

FIT2094-FIT3171 Databases
Week 12 Tutorial Activities
DB Connectivity, Web Technology
FIT Database Teaching Team

Complete Week 12 activities listed below

[12.1 Database Connectivity](#)

[12.2 Web-Database Connectivity using PHP](#)

[12.3 Web Frameworks and Security Consideration](#)

[12.3.1 Web Frameworks](#)

[12.3.2 SQL Injection](#)

[12.4 Web Modification Exercise](#)

[12.5 SETU](#)

FIT2094-FIT3171 2021 S2

FIT2094-FIT3171 Databases

Author: FIT Database Teaching Team

License: Copyright © Monash University, unless otherwise stated. All Rights Reserved.

COPYRIGHT WARNING

Warning

This material is protected by copyright. For use within Monash University only. NOT FOR RESALE.

Do not remove this notice.

Important

Remember before starting any lab activity which involves working with files, first use SQLDeveloper to pull from the FIT GitLab server so as to ensure your local files and the FIT GitLab server files are in sync.

Learning Objectives:

- understand the different technologies used to connect a database to the application (web app/frontend)
- be able to write PHP code to:
 - make and close connections from a web page to a database, and
 - retrieve data from the database and display in a web browser.
- understand security practices for database front ends including mitigation of SQL injection (SQLi)

12.1 Database Connectivity

Normal users interact with the database via application. For example, to pick a tutorial or move from one tutorial to another, students use Allocate+. In this section, we will discuss how an application communicates with the database server.

Discuss following terms:

1. Data layer
2. Application layer
3. Database middleware
 - a. ODBC
 - b. JDBC
 - c. OLE-DB
4. Web server
5. Web to database middleware

12.2 Web-Database Connectivity using PHP

Now that you are familiar with designing, creating and managing tables we will look at the manner in which such data can be accessed. To date you have been using SQL Developer, clearly, in practice, your users will not have access to/use this item of software. Access to your created tables by normal users will be via an application or web front end.

For this tutorial, we are going to develop a basic web front end. To do this we are going to make use of [PHP](#) (recursive acronym for PHP: Hypertext Preprocessor) one of the most [widely used programming languages](#), especially for web development. Please note that this is a very basic introduction so that you become aware of the possibilities. PHP is a full OO language with a wide range of features. Development under PHP is commercially carried out via [PHP Frameworks](#).

PHP enables the mixing of PHP code (marked between `<?php` and `?>`) and standard HTML code within a single file. When accessed via the web server the PHP code is handled via the PHP processor on the web server and replaced with appropriate output. For this unit we are not expecting you to become PHP experts, this is simply an exercise to increase your awareness of how a database can be accessed via the web. If you wish to delve further into PHP immediately there are a large number of good tutorials available on the web. A good starting point is <https://www.w3schools.com/php/>

The steps in using PHP to access table data are:

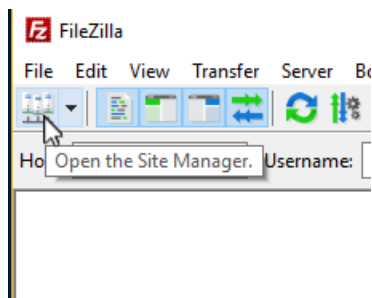
- connect to the database
- define a SQL query string
- parse the SQL query against the database
- execute the statement
- fetch and display the data, and finally
- free the resources being used and close the connection.

You have been given an account on the server <http://fit-db.infotech.monash.edu/> – this server can be accessed in two ways:

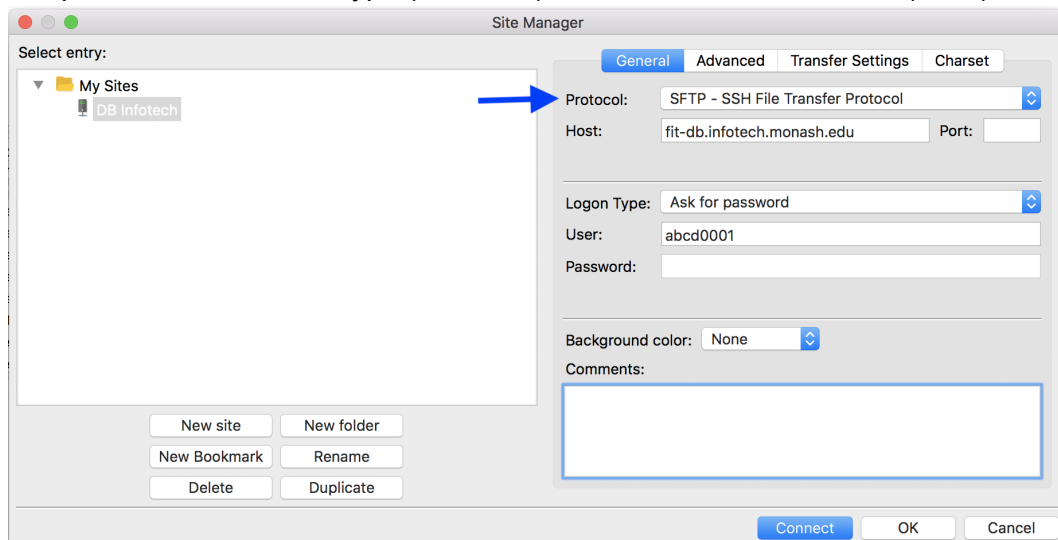
- via the web by using a URL of the form:
<http://fit-db.infotech.monash.edu/~yourauthcateusername/> or
- via SFTP for transferring files to the server.

This server can only be accessed within the Monash network or via [Monash VPN](#) or using MoVE.

To create a connection to the server open an SFTP client such as [FileZilla](#). With FileZilla select Open the Site Manager:



and pick **SFTP** as access type (**Protocol**) and enter the server name (**Host**):



Before selecting “Connect”, click on the NewSite under My Sites on the left and give your connection an appropriate name (here DB Infotech). Also please ensure you use the “Logon Type” “Ask for password” so that FileZilla will prompt you for your password at each connection and not remember it. Use your [Monash username and password](#) and click “Connect”.

After clicking on “Connect”, your connection will open and FileZilla will show your local files on the left and the server (remote) files on the right. Navigate using the right until you are located in your public_html folder by keying in **/srv/home/<yourauthcate>/public_html** in Remote site box and hit enter:



Documents placed in this folder will be published via the web server running on <http://fit-db.infotech.monash.edu/>.

To create or edit the PHP files we will need you should make use of a text editor such as [Visual Studio Code](#), [Atom](#) or [Brackets](#)

Download **week12_tutorial_files.zip** from the Week 12 block in Moodle. The zip file consists of four php files which are required for this tutorial. Unzip and place these files in your working directory (in your Tut12 folder).

The first PHP script we wish to discuss is a form which will ask you to input your Oracle connection details and call the page which will display the data from the database. The file is called **login_uni_student.php** and have the form:

```
1 <html>
2   <body>
3     <h2>Login to your Oracle Account</h2>
4     <form action="disp_uni_student.php" method="post">
5       User Name: <input type="text" name="username"><br>
6       Password: <input type="password" name="password"><br>
7         <input type="submit">
8     </form>
9   </body>
10 </html>
```

Now we wish to discuss the PHP script which accesses and displays the student data in the student table of the UNIVERSITY database that you used in previous weeks' tutorials. The file is called **disp_uni_student.php**. The file contains code to connect to the database and carry out a select of the student table and display the results in an [HTML table](#). You should look through the code and understand the details of what has been coded:

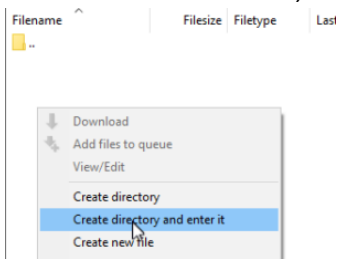
```
1 <html>
2 <head>
3 <title>Student List</title>
4 </head>
5 <body>
6 <h1>Student list UNIVERSITY database</h1>
7
8 <?php
9 // Set up the Oracle connection string
10 $dbInstance = "(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
11 (HOST=ora-fit.ocio.monash.edu)(PORT=1521))
12 (CONNECT_DATA=(SID=FITUGDB)))";
13
14 // Connect to the database - Open Oracle connection
15 $conn = oci_connect($_POST["username"], $_POST["password"], $dbInstance);
16 if (!$conn) {
17     $e = oci_error();
18     print "Error connecting to the database:<br>" ;
19     print $e['message'] ;
20     exit;
21 }
22 ?>
23
24 <!-- create HTML table header to display output -->
25 <table border =1 width=650>
26 <tr>
27 <th width=100><b>Student ID</b></th>
28 <th width=200><b>Name</b></th>
29 <th width=150><b>Date of Birth</b></th>
30 <th width=200><b>Email</b></th>
31 </tr>
32
33 <?php
34 //SQL query statement
35 $query = "SELECT
36         studid,
37         rtrim(studfname)
38         || ' '
39         || rtrim(studlname) AS sname,
40         to_char(studdob, 'dd-Mon-yyyy') AS sbdate,
41         studemail
42     FROM
43         uni.student
44     ORDER BY
45         studid";
46
```

```

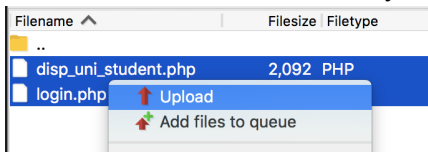
47 //Parse statement
48 $stmt = oci_parse($conn,$query);
49 ▼ if (!$stmt) {
50     $e = oci_error($conn);
51     print "Error on parse of statement:<br>" ;
52     print $e['message'] ;
53     exit;
54 }
55
56 // oci_define_by_name maps SQL Columns in a query to PHP variable names
57 // MUST be done before executing, Oracle names must be in UPPER case
58 oci_define_by_name($stmt,"STUDID",$studid);
59 oci_define_by_name($stmt,"SNAME",$stuname);
60 oci_define_by_name($stmt,"SBDATE",$stubdate);
61 oci_define_by_name($stmt,"STUEMAIL",$stuemail);
62
63 // Execute the STATEMENT
64 $r = oci_execute($stmt);
65 ▼ if (!$r) {
66     $e = oci_error($stmt);
67     print "Error execute of statement:<br>" ;
68     print $e['message'] ;
69     exit;
70 }
71
72 // Fetch the results of the query
73 ▼ while (oci_fetch($stmt)) {
74     print("
75         <tr>
76             <td width=100>$studid</td>
77             <td width=200>$stuname</td>
78             <td width=150>$stubdate</td>
79             <td width=200>$stuemail</td>
80         </tr>");
81 }
82
83 // Close table
84 print("</table>");
85
86 // Show the number of rows
87 $no_rows = oci_num_rows($stmt);
88 print "<p>Rows found:" . $no_rows . "</p>";
89
90 // Free resources associated with Oracle statement
91 oci_free_statement($stmt);
92 // Close the Oracle connection
93 oci_close($conn);
94 ?>
95 </body>
96 </html>

```

We now wish to transfer both files to the remote server into a folder below public_html called uni. First, right-click in your remote public_html folder and create a **uni** folder (select "Create directory and enter it" in Filezilla):\



Then locate your **login_uni_student.php** and **disp_uni_student.php** files in your local site file browser and transfer them to your remote site uni folder via upload:



Well done, you are now ready to examine your work. Go to the URL:

http://fit-db.infotech.monash.edu/~yourauthcate/uni/login_uni_student.php

(replace yourauthcate with your actual authcate/username e.g. abcd0001), and you should see something like:

 A screenshot of a web browser window. The address bar shows the URL 'fit-db.infotech.monash.edu/~authcate/uni/login_uni_student.php'. The page title is 'Login to your Oracle Account'. The form has two input fields: 'User Name:' and 'Password:'. Below the fields is a 'Submit' button.

Insert your Oracle username and password then click submit and it will redirect you to disp_uni_student.php and you should see something like:

 A screenshot of a web browser window. The address bar shows the URL 'fit-db.infotech.monash.edu/~authcate/uni/disp_uni_student.php'. The page title is 'Student list UNIVERSITY database'. The table has four columns: Student ID, Name, Date of Birth, and Email. The table contains 20 rows of student data.

Student ID	Name	Date of Birth	Email
12489379	Gilberto Bwy	30-Aug-1992	Gilberto.Bwy@student.monash.edu
12511467	Francyne Rigney	18-Jan-1992	Francyne.Rigney@student.monash.edu
12609485	Cassandra Sedcole	07-Aug-1990	Cassandra.Sedcole@student.monash.edu
12802225	Friedrick Geist	02-Mar-1997	Friedrick.Geist@student.monash.edu
12842838	Herminia Mendus	25-Apr-1996	Herminia.Mendus@student.monash.edu
13028303	Herculie Mendus	02-Aug-1998	Herculie.Mendus@student.monash.edu
13119134	Shandra Lindblom	20-Apr-2000	Shandra.Lindblom@student.monash.edu
13390148	Brier Kilgour	21-Feb-1995	Brier.Kilgour@student.monash.edu
13413109	Myriam Stirley	07-May-1994	Myriam.Stirley@student.monash.edu
13453333	Pierrette Moynihan	10-Jan-1993	Pierrette.Moynihan@student.monash.edu
13742778	Warden Eyden	31-Aug-1999	Warden.Eyden@student.monash.edu
13880303	Shadow Lamberton	24-Jul-1996	Shadow.Lamberton@student.monash.edu
14374036	Claudette Serman	21-May-1998	Claudette.Serman@student.monash.edu
14615430	Siffre Dibdale	06-May-1992	Siffre.Dibdale@student.monash.edu
14635701	Cord Yard	02-Jul-1990	Cord.Yard@student.monash.edu
14676780	Niki Sperrett	08-Jul-1993	Niki.Sperrett@student.monash.edu
14900897	Casper Rottiger	01-Apr-1994	Casper.Rottiger@student.monash.edu
14996864	Odele Toby	19-Feb-1996	Odele.Toby2@student.monash.edu
15067408	Eada Royds	14-Dec-1991	Eada.Royds@student.monash.edu
15127454	Lizabeth Stubbings	02-Jan-1995	Lizabeth.Stubbings@student.monash.edu
15543217	Averell Corp	11-Dec-1996	Averell.Corp@student.monash.edu
15875842	Marci Blabber	25-Aug-1990	Marci.Blabber@student.monash.edu
16476616	Mattheus Kardos	05-Jun-1993	Mattheus.Kardos@student.monash.edu
16929043	Billie Friedank	27-Mar-1997	Billie.Friedank@student.monash.edu
17013887	Harv Wothersed	22-Jul-1993	Harv.Wothersed@student.monash.edu

12.3 Web Frameworks and Security Consideration

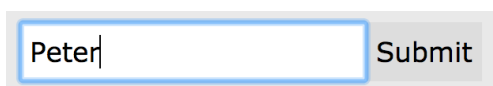
12.3.1 Web Frameworks

Nowadays, the use of frameworks is quite popular to develop entire interfaces and web applications (e.g. a Customer-Relationship Management app, using a database backend).

1. Name some other popular web frameworks that can utilise Oracle as a database back-end.
2. A common programming technique used in many frameworks (including Oracle) is Object-Relational Mapping (ORM). Briefly describe what it means.

12.3.2 SQL Injection

SQL Injection is a very common vulnerability when it comes to building web frontends (e.g. in PHP) for databases. To understand how serious the issue is, let's assume you have a website which lets you enter a first name as a search query:

A screenshot of a web form. It consists of a text input field with a blue border containing the text 'Peter' and a grey 'Submit' button to its right.

The website then uses your search string (e.g. "Peter") and places it in a SQL SELECT statement so that it can show you results, using the following SQL query. (Your search string is highlighted).

```
SELECT * FROM users WHERE first_name = 'Peter';
```

Aside from above example, you can also use this site to learn more about SQL injection:

https://www.w3schools.com/sql/sql_injection.asp

1. Discuss what SQL Injection means. In the example above, how can a malicious user craft a special search string in order to, say, view everything in another table they're not supposed to view?
2. How can you prevent these from happening to your own application?

12.4 Web Modification Exercise

This section required `login_uni_scheduled_class.php` and `disp_uni_scheduled_class.php` provided in the zip file which was discussed in section 12.2 above. These php files aim to display all lectures scheduled in semester 1 2020, when the unit detail is clicked, the information of the unit's lecturer will be displayed in a pop up window. The expected output is something like:

Scheduled Lecture S1 2020 - UNIVERSITY database

UNIT	DAY	TIME	DURATION
FIT1045 Algorithms and programming fundamentals in python	Mon	09:00	2 hours
FIT1050 Web fundamentals	Tue	11:00	2 hours
FIT2094 Databases	Mon	09:00	2 hours
FIT3157 Advanced web design	Wed	09:00	2 hours
FIT3176 Advanced database design	Thu	11:00	2 hours
FIT5145 Introduction to data science	Wed	13:00	1 hour
FIT5196 Data wrangling	Fri	11:00	1 hour
FIT9132 Introduction to databases	Fri	09:00	1 hour
FIT9136 Algorithms and programming foundations in Python	Tue	11:00	1 hour
FIT9137 Introduction to computer architecture and networks	Mon	09:00	1 hour

Rows found:10

and when the unit name is clicked the pop up window appears like:

Scheduled Lecture S1 2020

fit-db.infotech.monash.edu says

Lecturer Name: Sandro Wethered

OK

UNIT	DAY	TIME	DURATION
FIT1045 Algorithms and programming fundamentals in python	Mon	09:00	2 hours
FIT1050 Web fundamentals	Tue	11:00	2 hours
FIT2094 Databases	Mon	09:00	2 hours
FIT3157 Advanced web design	Wed	09:00	2 hours
FIT3176 Advanced database design	Thu	11:00	2 hours
FIT5145 Introduction to data science	Wed	13:00	1 hour
FIT5196 Data wrangling	Fri	11:00	1 hour
FIT9132 Introduction to databases	Fri	09:00	1 hour
FIT9136 Algorithms and programming foundations in Python	Tue	11:00	1 hour
FIT9137 Introduction to computer architecture and networks	Mon	09:00	1 hour

Rows found:10

You should look through the code in both files and understand the details of what has been coded, then complete the **\$query** part with an SQL select statement. Check if your web page works and displays correct data by uploading the files onto fit-db.infotech.monash.edu server.

12.5 SETU

Spend the rest of the time of this tutorial to fill up the SETU.

Important

You need to get into the habit of establishing this as a standard FIT2094-FIT3171 workflow - Pull at the start of your working session, work on the activities you wish to/are able to complete during this session, add (stage), commit changes and then Push the changes back to the FIT GitLab server