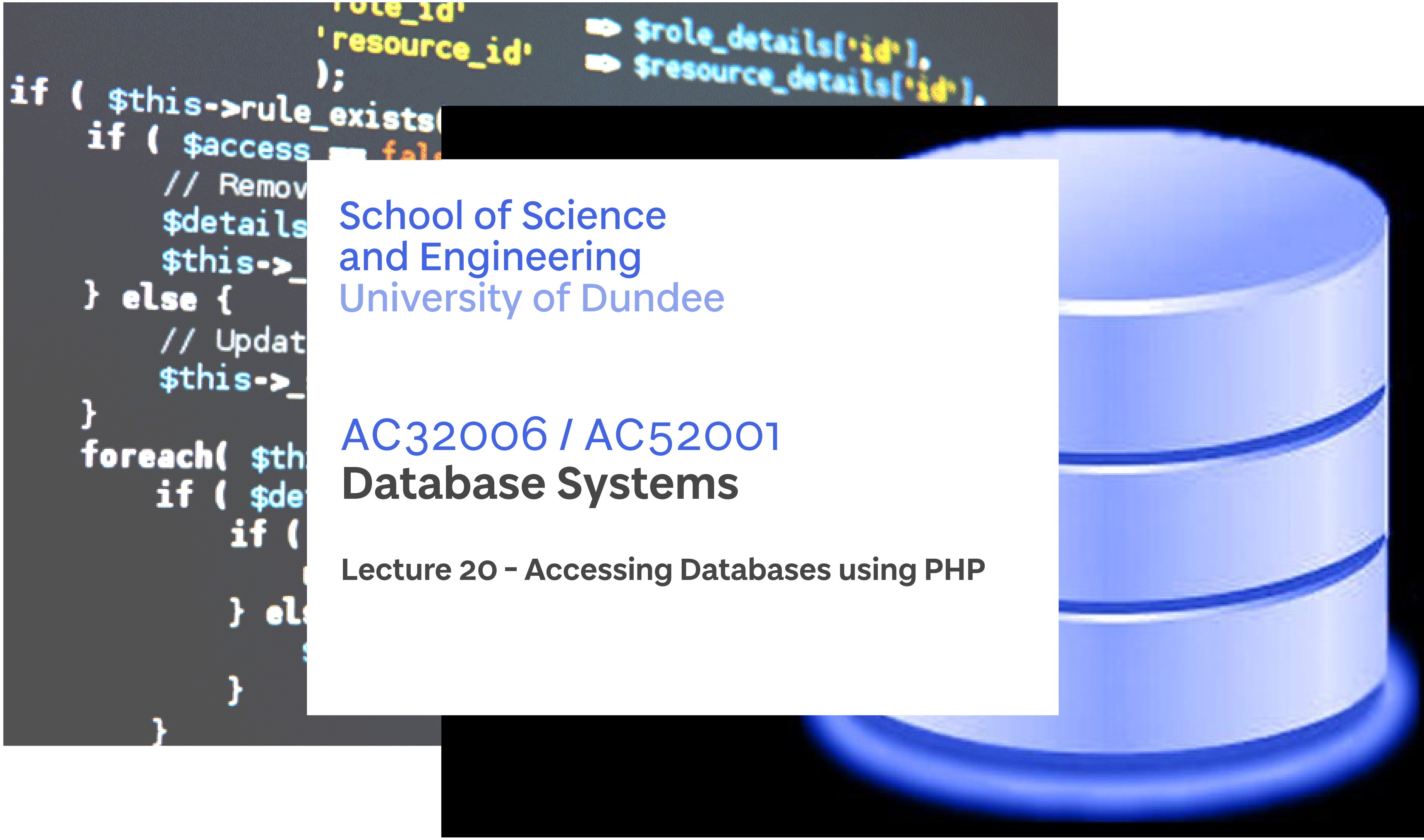


School of Science and Engineering

University of Dundee

AC32006 / AC52001

Database Systems

Lecture 20 - Accessing Databases using PHP

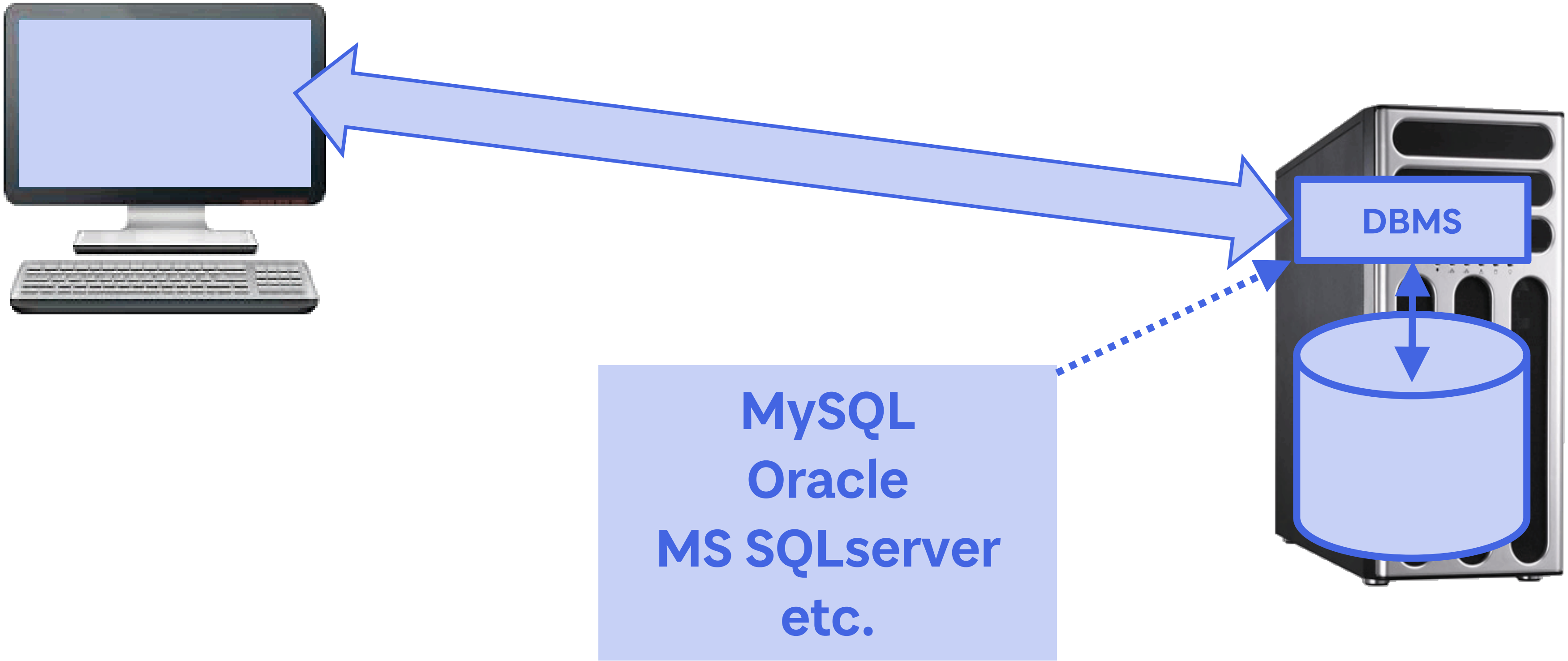
In today’s lecture

- → Remote access to databases
- → PHP - another programming language
- → How to connect a web interface to a MySQL database using PHP
- → Pointers for coursework #2

Thought for the Day

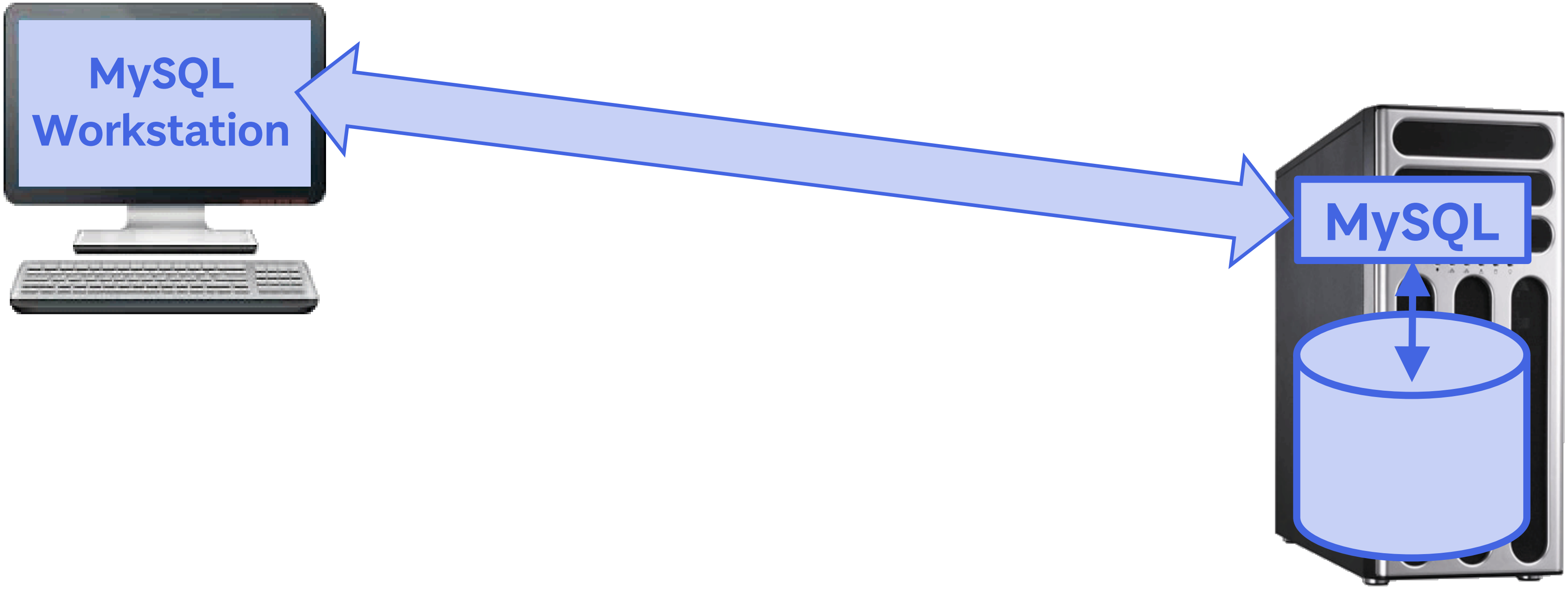
Introduction

Direct access to a database

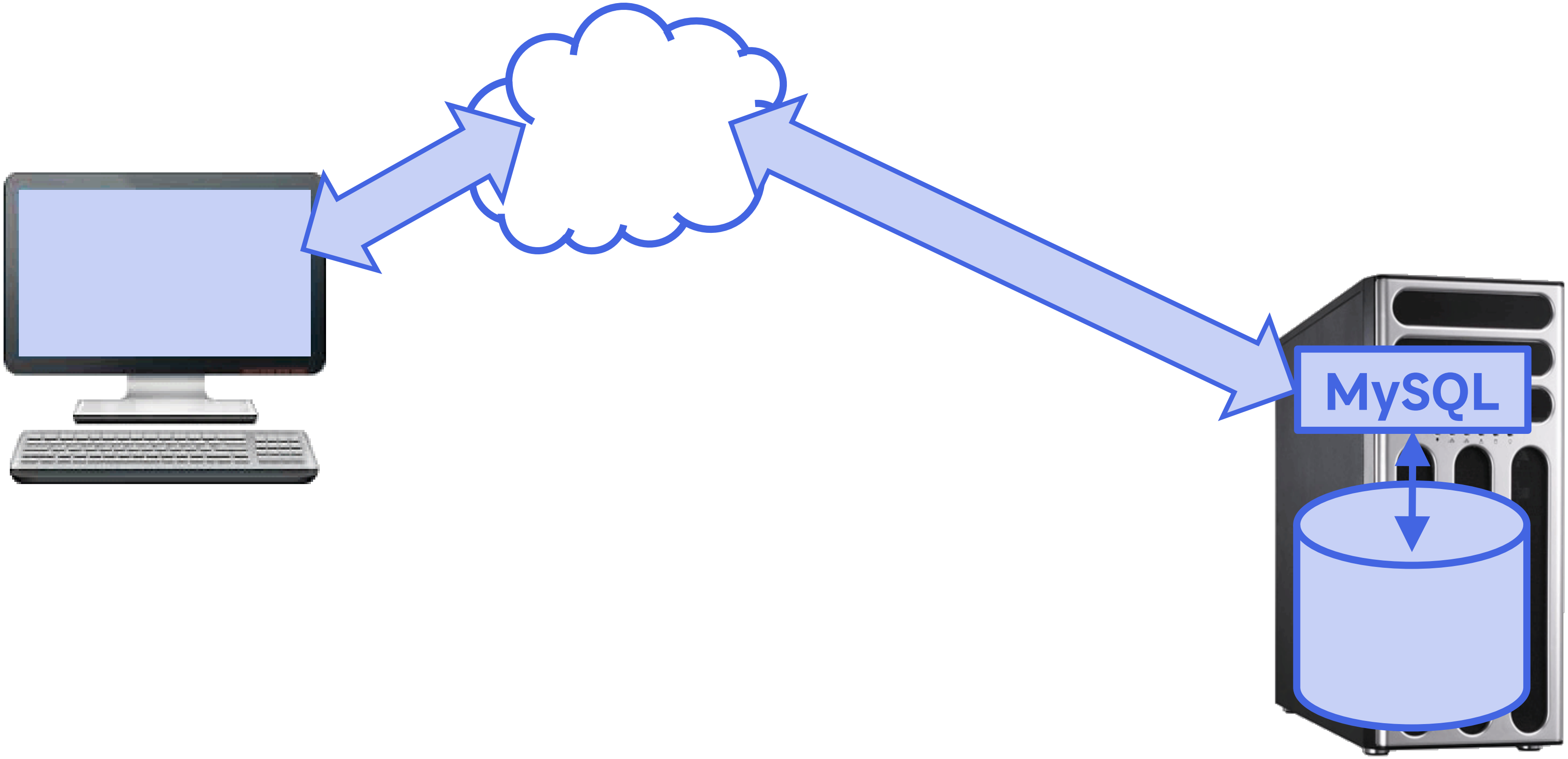


MySQL
Oracle
MS SQLserver
etc.

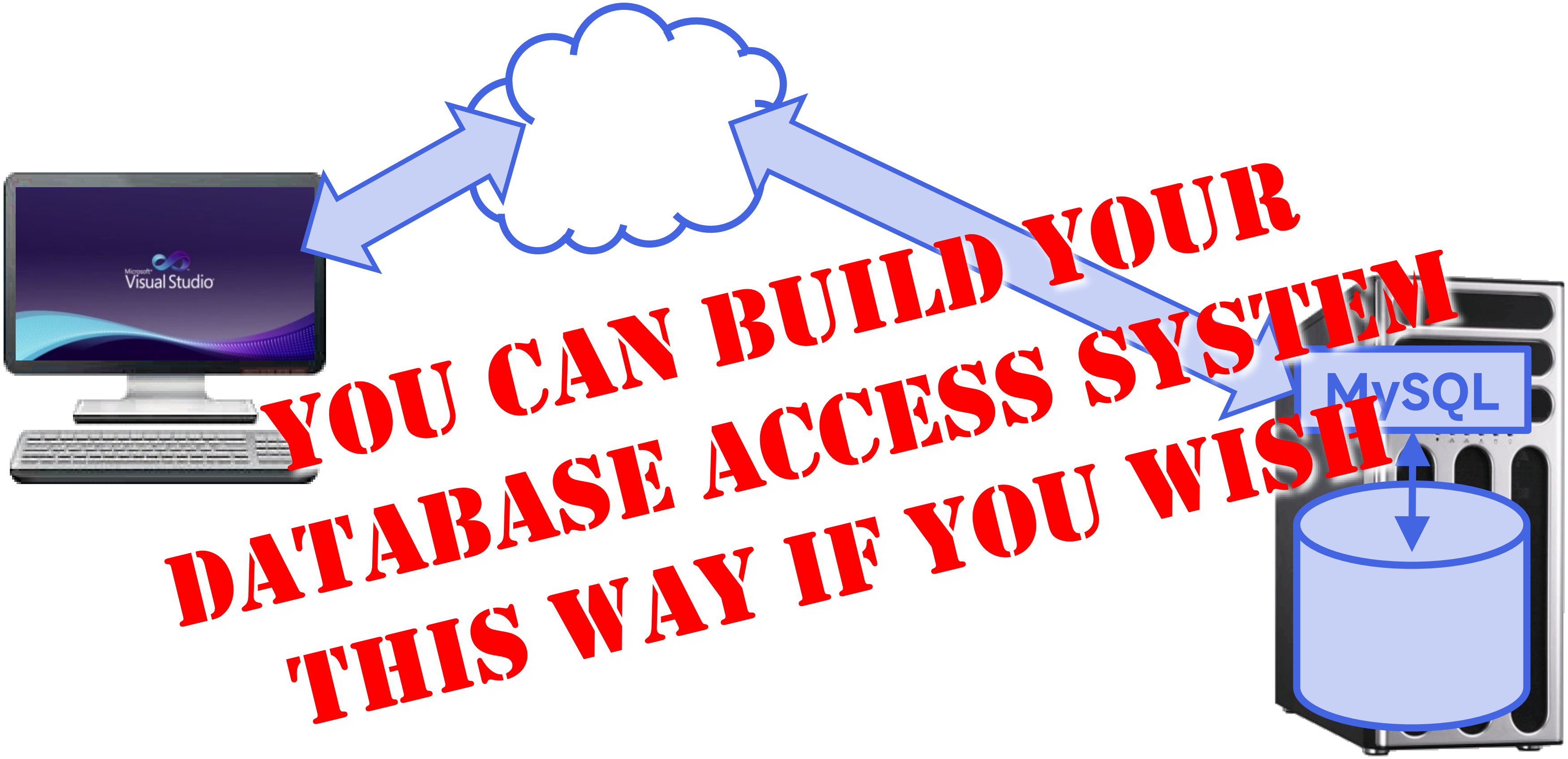
Direct access to *our* database



Remote access to *our* database

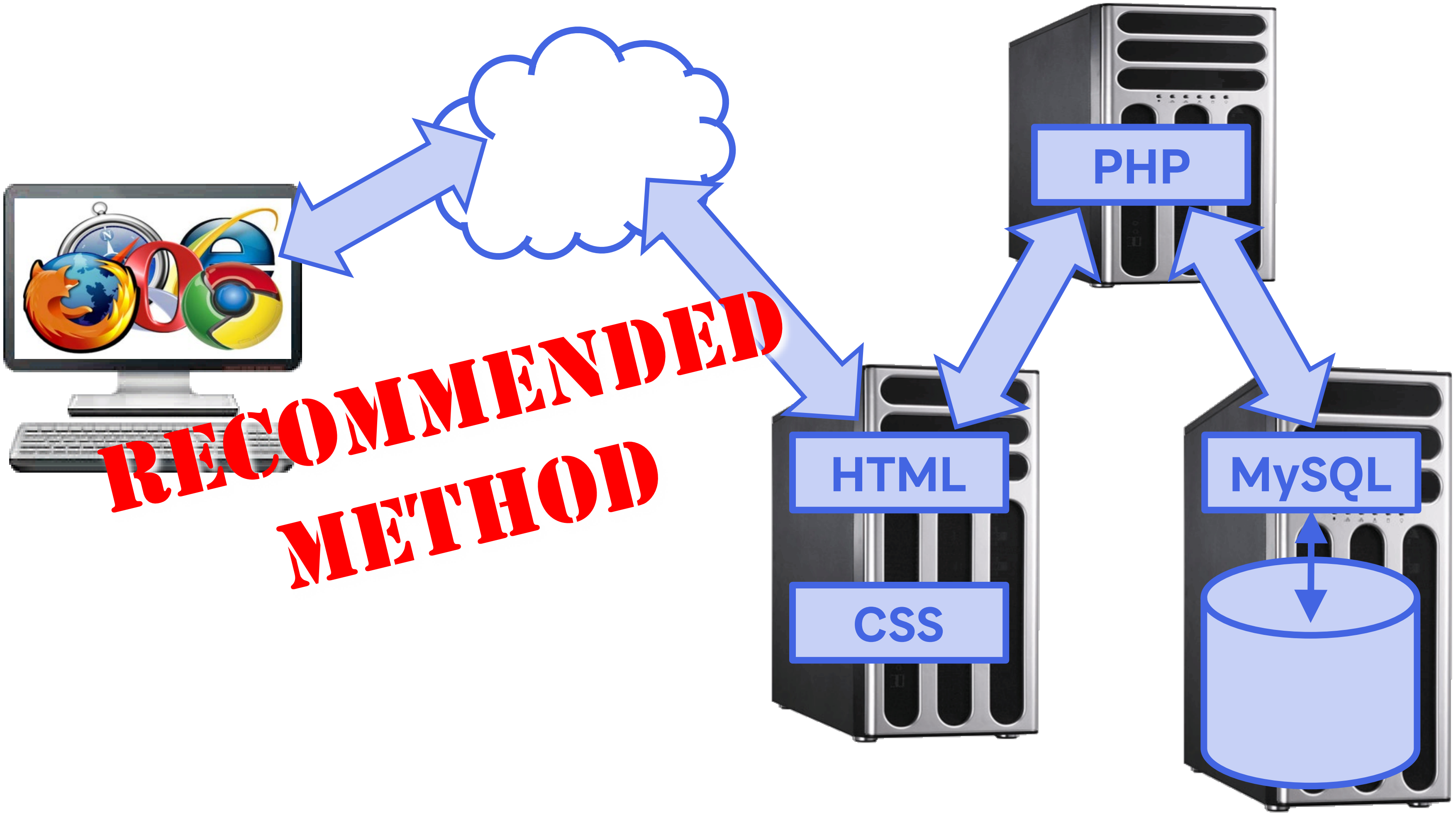


Standard Application



YOU CAN BUILD YOUR DATABASE ACCESS SYSTEM THIS WAY IF YOU WISH

Web Application



Three-Part Process

Server structure is in three parts, each with its own language(s):

- data access (e.g. MySQL):
  - *managed* access to the database
- functionality (e.g. PHP) :
  - extract data from forms
  - send data to MySQL and get reply
- presentation:
  - structure (including forms) for data entry - HTML
  - styling and layout for results display - CSS

Uncle Iain's PHP hints

- As the client only sees the *HTML output* from the PHP, this means the PHP itself is *hidden*:
  - *good* for security
  - *bad* for debugging
- If there are any errors in the PHP code, you may get
- However, MySQL errors *can* be trapped by PHP and displayed

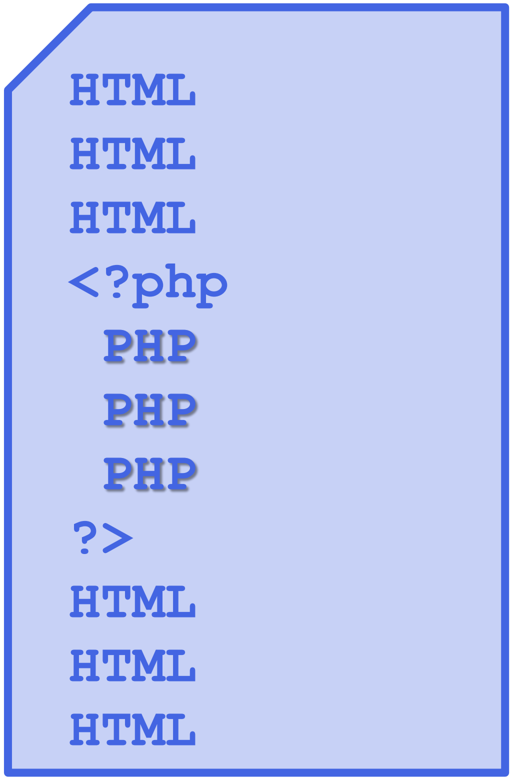
PHP

PHP

- PHP is a server-side scripting language
- PHP code is embedded into HTML files with extension `.PHP`
- PHP code is run on the server, and produces HTML which is served together with the rest of the HTML from the file containing the PHP
- syntax and data structures should be familiar to most
- libraries can be included
- code lines end in `;`

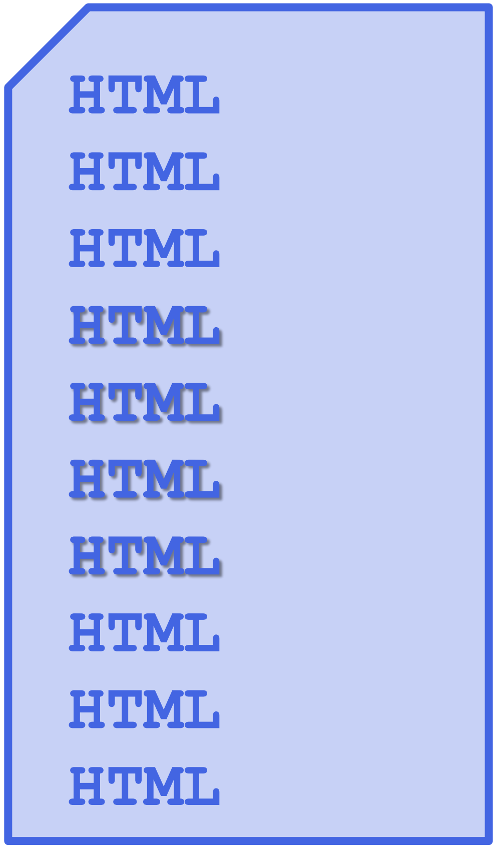
PHP

On the server:



```
HTML
HTML
HTML
<?php
  PHP
  PHP
  PHP
?>
HTML
HTML
HTML
```

The browser sees:



```
HTML
HTML
HTML
HTML
HTML
HTML
HTML
HTML
HTML
HTML
```

PHP - variables

- variables don’t have to be explicitly declared (recommended, though)
- has the usual data types (boolean, int, float)
- type is assumed by the value used
- variable names all start with a dollar:

```
$varNam
$a = 10
$b = 10.5
$c = "hello there"
```

PHP - strings

- → denoted by single or double quotes
  - → for quotes-within-quotes, use `\"`
- → concatenated using the dot operator `.`
- → new line as `\n` or HTML `<br>`
- → display using `echo`

```
echo "hello world<br>";
$i=12;
echo 'var i = '.i.'<br>';
```

PHP - conditions

→IF-ELSE statements used for conditions:

```
$error = true;
if ($error != true){
   //do some work
}
else
{
   echo "<h1>Something is
broken!</h1>";
}
```

PHP - loops

→FOR and WHILE loops:

```
for($i=0;$i<10;$i++)
{
    echo $i.'<br>';
}
$i=0;
while($i<10)
{
    echo $i.'<br>';
    $i++
}
```

PHP - versions

- PHP 5.1 added PHP Data Objects (PDOs) as a consistent interface for accessing databases in PHP
- a PDO is a data-access abstraction layer - whichever type of database is being used, the same functions are used to issue queries and fetch data
- You will be using PHP 7:
  - PHP 6 was never released
  - some older database functionality was deprecated and so does not work in PHP 7

Connecting to the Database

PHP - Connecting

- →We create a PDO with details of the database
- →Good practice to wrap in error trapping code

```
try{
    $mysql = new
    PDO("mysql:host=".$host.";dbname=".
    $database, $username, $password);
} catch (PDOException $e){
    // Could not connect
    echo "Database Connection Error!";
    var_dump($e);
    die();
}
```

your database *user*name

your database *password*

your database *name*

PHP - Using SQL

- Now we can use the MySQL **EXEC** command to send standard SQL queries to the database:

```
$mysql->exec('CREATE DATABASE new_db');
$mysql->exec('CREATE TABLE People
    (FirstName varchar(20),
     LastName varchar(20),
     id int,
     PRIMARY KEY (id)
     )
    ');
```

PHP - Inserting

→ Code which inserts values into SQL statements should always be prepared to prevent SQL injection attacks:

```
$stmt = $mysql->prepare('INSERT INTO People
(FirstName, LastName, id) VALUE (:FirstName,
:LastName, :id)');
$firstname = "John";
$lastname = "Smith";
$id = 321;
$stmt->bindParam(':FirstName', $firstname);
$stmt->bindParam(':LastName', $lastname);
$stmt->bindParam(':id', $id);
$stmt->execute();
```

PHP - Inserting from an HTML form

→We can take variables sent via the POST method from an HTML form:

```
$stmt = $mysql->prepare('INSERT INTO People (FirstName,
LastName, id) VALUE (:FirstName, :LastName, :id)');
if (isset($_POST['submit'])) {
    $stmt->bindParam(':FirstName', $_POST['firstname']);
    $stmt->bindParam(':LastName', $_POST['lastname']);
    $stmt->bindParam(':id', $_POST['id']);
    $stmt->execute();
    }
```

PHP - Queries

→Good practice to use prepare statement with queries too, especially those with a **WHERE** clause:

```
$stmt = $mysql->prepare('SELECT * FROM People');
$stmt->execute();
// Fetch each row
foreach ($stmt->fetchAll() as $result) {
    echo $result['FirstName'] . " " . $result['LastName'] .
           "<br>";   }

$stmt = $mysql->prepare('SELECT * FROM People WHERE
FirstName = :FirstName');
$firstname = 'Tim';
$stmt->bindParam(':FirstName', $firstname);
$stmt->execute();
```

PHP to MySQL

- →Create a *string variable* which holds the MySQL statement that you want to run (just as you would type the statement into MySQL Workbench)
- →Send the string to MySQL and *capture* the result
- →*Handle* the result as appropriate:
  - → error flag
  - → no error flag
- →… or create a stored procedure within your database and call it by name (more secure)

Web Forms

Access via forms

One standard way to build a web database interface is to:

→ use *web form* components to collect data from the user

→ use *PHP* to insert the collected data into *SQL queries* and send these to the database

→ use *HTML* and *CSS* to format the results returned by the PHP

Standard HTML Form

```
<html>
<form name="Add New Student"
action="form2.php" method="post">
Firstname:<input type="text"
name="firstname">
Surname: <input type="text"
name="surname"><br>
<input type="submit" value="Add to
database"/>
</form>
</html>
```

… where form2.php is …

```
$firstname = $_POST["firstname"];
$surname = $_POST["surname"];
$sqlInsert = "INSERT INTO Students
(Firstname, Surname) VALUES
('$firstname', '$surname')";
if (!mysql_query($sqlInsert))
{
die('Error: ' . mysql_error());
}
echo "<br>New person added";
```

Output

Just dump the data …

```
$query = "SELECT * FROM Students
WHERE Surname<='m' ORDER BY Surname
ASC";
$result = mysql_query($query, $db);
while($row =
mysql_fetch_array($result)){
 echo $row["Firstname"]." "
.$row["Surname"];
}
 echo "<br>Done.<br>";
```

… or add layout with HTML

```
$query = "SELECT * FROM Students ORDER BY Surname
DESC";

$result = mysql_query($query, $db);

echo "<br><table border=\"5\" bordercolor=\"black\"
cellpadding=\"10\" width=\"100%\" style=\"border-
collapse: collapse\" align=\"center\"><tr>";

while($row = mysql_fetch_array($result)){

      echo "<td>";

      echo
$row["Firstname"]."</td><td>".$row["Surname"];

      echo "</td></tr>";

}

      echo "</table>";

      echo "<br>Done.<br>";
```

Your Decisions

Database interface:

→web using PHP ★

→web using other methods

→Visual Studio

→other

User interface:

→bare HTML

→HTML+CSS created from scratch ★

→HTML+CSS via a 3rd party template ★

→application (VS or other)

Useful Resources

```
https://www.w3schools.com/sql
                         /php
                         /html
                         /css
```

PHP 7 Solutions: Dynamic Web Design Made Easy,
David Powers, 2019, Springer eBooks

PHP 7 Quick Scripting Reference,
Mikael Olsson, 2016, Springer eBooks

`https://www.phpjabbers.com/free-website-templates.php`

Summary

We have seen:
→PHP used as HTML / MySQL interface
→PHP basic syntax
→PHP accessing database via MySQL
→the basics of web forms for a web interface

Next meetings:
→Tuesday 0900 in QM Labs - PHP practical
→Tuesday 1600 in Dalhousie LT2

University of Dundee

dundee.ac.uk