  - Pass data between web pages in the same site

HTTP Session

- How to use HTTP Session?
  - Call session_start()
    - Initialize a session OR
    - Resume an existing session
  - Use $_SESSION superglobal to add new key-value pairs into the HTTP Session

- **Note:** Make sure session_start() is called before accessing $_SESSION superglobals

HTTP Session



HTTP Session

```
<html><body>
    <form method="post" action="session2.php">
        Name: <input type="text" name="name"/>
        <input type="submit" value="Next"/>
    </form>
</body></html>
```

session1.php

```
<?php session_start(); ?>
<html><body>
    <form method="post" action="summary-session.php">
        Age: <input type="text" name="age"/>
        <?php
            $_SESSION['name'] = $_POST['name'];
        ?>
        <input type="submit" value="Next"/>
    </form>
</body></html>
```

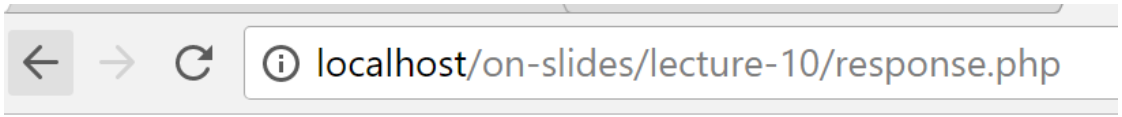
session2.php

HTTP Session

```
<?php
    session_start();
    echo "Name: " . $_SESSION["name"];
    echo "<br>";
    echo "Age: " . $_POST["age"];
?>
```

summary-session.php

HTTP Session: Another Example



You have accessed the page 1 times

You have accessed the page 2 times

HTTP Session: Another Example

```
<?php
session_start();
if(!isset($_SESSION["count"])){
    $_SESSION["count"] = 0;
}
$_SESSION["count"]++;
echo "You have accessed the page ".$_SESSION["count"]." times";
?>
```

response.php

HTTP Session: Another Example

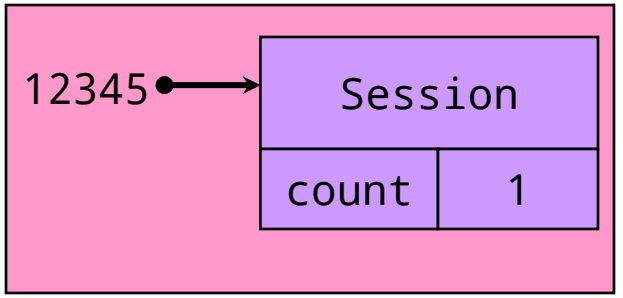
1. Client sends HTTP request

2. Server creates a session and generates a unique session id

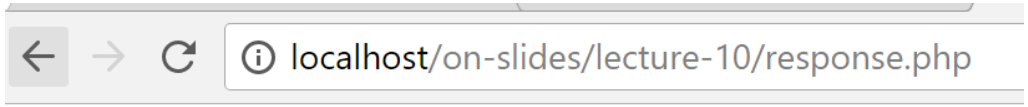
3. Session id is returned to the client

Client

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Date: Sun, 3 Jan 2018 11:02:15 GMT
Set-Cookie:PHPSESSID=12345;path=/
```



Server



You have accessed the page 1 times

HTTP Session: Another Example

5. Server retrieves the session and use value stored in the session during previous request.

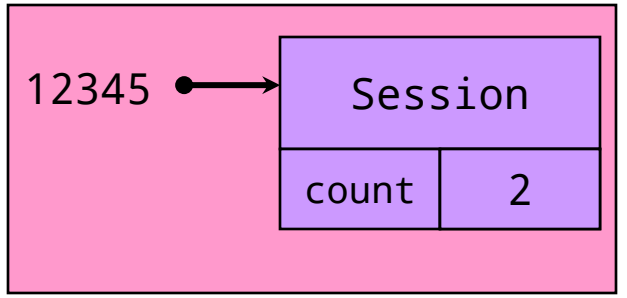
4. Client sends another HTTP request

```
GET / HTTP/1.1
Host: blue.smu.edu.sg
Cookie: PHPSESSID=12345
```

6. Increment count.

Client

6. Server sends response.



Server

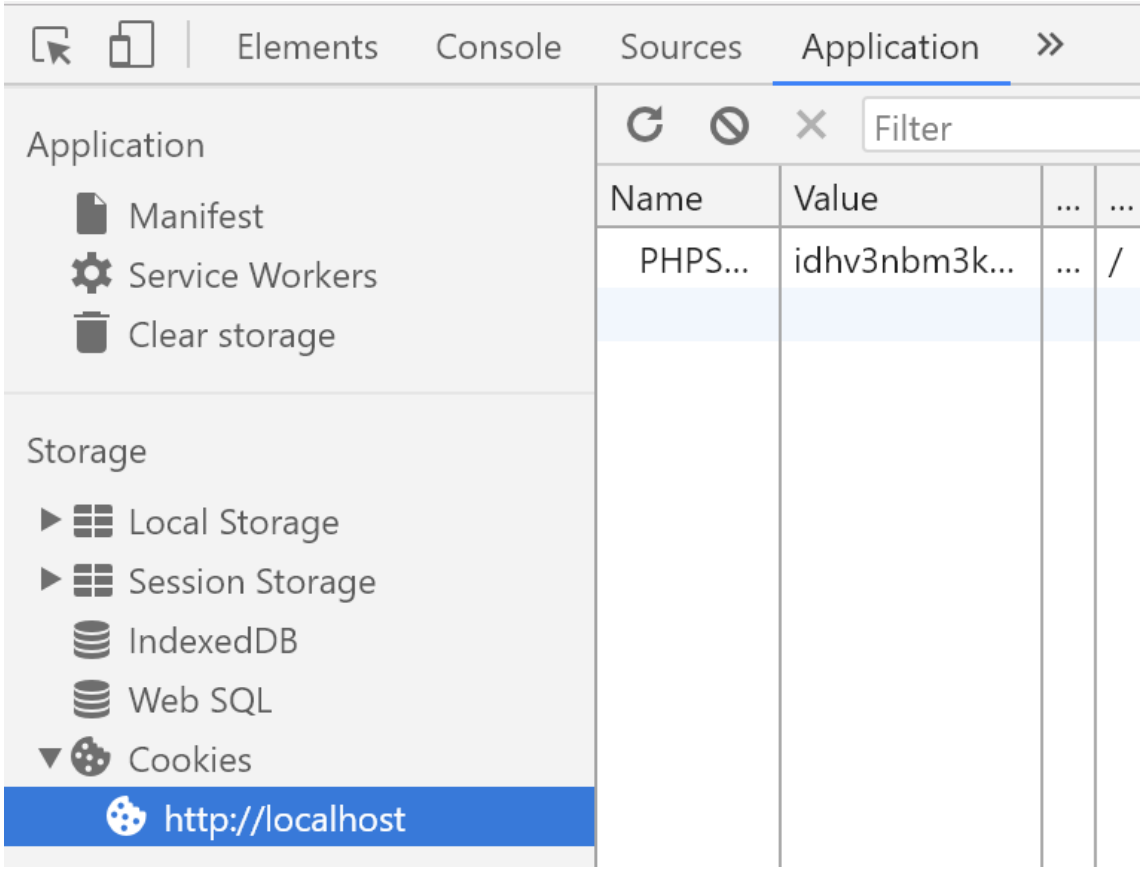
You have accessed the page 2 times

HTTP Session: Clearing Contents

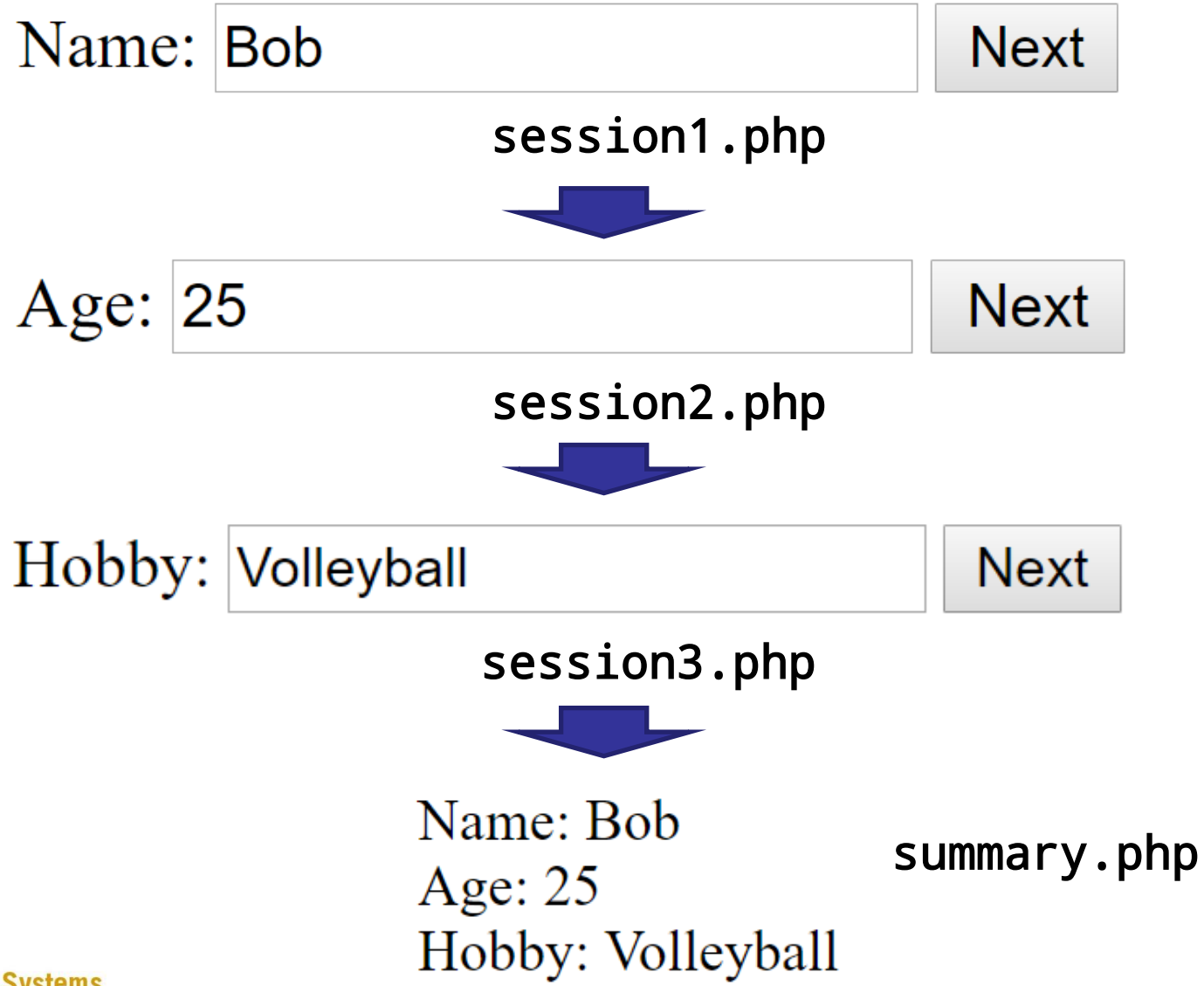
- Session would be cleared automatically after some period of time has lapsed
- What can we do to clear it earlier?
- On **server side** (i.e. PHP file):
  - We can set $_SESSION to an empty array, or
  - We can use *unset*($_SESSION[<key>]), e.g., *unset*($_SESSION["count"])

HTTP Session: Clearing Contents

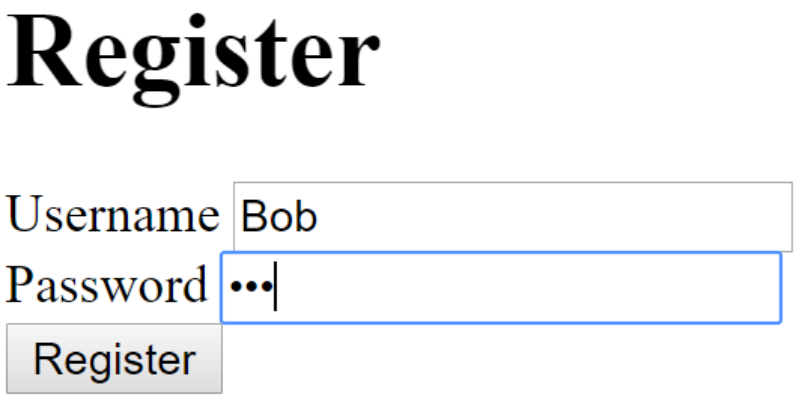
- On **client side** (i.e., web browser):
  - Session id can be forced to be cleared using, e.g., Chrome Dev Tools (Ctrl+Shift+I)

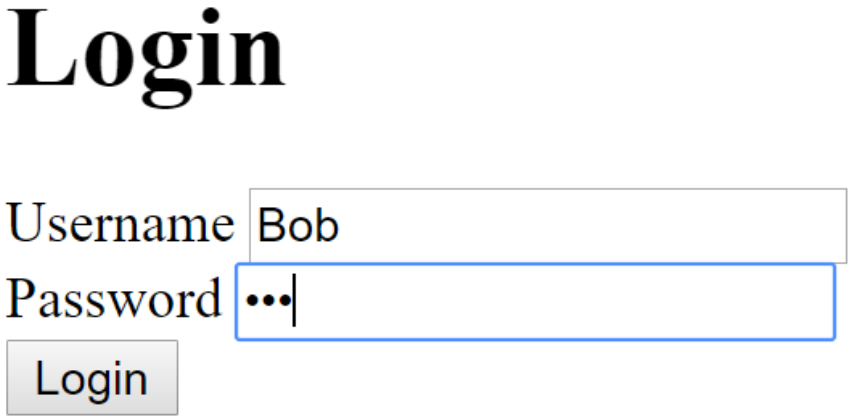


Exercise 3: Session

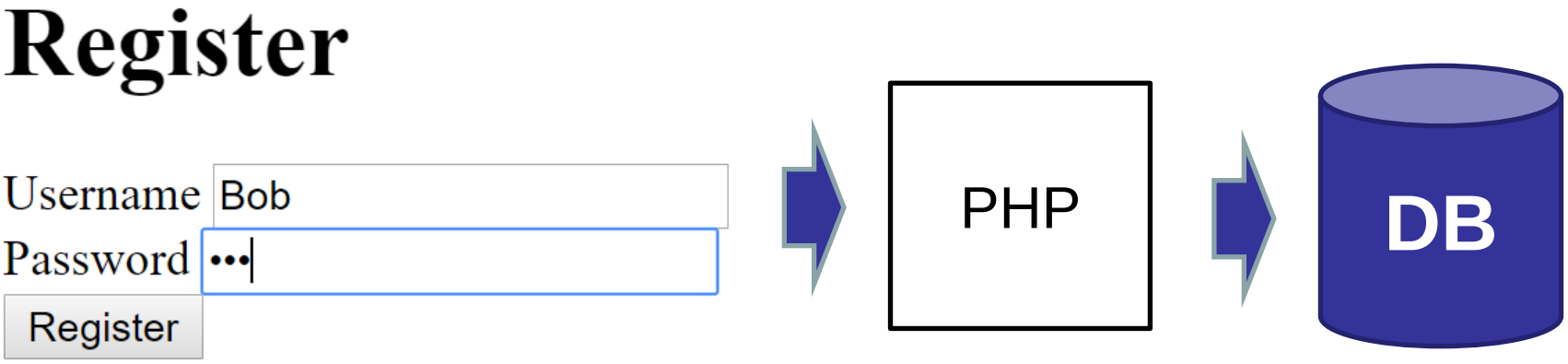


III. Authenticating Users





III. Authenticating Users



`register.php` `process_register.php`

- The user creates an account and his/her password is stored in the database.
- For **security reasons**, we do not want to store plain text password in the database.

Solution: Password Hashing

- A **one-way** transformation on a password
  - Turn the password into another string
- Password can be transformed to its hashed string
  - But, **not the other way round**

- We want to store **hashed** password in the DB

Password Hashing

```
<html><body>
    <h1>Register</h1>
    <form method="post" action="process_register.php">
        Username <input type="text" name="username"/><br/>
        Password <input type="password" name="password"/><br/>
        <input type="submit" value="Register"/>
    </form>
</body></html>
```

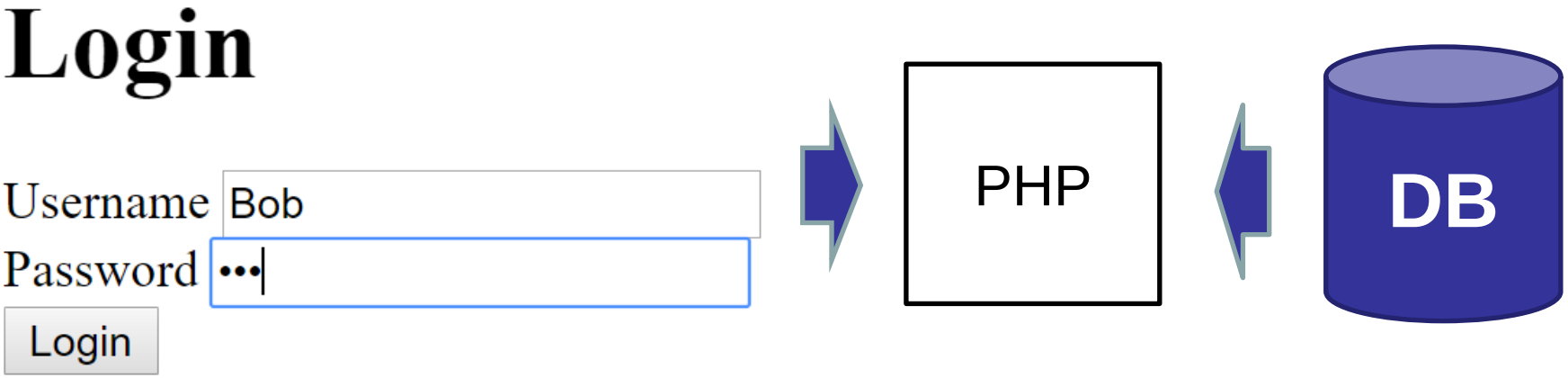
register.php

Password Hashing

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $hashed = password_hash($password, PASSWORD_DEFAULT);
    $dao = new UserDAO();
    $status = $dao->add($username,$hashed);
    if($status){
        echo "Registered successfully";
    }
    else{
        echo "Failed to register";
    }
?>
```

process-register.php

Login with Password Hashing



- When the user attempts to login:
  - Get the hash of the user's real password from DB
  - It will be checked against the entered password.

Password Hashing

```
<<html><body>
    <h1>Login</h1>
    <form method="post" action="process_login.php">
        Username <input type="text" name="username"/><br/>
        Password <input type="password" name="password"/><br/>
        <input type="submit" value="Login"/>
    </form>
</body></html>
```

login.php

Password Hashing

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $dao = new UserDAO();
    $hashed = $dao->getHashedPassword($username);
    $status = password_verify($password,$hashed);
    if($status){
        echo "Successful Login";
    }
    else{
        echo "Failed Login";
    }
?>
```

process-login.php

Using Session to Protect Your Pages

- Create a session entry for successful login

```
<?php
    require_once "UserDAO.php";
    $username = $_POST["username"];
    $password = $_POST["password"];
    $dao = new UserDAO();
    $hashed = $dao->getHashedPassword($username);
    $status = password_verify($password,$hashed);
    if($status){
        session_start();
        $_SESSION["user"] = $username;
        echo "Successful Login";
    }
    else{
        echo "Failed Login";
    }
?>
```

process-login.php

Using Session to Protect Your Pages

- For every page that needs to be protected

```
<?php

session_start();

// No session variable "user" => no login
if ( !isset($_SESSION["user"]) ) {

    // redirect to login page
    header("Location: login.php");

    // stop all further execution
    // (if there are statements below)
    exit;
}

?>
```

another_page.php

Using Session to Protect Your Pages

- To avoid repeating code everywhere:

```
<?php
session_start();
if ( !isset($_SESSION["user"]) ) {
    // No session variable "user" =>no login

    // redirect to login page
    header("Location: login.php");
    exit;
}
?>
```

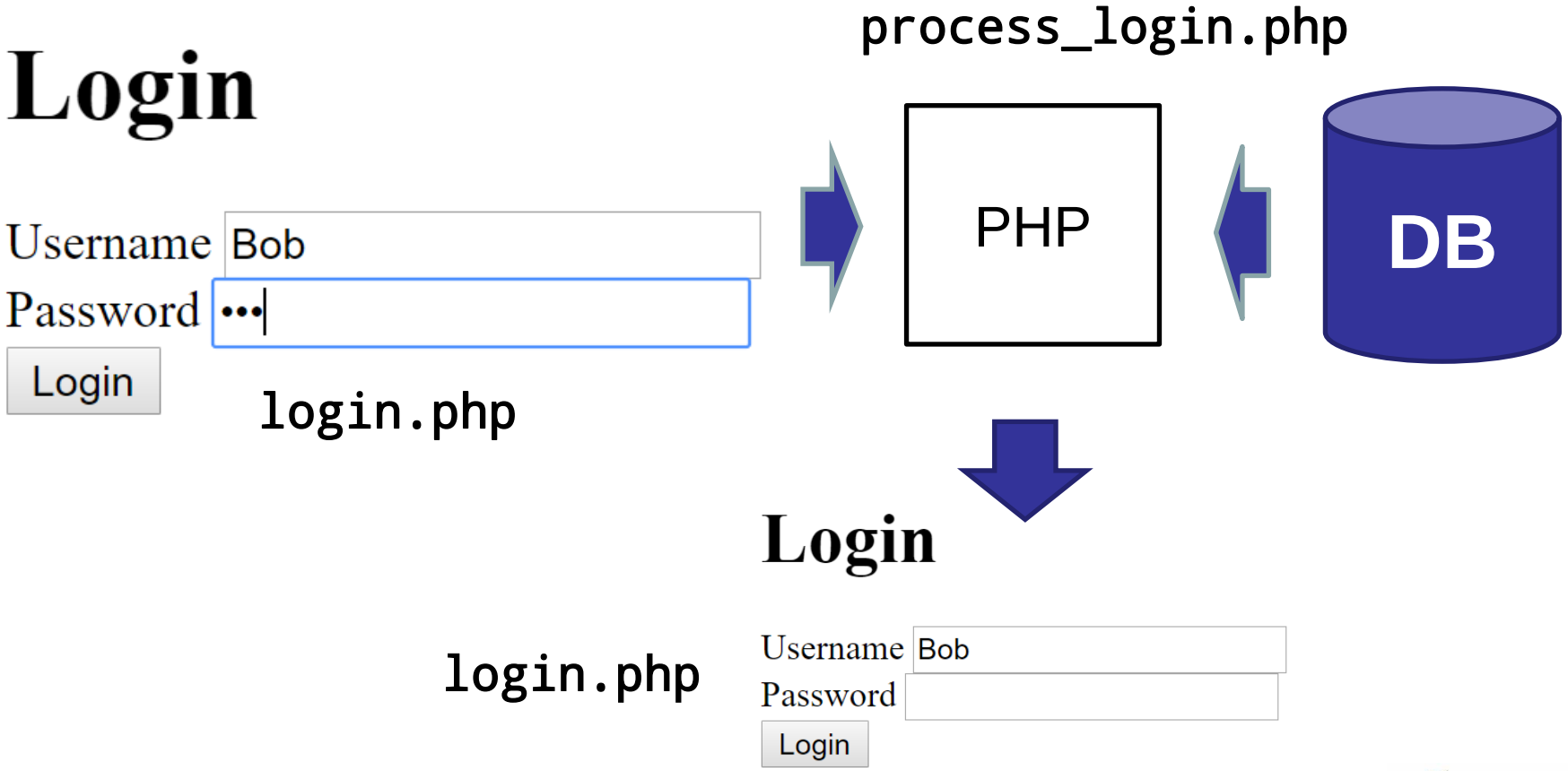
protect.php

```
<?php
require_once "protect.php";
// content below ..
?>
```

another_page.php

Exercise 4: Login

- Modify functionality of process_login.php when login fails



Exercise 4: Login

- Pass data from process_login.php to login.php using both HTTP GET and Session
  - Pass username using HTTP GET (i.e., through the URL)
  - Pass error message ("Failed Login") using Session

Key Points

- Passing Controls
  - Automatic page redirection

- Passing Data
  - Hidden fields
  - Through URL
  - Session
- Authenticating Users
  - Password hashing
  - Using session to protect pages