

COS10011/60004

Creating Web Applications

Lecture 11

PHP and MySQL Part 2



Outline



Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records



Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- Selecting, Creating, Updating, and Deleting Records
- Handling errors



Accessing Databases with PHP

- There are three main options when considering connecting to a MySQL database server using PHP:
 - PHP's mysql Extension
 - PHP's mysqli Extension
 - PHP Data Objects (PDO)
- The mysqli extension features a dual interface, supporting both procedural (functions) and object-oriented interfaces.
- These notes and examples use the procedural interface.

We will use mysqli

<http://php.net/manual/en/mysqli.summary.php>

3 - Creating Web Applications, © Swinburne



Hint: Separate file for your login info



Example

`<?php`

`$host = "feenix-mariadb.swin.edu.au";`

`$user = "s1234567";`

`$pwd = "password";`

`$sql_db = " s1234567_db";`

`?>`

Edit the host name
when ported to a
production server

Your student id

Initially ddmmy.
Change, but don't
use your SIMs
password

ITS has created a
predefined
database for you

4 - Creating Web Applications, © Swinburne





Template 1 – for SQL* queries

- * Create and drop tables
- * Insert update and delete records

<?php

require_once "settings.php";

\$conn = @mysqli_connect (\$host,\$user,\$pwd,\$sql_db);

if (\$conn) {

Step 2: Create your SQL query

\$query = "replace with a valid SQL query";

\$result = mysqli_query (\$conn, \$query);

if (\$result) { ...}

else {...}

Step 4:
Did it
work?

Step 3: Execute your SQL query

mysqli_close (\$conn);

} else echo "<p>Unable to connect to the db.</p>";

?>

Step 5: Close connection

HUPD

5 - Creating Web Applications, © Swinburne



Connecting to MySQL



- Open a connection to a MySQL database server with the **mysqli_connect()** function
- The **mysqli_connect()** function returns **an object representing the connection** if it connects to the database successfully or **false** if it does not
- Assign the return value from the **mysqli_connect()** function to a variable that you can use to access the database in your script

- Example

```
$yourconn= mysqli_connect("feenix-mariadb.swin.edu.au",
"s1234567", "yourMySQLpassword", "s1234567_db");
```

6 - Creating Web Applications, © Swinburne



Connecting to MySQL (continued)



- The syntax for the `mysqli_connect()` function is:

```
$connection = mysqli_connect("host"  
[, "user", "password", "database"]) HUPD
```

- The **host** argument specifies the host name where your MySQL/MariaDB database server is installed

e.g. `feenix-mariadb.swin.edu.au`

- The **user** and **password** arguments specify a MySQL/MariaDB account name and password

e.g. `s1234567 yourMySQLpassword`

- The **database** argument specifies a database

e.g. `s1234567_db`

7 - Creating Web Applications, © Swinburne



Connecting and Selecting



- The `mysqli_connect` also allows one to connect and select the database in one step.

```
$dbConnect = mysqli_connect(  
"feenix-mariadb.swin.edu.au",  
"s1234567", "ddmmyy", "s1234567_db");
```

Your MySQL/MariaDB
password

8 - Creating Web Applications, © Swinburne



Selecting a Database



We can `connect()` and `select_db()` in separate steps

- The statement for selecting a database with the MySQL Monitor is **`use database;`**
- The function for selecting a database with PHP is **`mysqli_select_db(connection, database)`**
- The function returns a value of **`true`** if it successfully selects a database or **`false`** if it does not

Executing SQL Statements



Database and Table queries:

The **`mysqli_query()`** function returns one of three values:

- For SQL statements that *do not* return results (**`CREATE DATABASE`** and **`CREATE TABLE`** statements) they return a value of **`true`** if the statement executes successfully
- For SQL statements that *do* return results (**`SELECT`** and **`SHOW`** statements) they return a ***result pointer*** that represents the query results
 - A **result pointer** is a special type of variable that refers to the currently selected row in a resultset
- For SQL statements that fail, **`mysqli_query()`** function returns a value of **`false`**, regardless of whether they return results

Closing Connection



- Close a connection to a MySQL/MariaDB database server with the `mysqli_close()` function

```
mysqli_close($dbconnect);
```

Outline



Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

Accessing Databases with PHP



- **Creating and Deleting Databases and Tables**
- Selecting, Creating, Updating, and Deleting Records
- Handling errors

Creating Tables



- The `CREATE TABLE` statement specifies the table and column names and the data type for each column
- The syntax for the `CREATE TABLE` statement is:

```
CREATE TABLE table_name  
    (column_name TYPE, ...);
```
- Execute the `USE` statement to select a database before executing the `CREATE TABLE` statement

Creating and Deleting Tables (continued)



```
...  
$sqlString = "CREATE TABLE cars(  
    car_id    AUTO_INCREMENT PRIMARY KEY,  
    model     VARCHAR(30),  
    make      VARCHAR(25),  
    price     INT,  
    yom       DATE) ";  
$queryResult = @mysqli_query($dbConnect, $sqlString)  
...
```

Use INT if you do not want to store any decimal figures

add NOT NULL if field is required

Note: Usual to check to see if the table exists, and if not, create table.

Creating Tables (continued)



Type	Range	Storage
BOOL	-128 to 127 with 0 considered false	1 byte
INT or INTEGER	-2147483648 to -2147483647	4 bytes
FLOAT	-3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E+38 to 3.402823466E+38	8 bytes
DOUBLE	-1.7976931348623157E+308 to -2.2250738585072014E+308, 0, and 2.2250738585072014E+308 to 1.7976931348623157E+308	8 bytes
DATE	'1000-01-01' to '9999-12-31'	Varies
TIME	'-838:59:59' to '838:59:59'	Varies
CHAR(n)	Fixed length string between 0 to 255 characters	Number of bytes specified by n
VARCHAR(n)	Variable length string between 0 to 65,535 characters	Varies according to the number of bytes specified by n

Common MySQL field data types

15 - Creating Web Applications, © Swinburne



Deleting Tables



- The `DROP TABLE` statement removes all data and the table definition
- The syntax for the `DROP TABLE` statement is:

```
DROP TABLE table_name;
```

16 - Creating Web Applications, © Swinburne



Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- • **Selecting, Creating, Updating, and Deleting Records**
- Handling errors

Structured Query Language (SQL)

Common SQL keywords

Keyword	Description
INSERT	Inserts a new row into a table
UPDATE	Update field value in a record
DELETE	Deletes a row from the table
SELECT	Retrieve records from table(s)
INTO	Specifies the table into which to insert the record(s)
FROM	Specifies the table(s) from which to retrieve or delete record(s)
WHERE	Specifies the condition that must be met
ORDER BY	Sorts the records retrieved (does not affect the table)

e.g. **SELECT * FROM employees**



Adding Records

- Use the **INSERT** statement to add individual records to a table
- The syntax for the **INSERT** statement is:

```
INSERT INTO table_name VALUES (value1, value2, ...);
```

OR

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```
- In the first case, the values entered in the **VALUES** list must be in the same order in which you defined the table fields
- Specify **NULL** in any fields for which you do not have a value e.g. for **AUTO_INCREMENT** field



Adding record with INSERT: PHP example



<?php

```
require_once "settings.php";
```

```
$conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
```

```
if ($conn) {
```

```
    $query = "INSERT INTO
```

Field names and values must
be in the same order

Table name

```
tutors (userid, username, password, datejoined)
```

```
VALUES (1,'Alex','8376',curdate());";
```

```
$result = mysqli_query ($conn, $query);
```

```
if ($result) { echo "<p>Insert operation successful.</p>";}
```

```
else { echo "<p>Insert operation unsuccessful.</p>"; }
```

```
mysqli_close ($conn);
```

```
} else echo "<p>Unable to connect to the db.</p>";
```

?>





UPDATE record in PHP example

<?php

```
require_once "settings.php";
$conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
if ($conn) {
    $query = "UPDATE tutors
              SET password='1234'
              WHERE userid = 1";

    $result = mysqli_query ($conn, $query);
    if ($result) {echo "<p>Update operation successful.</p>";}
    else { echo "<p>Update operation unsuccessful.</p>"; }
    mysqli_close ($conn);
} else echo "<p>Unable to connect to the db.</p>";
```

What happens if we forget
the WHERE clause?

?>

21 - Creating Web Applications, © Swinburne



Updating Records

- To update records in a table, use the UPDATE statement
- The syntax for the UPDATE statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```

- The UPDATE keyword specifies the name of the table to update
- The SET keyword specifies the value to assign to the fields in the records that match the condition in the WHERE keyword

22 - Creating Web Applications, © Swinburne





Delete record in PHP example

<?php

```
require_once "settings.php";
$conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
if ($conn) {
    $query = "DELETE FROM tutors WHERE userid = 1";
    $result = mysqli_query ($conn, $query);
    if ($result) { echo "<p>Deleted"
        .mysqli_affected_rows($dbConnect) . " record(s).</p>"; }
    else { echo "<p>Insert operation unsuccessful.</p>"; }
    mysqli_close ($conn);
} else echo "<p>Unable to connect to the db.</p>";
```

?>

23 - Creating Web Applications, © Swinburne



Deleting Records

To Delete records from a table:

- Use the **DELETE** and **WHERE** keywords with the `mysqli_query()` function
- The **WHERE** keyword determines which records to delete in the table
- *Be careful*, if no **WHERE** keyword, *all records are deleted !!*

24 - Creating Web Applications, © Swinburne



Using the `mysqli_affected_rows()` Function



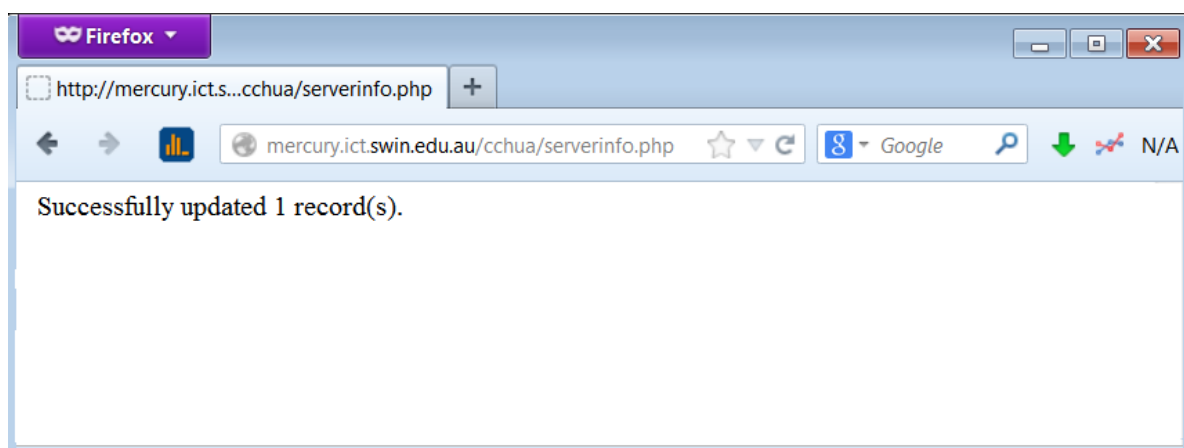
- With queries that modify tables but do not return results (**INSERT**, **UPDATE**, and **DELETE** queries), use the **`mysqli_affected_rows()`** function to determine the *number of affected rows* by the query

```
$sqlString = "UPDATE cars SET price=4500
              WHERE make='Fender' AND model='DG7'";
$queryResult = @mysqli_query($dbConnect, $sqlString);
if ($queryResult){
    echo "<p>Successfully updated "
        . mysqli_affected_rows($dbConnect) . "record(s).</p>";
}
```

25 - Creating Web Applications, © Swinburne



Using the `mysqli_affected_rows()` Function



**Output of `mysqli_affected_rows($con)`
function for an UPDATE query**

26 - Creating Web Applications, © Swinburne



Selecting and Retrieving Records



- Use the `SELECT` statement to retrieve records from a table:

```
SELECT criteria FROM table_name;
```

- Use the asterisk (*) wildcard with the `SELECT` statement to retrieve all fields from a table
- To return multiple fields, separate field names with a comma

```
mysql> SELECT model, quantity FROM inventory;
```

Retrieving Records – Filter



- The **criteria** portion of the `SELECT` statement determines which fields to retrieve from a table
- You can also specify which records to return by using the `WHERE` keyword

```
mysql> SELECT * FROM inventory  
-> WHERE make='Martin';
```

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return

```
mysql> SELECT * FROM inventory  
-> WHERE make='Washburn' AND price<400;
```



Retrieving Records – Sorting

- Use the `ORDER BY` keyword with the `SELECT` statement to perform an alphanumeric sort of the results returned from a query

```
mysql> SELECT make, model FROM inventory  
-> ORDER BY make, model;
```

- To perform a reverse sort, add the `DESC` keyword after the name of the field by which you want to perform the sort

```
mysql> SELECT make, model FROM inventory  
-> ORDER BY make DESC, model;
```

Selecting Records in PHP



Be careful when constructing query:

```
$make = "Holden";
```

```
$dbTable = "inventory";
```

```
$sqlString = "SELECT model, quantity FROM  
$dbTable WHERE model = '$make'";
```

Field name
not in 'quotes'

Variable name
must be in
'quotes' if string

Template 2 – for SQL SELECT queries



<?php

```
require_once "settings.php";
$conn = @mysqli_connect ($host,$user,$pwd,$sql_db);
if ($conn) {
    $query = "replace with a MySQL SELECT query";
    $result = mysqli_query ($conn, $query);
    if ($result) {
        $record = mysqli_fetch_assoc ($result);
        if ($record) {
            echo "<p>At least 1 record was retrieved.</p>";
        } else echo "<p>No records retrieved.</p>";
    } else echo "<p>MySQL operation unsuccessful.</p>";
    mysqli_close ($conn);
} else echo "<p>Unable to connect to the db.</p>";
```

Checks if query successful

Checks if any records exist

?>

Note: we haven't done anything with the records yet

31 - Creating Web Applications, © Swinburne



How to put the records in a html table?



Make	Model	Price	Yr of Manufacture
HOLDEN	ASTRA	14000	2005
FORD	FALCON	39000	2010
HOLDEN	COMMODORE	28000	2009
FORD	ABC	10000	2009
FORD	ESCORT	11000	2007

Output of the cars table in a Web browser



Selecting Records (continued)



Retrieving Records into an Associative Array

- The `mysqli_fetch_assoc()` function returns the fields in the current row of a result set into an associative array and moves the result pointer to the next row

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
    <th>Price</th><th>Yr of Manufacture</th></tr>";
$row = mysqli_fetch_assoc($queryResult);
while ($row) {
    echo "<tr><td>{$row['make']}</td>";
    echo "<td>{$row['model']}</td>";
    echo "<td>{$row['price']}</td>";
    echo "<td>{$row['yom']}</td></tr>";
    $row = mysqli_fetch_assoc($queryResult);
}
echo "</table>";
```

Add \n after the html if you want tidy code. `echo "</table>\n";`

33 - Creating Web Applications, © Swinburne



Selecting Records (continued)



- Assignment and comparison can also be combined to reduce the size of the code

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
    <th>Price</th><th>Yr of Manufacture</th></tr>";

while ($row = mysqli_fetch_assoc($queryResult)) {
    echo "<tr><td>{$row['make']}</td>";
    echo "<td>{$row['model']}</td>";
    echo "<td>{$row['price']}</td>";
    echo "<td>{$row['yom']}</td></tr>";
}
echo "</table>";
```

This is an assignment expression, not a comparison

34 - Creating Web Applications, © Swinburne



Selecting Records (continued)



Function	Description
<code>mysqli_data_seek(\$result, position)</code>	Moves the result pointer to a specific row in the result set
<code>mysqli_fetch_array(\$result, mysqli_assoc mysqli_num mysqli_both)</code>	Returns the fields in the current row of the result set into an associative array, indexed array or both, and moves the result pointer to the next row
<code>mysqli_fetch_assoc(\$result)</code>	Returns the fields in the current row of the result set into an associative array, and moves the result pointer to the next row
<code>mysqli_fetch_row(\$result)</code>	Returns the fields in the current row of the result set into an indexed array, and moves the result pointer to the next row
<code>mysqli_fetch_lengths(\$result)</code>	Returns the field lengths for the current row in a result set into an indexed array

Common PHP functions for accessing database results

35 - Creating Web Applications, © Swinburne



Selecting Records (continued)



- The difference between `mysqli_fetch_assoc()` and `mysqli_fetch_row()` is that instead of returning the fields into an *indexed array*,
- `mysqli_fetch_assoc()` function returns the fields into an *associate array* and uses each ***field name*** as the *array key*

36 - Creating Web Applications, © Swinburne



Selecting Records (continued)



Retrieving Records into an Indexed Array

- The `mysqli_fetch_row()` function returns the fields in the current row of a result set into an indexed array and moves the result pointer to the next row

```
echo "<table border='1'>";
echo "<tr><th>Make</th><th>Model</th>
    <th>Price</th><th>Yr of Manufacture</th></tr>";
$row = mysqli_fetch_row($queryResult);
while ($row) {
    echo "<tr><td>{$row[0]}</td>";
    echo "<td>{$row[1]}</td>";
    echo "<td>{$row[2]}</td>";
    echo "<td>{$row[3]}</td></tr>";
    $row = mysqli_fetch_row($queryResult);
}
echo "</table>";
```

Add \n after the html if you want tidy code. `echo "</table>\n";`

37 - Creating Web Applications, © Swinburne



Selecting Records (continued)



Accessing Query Result Information for queries that return result sets:

- The `mysqli_num_rows()` function returns the number of rows in a query result
- The `mysqli_num_fields()` function returns the number of fields in a query result
- Both functions accept a database result variable, eg. a query result, as an argument

38 - Creating Web Applications, © Swinburne



Selecting Records (continued)



**Output of the number of rows and fields
returned from a query**

Cleaning Up



- When you are finished working with query results retrieved with the `mysqli_query()` function, use the `mysqli_free_result()` function to close the resultset
- To close the resultset, pass to the `mysqli_free_result()` function the variable containing the result pointer from the `mysqli_query()` function
e.g. `mysqli_free_result($queryResult);`

Understanding the Basics of Databases

- Working with MySQL Databases
- Managing Databases and their Tables
- Managing Tables and their Records

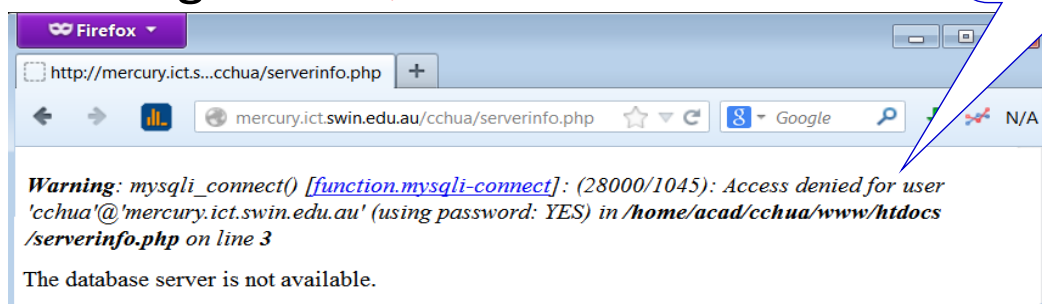
Accessing Databases with PHP

- Creating and Deleting Databases and Tables
- Selecting, Creating, Updating, and Deleting Records
- • Handling errors

Handling MySQL Errors

- Reasons for not connecting to a database server include:
 - The database server is not running
 - Insufficient privileges to access the data
 - Invalid username and/or password
- e.g. `if (!$dbConnect) ...`

We do not want users to see any database error messages !



Database connection error message

Handling MySQL Errors



Suppressing Errors with the Error Control Operator

- Writing code that anticipates and handles potential problems is often called **bulletproofing**
- Bulletproofing techniques include:

- Checking submitted form data

e.g. `if (isset($_GET['height'])) ...`

- Using the **error control operator (@)** to suppress error messages

e.g. `$dbConnect = @mysqli_connect(...);`
`if (!$dbConnect) ...`

Handling MySQL Errors



Terminating Script Execution

- **die()** and **exit()** **terminate** script execution
- **die()** version is usually used when attempting to access a data source
- Both functions accept a single string argument
- Invoke the **die()** and **exit()** as separate statements or by appending either function to an expression with the **or** operator

Note: When script is **terminated**, an *incomplete html page* is sent to the client. This is useful for error diagnostics, *but poor in a production application*.

Handling MySQL Errors (continued)



```
$dbConnect = @mysqli_connect(("feenix-mariadb.swin.edu.au",  
    "s1234567", "ddmmyy")  
    or die("<p>The database server is not available.</p>");  
  
// the above is one statement: connected OK or die  
  
echo "<p>Successfully connected to the database server.</p>";  
  
@mysqli_select_db($dbConnect, "s1234567_db")  
    or die("<p>The database is not available.</p>");  
  
echo "<p>Successfully opened the database.</p>";  
  
// additional statements that access the database server  
  
mysqli_close($dbConnect);
```

No if required here

45 - Creating Web Applications, © Swinburne



Handling MySQL Errors (continued)



MySQL error reporting functions

Function	Description
mysqli_connect_errno()	Returns the error code from the last database connection attempt, 0 if no error
mysqli_connect_error()	Returns the error message from the last database connection attempt, empty string if no error
mysqli_errno(connection)	Returns the error code from the last MySQL function call attempted, 0 if no error
mysqli_error(connection)	Returns the error message from the last MySQL function call attempted, empty string if no error
mysqli_sqlstate(connection)	Returns a string of five character error code from the last MySQL operation, '00000' if no error

46 - Creating Web Applications, © Swinburne



Handling MySQL Errors (continued)

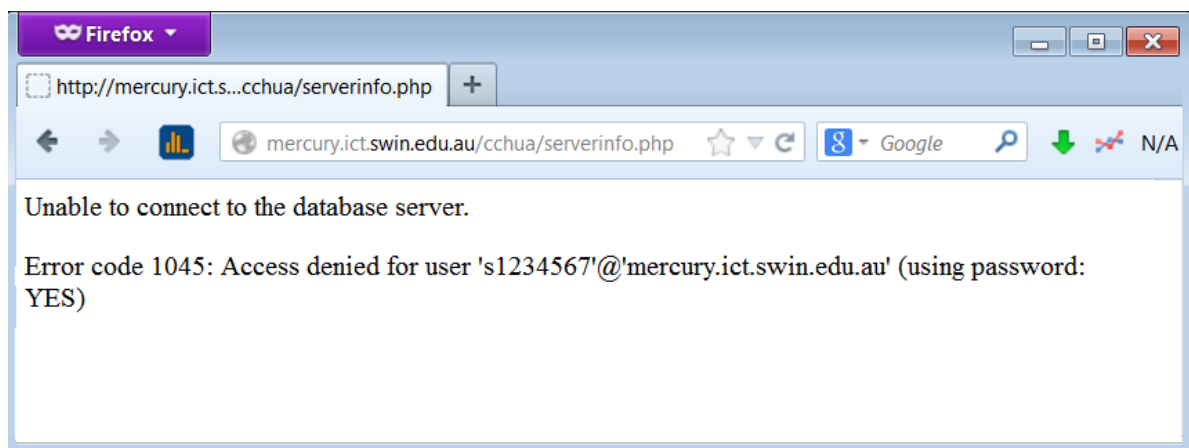


```
$user = $_GET['username'];
$password = $_GET['password'];
$dbConnect = @mysqli_connect("feenix-mariadb.swin.edu.au",
    $user, $password)
    or die("<p>Unable to connect to the database server.</p>"
        . "<p>Error code " . mysqli_connect_errno()
        . ": " . mysqli_connect_error() . "</p>");
echo "<p>Successfully connected to the database server.</p>";
@mysqli_select_db($dbConnect, "s1234567_db")
    or die("<p>The database is not available.</p>");
echo "<p>Successfully opened the database.</p>";
// additional statements that access the database
mysqli_close($dbConnect);
```

47 - Creating Web Applications, © Swinburne



Handling MySQL Errors (continued)



**Error number and message generated by
an invalid username and/or password**

48 - Creating Web Applications, © Swinburne



Reminder: Checking Data Entry



- ***Never trust the user! Never!***
 - **Always** check that input values are of the **type** you expect
 - If possible, test that a text value is **within** a **set** of values
 - If showing the content gathered from users, **remove** anything that shouldn't be there, and **encode** everything else to make sure that nothing is **inserted** into your code! (HTML, JS, CSS or other!)
 - If using information from users as part of a database **query**, **escape** all (string) values, always surround values with **quotes** and log/test whatever you can.

49 - Creating Web Applications, © Swinburne



Next Week



- **Web security**
- **Recent trends**

50 - Creating Web Applications, © Swinburne

