

Chapter 3

Database

Programming

3.1 Database Programming

- PHP is particularly strong in its ability to interact with databases.
- PHP supports pretty much every database out there
- The most commonly used database with PHP is MySQL
- MySQL is a free and open source database that has a lot of users especially for web applications.
- It can be downloaded from internet freely.
- Whichever database you're using, the steps to interact with a database are similar:
 1. Connect to the database.
 2. Send an SQL query that contains instructions for the database software.
 3. If you retrieved data from the database, process the data.
 4. Close the connection to the database.

3.1 Database Programming...

□ What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- The data in a MySQL database are stored in tables.
- A table is a collection of related data, and it consists of columns and rows.

3.1 Database Programming...

Connecting to the database

- The first step in a database interaction is connecting to the database.
- You use a PHP function to connect to the database.
- To make the connection, you need to supply the function with three things:
 - **Location:**
 - The database may not be on the same computer where PHP is installed.
 - Therefore, you need to tell the PHP connect function the name of the computer where the database is located.
 - You can supply either a domain name (such as mycompany.com) or an IP address (such as 172.17.204.2).
 - If the database is on the same computer as PHP, you can use localhost for the hostname.
 - **Account name:** You must provide a valid account name that can be used to access the database.
 - **Password:** You have to have a valid password to access the database.

3.1 Database Programming...

- The syntax for creating connection to MySQL database:

```
<?php
$connect = mysql_connect($host, $account,
$password);
$servername = "localhost";
$username = "root";
$password = "";
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
} else
echo "Connected successfully";
?>
```

3.1 Database Programming...

- **Create a MySQL Database Using MySQL:** The CREATE DATABASE statement is used to create a database in MySQL. The following examples

```
<?php
create a database named "myDB":
$servername = "localhost";
$username = "root";
$password = "";
$conn = new mysqli($servername, $username, $password);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error); }
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE){echo "Database";
} else { echo "Error creating database: " . $conn->error; }
$conn->close();
?>
```

3.1 Database Programming...

□ Create a MySQL Table Using MySQLi:

- The **CREATE TABLE** statement is used to create a table in MySQL.
- We will create a table named "**MyGuests**", with five columns: "***id***", "***firstname***", "***lastname***", "***email***" and "***reg_date***":

```
<?php
```

```
CREATE TABLE MyGuests (  
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
firstname VARCHAR(30) NOT NULL,  
lastname VARCHAR(30) NOT NULL,  
email VARCHAR(50),  
reg_date TIMESTAMP  
) ?>
```

3.1 Database Programming...

- The data type specifies what type of data the column can hold. After the data type, you can specify other optional attributes for each column:
 - NOT NULL - Each row must contain a value for that column, null values are not allowed
 - DEFAULT value - Set a default value that is added when no other value is passed
 - UNSIGNED - Used for number types, limits the stored data to positive numbers and zero
 - AUTO INCREMENT - MySQL automatically increases the value of the field by 1 each time a new record is added
 - PRIMARY KEY - Used to uniquely identify the rows in a table. The column with PRIMARY KEY setting is often an ID number, and is often used with AUTO_INCREMENT

3.1 Database Programming...

- ❑ The following example shows how to create the table in PHP:

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error); }
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";
if ($conn->query($sql) === TRUE) { echo "Table MyGuests created";
} else { echo "Error creating table: " . $conn->error;
$conn->close();
?>
```

3.1 Database Programming...

- After connecting to the database, the next step is selecting database.
- An DBMS can create and maintain many databases, so you need to tell it which database you want to use.

- The syntax for selecting database is:

```
$db = mysql_select_db(string database, [int  
database_connection] );
```

- Example:

```
<?php  
    $con = mysql_connect("localhost","root","vertrigo");  
    mysql_select_db("books");  
    //other statements  
?>
```

3.1 Database Programming...

Querying the Database

- To actually perform the query, we can use the `mysql_query()` function.
- Before doing this, however, it's a good idea to set up the query you want to run:
`$query = "select * from books";`
- We can now run the query:
`$result = mysql_query($query);`
- The `mysql_query()` function has the following prototype:
`int mysql_query(string query, [int dbconnection] );`
- You pass it the query you want to run, and optionally, the database connection created.

3.1 Database Programming...

- If connection isn't specified, the function will use the last opened link.
- If there isn't one, the function will open the default one as if you had called `mysql_connect()`.
- You might want to use the `mysql_db_query()` function instead.
`int mysql_db_query(string database, string query, [int dbconnection])`;
);
- It's very similar but allows you to specify which database you would like to run the query on.
- It is like a combination of the `mysql_select_db()` and `mysql_query()` functions.
- Both of these functions return a result identifier that allows you to retrieve the query results or false on failure.

3.1 Database Programming...

- Example: executing query

```
$sql = "SELECT * FROM Product";  
$result = mysql_query($sql, $connect);
```

- Putting all together:

```
<?php  
$account = "david";  
$password = "gotago";  
$host= "localhost";  
$connect = mysql_connect($host, $account, $password);  
$db = mysql_select_db("Catalog", $connect);  
$sql = "SELECT * FROM Product";  
$result = mysql_query($sql, $connect);  
//other code  
?>
```

3.1 Database Programming...

□ Customers table

ID	Name	Address	Sex	City	income
1	John Walter	23 New Street	M	New York	43254
2	Samuel David	Que street 70	M	Montreal	54242
3	George Graham	Lamore 32	M	Paris	26343
4	Ana Bush	Prus street 52	F	Berlin	34627
5	Li Xien	Tianmen square 30	M	Beijing	9056

<i>Privilege</i>	<i>Applies To</i>	<i>Description</i>
SELECT	tables, columns	Allows users to select rows (records) from tables.
INSERT	tables, columns	Allows users to insert new rows into tables.
UPDATE	tables, columns	Allows users to modify values in existing table rows.
DELETE	tables	Allows users to delete existing table rows.
INDEX	tables	Allows users to create and drop indexes on particular tables.
ALTER	tables	Allows users to alter the structure of existing tables by, for example, adding columns, renaming columns or tables, and changing data types of columns.
CREATE	databases, tables	Allows users to create new databases or tables. If a particular database or table is specified in the GRANT, they can only CREATE that database or table, which means they will have to DROP it first.
DROP	databases, tables	Allows users to drop (delete) databases or tables.

3.1 Database Programming...

- ❑ Create table Customers(
customerid int not null auto_increment primary key,
name varchar(30) not null,
sex varchar(15) not null,
address varchar(40) not null,
city varchar(20) not null,
income int,
);
- ❑ Insert into table[(column1, column2, column3, ...)] values
(value1, value2, value3, ...)
- ❑ Insert into customers(customerID, name, address, city,
income) values(NULL, "Anna Schmit", '25 Oak Street',
'Qirport West', 3000)

3.1 Database Programming...

- Select *columnname list* or *
From *tablename*
[Where *condition*]
[Group By *group type*]
[Having *where_definition*]
[Order By *order_type* [ASC|DESC]]
[Limit *limit_criteria*];
- Select name, city
from Customers
- Select *
from Customers
where city='Jimma'

3.1 Database Programming...

<i>Name</i>	<i>Description</i>
AVG(column)	Average of values in the specified column.
COUNT(items)	If you specify a column, this will give you the number of non-NULL values in that column. If you add the word DISTINCT in front of the column name, you will get a count of the distinct values in that column only. If you specify COUNT(*), you will get a row count regardless of NULL values.
MIN(column)	Minimum of values in the specified column.
MAX(column)	Maximum of values in the specified column.
STD(column)	Standard deviation of values in the specified column.
STDDEV(column)	Same as STD(column).
SUM(column)	Sum of values in the specified column.

3.1 Database Programming...

- ❑ `Select AVG(income)`
`From customers`
- ❑ `Select count(*)`
`From customers`
- ❑ `SELECT column_name(s)`
`FROM table1LEFT JOIN|RIGHT JOIN|INNER JOIN|JOIN table2`
`ON table1.columnname=table2.columnname`

`SELECT *`

`FROM customer inner join security`

`on customer.CustomerID=security.userID`

3.1 Database Programming...

```
SELECT name, city, username  
FROM customer left join security  
on customer.CustomerID=security.userID
```

- Update tablename
Set column1=value1, column2=value2,...
[where *condition*]
- Update customers
Set address='250 Olsens Road', city='New York'
Where customerid='4'

3.1 Database Programming...

- Delete from tablename
[Where condition]
- Delete from customers
where customerid=5;
- What does this do?
Delete from table;

3.1 Database Programming...

- Alter table *tablename*
ADD|DROP|MODIFY|CHANGE Column
- Alter table customers
Add Column sex varchar(10) not null
After name;
- Alter table customers
Drop city
- Alter table customers
Modify name varchar(50) not null;

3.1 Database Programming...

- Drop table *tablename*
Drop table customers
- Drop database *databasename*
Drop database lab

3.1 Database Programming...

- Example: executing query

```
$sql = "SELECT * FROM Product";  
$result = mysql_query($sql, $connect);
```

- Putting all together:

```
<?php  
$account = "david";  
$password = "gotago";  
$connect = mysql_connect($host, $account, $password);  
$db = mysql_select_db("Catalog", $connect);  
$sql = "SELECT * FROM Product";  
$result = mysql_query($sql, $connect);  
//other code  
?>
```

3.1 Database Programming...

Processing Data

- To process the data returned from database, you need to get it from the temporary table where it is placed when the SQL query is executed.
- We use PHP database functions to get the data from the temporary table.
- The data is stored in the temporary table in rows and columns.
- You can use PHP functions to retrieve one row from the table and store it in an array
- The array has the field names as the array keys.
- For MySQL, the statement is as follows:
`$row = mysql_fetch_array($result);`

3.1 Database Programming...

- It requires one of the `mysql_fetch` functions to make the data fully available to PHP.
- The fetching functions of PHP are as follows:
 - `mysql_fetch_array`: returns rows as associative or numbered array
 - `mysql_fetch_assoc`: returns rows as associative array
 - `mysql_fetch_row`: returns row as an enumerated array
 - `mysql_fetch_object`: returns row as an object
 - `mysql_result`: returns one cell of data
- The difference between the three main fetching functions is small.

3.1 Database Programming...

- The most useful fetching function, `mysql_fetch_array`, offers the choice of results as an associative or an enumerated array or both
- The default is returning as both (index or key).
- This means you can refer to outputs by database field name rather than number:

```
$query = "SELECT ID, LastName, FirstName FROM users WHERE Status = 1";  
$result = mysql_query($query);  
while ($row = mysql_fetch_array($result))  
{  
    echo "$row['ID'], $row['LastName'], $row['FirstName']<BR>\n";  
}
```

3.1 Database Programming...

- `mysql_fetch_array` can also be used with numerical identifiers rather than field names.
- If you want to specify index or field name rather than making both available, you can do it like this:

```
$offset_row = mysql_fetch_array($result, MYSQL_NUM);  
$associative_row = mysql_fetch_array($result,  
MYSQL_ASSOC);
```

- It's also possible to use `MYSQL_BOTH` as the second value.

```
$both = mysql_fetch_array($result, MYSQL_BOTH);
```

3.1 Database Programming...

- The `mysql_fetch_assoc` function returns an associative array that corresponds to the fetched row.
- It returns `FALSE` if there are no more rows.
- `mysql_fetch_assoc()` is equivalent to calling `mysql_fetch_array()` with `MYSQL_ASSOC` for the optional second parameter.
- It only returns an associative array.

```
$query = "SELECT ID, LastName, FirstName FROM users WHERE Status =  
1";  
$result = mysql_query($query);  
while ($row = mysql_fetch_assoc($result))  
{  
    echo "$row['ID'], $row['LastName'], $row['FirstName']<BR>\n";  
}
```

3.1 Database Programming...

- The function `mysql_fetch_object` performs much the same task
- But the row is returned as an object rather than an array.
- Obviously, this is helpful for those among those who utilize the object-oriented notation:

```
$query="SELECT ID, LastName, FirstName FROM users WHERE  
    Status = 1";
```

```
$result = mysql_query($query);
```

```
while ($row = mysql_fetch_object($result))
```

```
{
```

```
    echo $row->ID;
```

```
    echo $row->FirstName;
```

```
    echo $row->LastName<BR>\n";
```

```
}
```

3.1 Database Programming...

- The most general one is `mysql_fetch_row`, which can be used something like this:

```
$query = "SELECT ID, LastName, FirstName FROM users WHERE  
Status = 1";
```

```
$result = mysql_query($query);
```

```
while ($name_row = mysql_fetch_row($result))
```

```
{
```

```
    print("$name_row[0] $name_row[1] $name_row[2]<BR>\n");
```

```
}
```

- This code will output the specified rows from the database, each line containing one row or the information associated with a unique ID.

3.1 Database Programming...

- In early versions of PHP, `mysql_fetch_row` was considered to be faster than `mysql_fetch_object` and `mysql_fetch_array`, but this is no longer an issue.
- The PHP group now recommends use of `mysql_fetch_array` over `mysql_fetch_row` because it offers increased functionality and choice.
- Last and least of the fetching functions is `mysql_result()`.
- Use this function only in situations where you are positive you need only one piece of data to be returned from MySQL.

```
$query = "SELECT count(*) FROM users";  
$db_result = mysql_query($query);  
$datapoint = mysql_result($db_result, 0, 0);
```
- The `mysql_result` function takes three arguments: result identifier, row identifier, and field.
- Field can take the value of the field offset as above, or its name as in an associative array, or its MySQL field-dot-table name.

- Example: a program that retrieves data from database and display in table

```
<?php
```

```
$con = mysql_connect("localhost","root","vertrigo");  
mysql_select_db("customers");  
$query = "SELECT * FROM users";  
$result = mysql_query($query);  
echo("<TABLE border='1'>");  
while ($row = mysql_fetch_array($result)) {  
    echo "<tr>";  
    echo('<TD> '.$row["ID"].' </TD> ');  
    echo('<TD> '.$row["firstname"] . ' </TD> ');  
    echo('<TD> '.$row["lastname"] . ' </TD> ');  
    echo('<TD> '.$row["sex"].' </TD> ');  
    echo "</tr>";  
}  
echo("</TABLE><BR>\n");
```

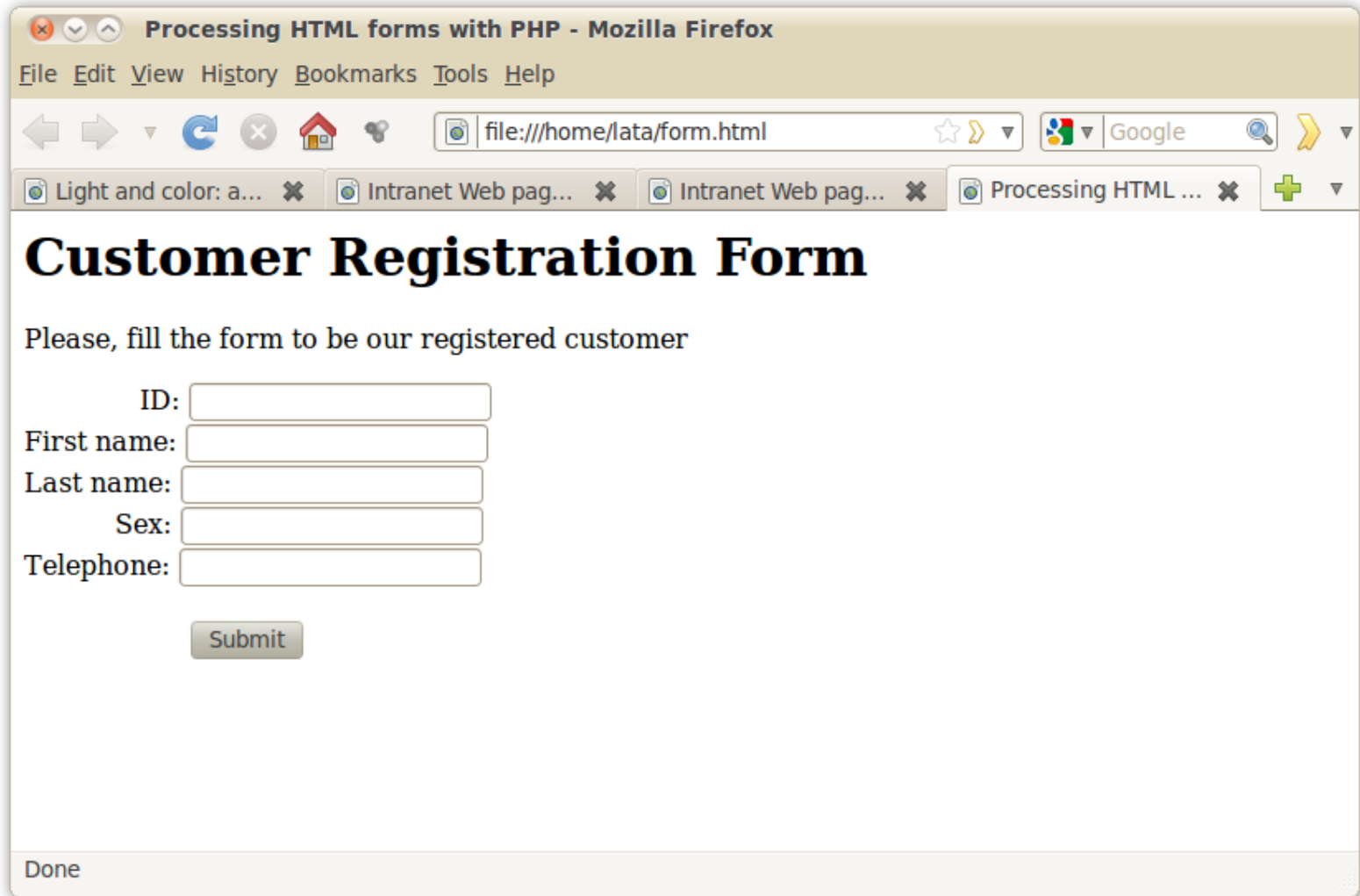
```
?>
```

3.1 Database Programming...

Inserting Data into Database

- Inserting new items into the database is remarkably similar to getting items out of the database.
- You follow the same basic steps
 - make a connection,
 - send a query, and
 - check the results.
- In this case, the query you send will be an INSERT rather than a SELECT.

3.1 Database Programming...



The screenshot shows a Mozilla Firefox browser window titled "Processing HTML forms with PHP - Mozilla Firefox". The address bar displays "file:///home/lata/form.html". The browser's menu bar includes File, Edit, View, History, Bookmarks, Tools, and Help. The tab bar shows four tabs: "Light and color: a...", "Intranet Web pag...", "Intranet Web pag...", and "Processing HTML ...". The main content area features a heading "Customer Registration Form" in a large, bold, black serif font. Below the heading is a paragraph: "Please, fill the form to be our registered customer". The form consists of five labeled text input fields: "ID:", "First name:", "Last name:", "Sex:", and "Telephone:". Each label is positioned to the left of its corresponding input field. Below the input fields is a "Submit" button with a grey gradient and rounded corners. The status bar at the bottom of the browser window displays the word "Done".

Customer Registration Form

Please, fill the form to be our registered customer

ID:

First name:

Last name:

Sex:

Telephone:

Done

3.1 Database Programming...

```
<HTML>
<HEAD>
<TITLE>Processing HTML forms with PHP</TITLE>
</HEAD>
<BODY>
<H1>Newsletter sign-up form</H1>
<P> Please, fill the form to be our registered customer</P>
<FORM METHOD="post" ACTION="customer.php">
ID: <INPUT TYPE="text" NAME="ID"> <BR>
First name: <INPUT TYPE="text" NAME="FirstName"> <BR>
Last name: <INPUT TYPE="text" NAME="LastName"> <BR>
Sex: <INPUT TYPE="text" NAME="Sex"> <BR>
Telephone: <INPUT TYPE="text" NAME="Telephone"> <BR><BR>
<INPUT TYPE="submit" NAME="submit" VALUE="Submit">
</FORM>
</BODY>
</HTML>
```

The customer.php file that process the data collected by the above form is the following:

```
<?php
// Open connection to the database
$con = mysql_connect("localhost", "phpuser", "sesame") or
    die("Failure to connect to database");
mysql_select_db("customers");

$ID = $_POST["ID"];
$fn = $_POST["FirstName"];
$ln = $_POST["LastName"];
$sx = $_POST["Sex"];
$tl = $_POST["Telephone"];
$st = "1";

$query = "INSERT INTO users VALUES($ID, $fn, $ln, $sx, $tl, $st)";
$result = mysql_query($query);
if (mysql_affected_rows($result) >= 1)
    echo '<P>Your information has been saved.</P>';
else
    echo '<P>Something went wrong with your signup attempt.</P>';
?>
```

3.1 Database Programming...

- In addition to insertion, you can do many other operations on database using SQL
- *SELECT item_list FROM table_list WHERE search_condition;*
select lname, fname, balance from account where balance > 10000
- *CREATE TABLE table-name (column-name data-type, ...)*
Create table customer(lname varchar(20), fname varchar(20), branch varchar(30), balance number);
- *INSERT INTO table_name(column_names) VALUES (value_list)*
INSERT INTO account (lname, fname, branch, balance) VALUES ('john', 'smith', 101, 10000)
- *DELETE FROM table_name WHERE search_condition*
DELETE FROM account where fname = 'john' and lname = 'smith'
- *UPDATE table_name SET column_names expression WHERE condition*
*UPDATE account SET balance = balance * 1.06 where balance > 1000*

3.1 Database Programming...

Closing the connection

- Any open database connections are closed when the script ends.
- However, it is good programming practice to close the connections in the script to avoid any possible problems.
- You close database connections with a PHP function
- For example, for MySQL, use the following function to close a database connection:

```
mysql_close($connect);
```