

Part 2: JavaScript, PHP, SQL, Advanced PHP

EE4717/IM4717 Web Application Design MySQL

Lecturer :

Dr ANG Yew Hock

E-mail: iyhang@ntu.edu.sg

Tel: 67906361



A PDF file is available for printing purpose.

No re-distribution and upload of the teaching slides, supplementary materials and recorded multimedia presentations to any publicly accessible media platform and websites.

PHP+MySQL

- Many websites on the internet are implemented using PHP and MySQL.

- Recommended Book:

Title : PHP and MySQL Web development – 4th ed.

Authors: Luke Welling, Laura Thomson.

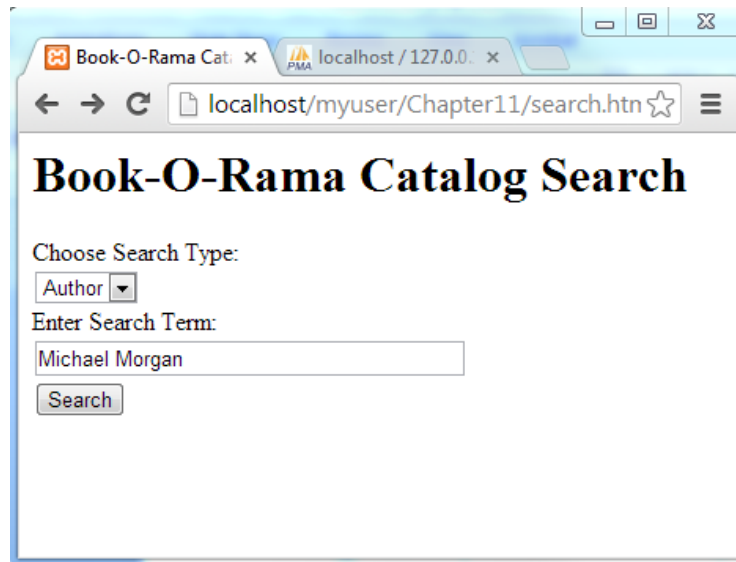
ISBN 978-0-672-32916-6

QA76.73.P224W45 2008

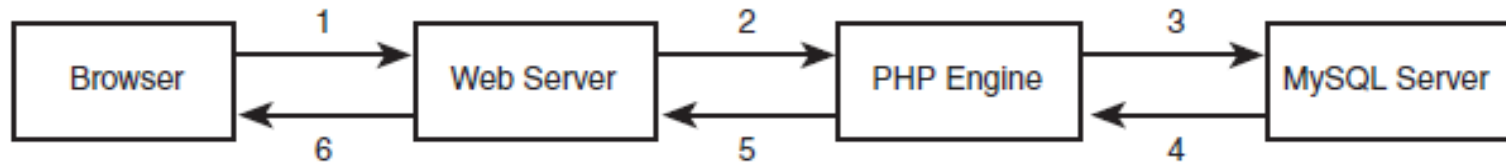
Published by Addison Wesley.

Bookshop @ Book-O-Rama

- Database is required to support **dynamic web content**
 - Data in Bob's Auto Parts is static and one-dimensional
- Bookshop @ Book-O-Rama
 - Uses RDBMS (such as MySQL) provides fast and efficient access and manipulation of databases
 - Allow dynamic search on author, title, and ISBN of books



Web Database Architecture



1. A user's browser issues an **HTTP request** for a particular web page.
 - For example, the user requested a search for all books written by Michael Morgan at Book-O-Rama, using an HTML form. The search results page is called results.php
2. The web server receives the **request for results.php**
 - It retrieves the file, and passes it to the PHP engine for processing.
3. The PHP engine begins **parsing the script**.
 - Inside the script is a command to connect to the database and execute a query (perform the search for books).
 - PHP opens a **connection to the MySQL server** and sends on the appropriate query.
4. The MySQL **server receives the database query**, processes it, and sends the results—a list of books—back to the PHP engine.
5. The PHP engine finishes running the script.
 - This usually involves **formatting the query results nicely in HTML**. It then returns the resulting HTML to the web server.
6. The web **server passes the HTML back to the browser**, where the user can see the list of books requested.

Structured Query Language (SQL)

- **Widely used** database language to store data to and retrieve it from a database
- An **ANSI standard** language used in almost all modern Relational Database Management System (RDBMS)
- It is used in database systems such as MySQL, Oracle, Sybase, and Microsoft SQL Server, among others.
 - Some differences between **ANSI SQL** and **MySQL's SQL**, but **mostly compatible**

Relational Database Concepts - Tables

➤ Tables

- Relational databases are made up of relations, called tables.

CUSTOMERS

| Customer ID | Name | Address | City |
|-------------|-----------------|--------------------|--------------|
| 1 | Julie Smith | 25 Oak Street | Airport West |
| 2 | Alan Wong | 1/47 Haines Avenue | Box Hill |
| 3 | Michelle Arthur | 357 North Road | Yarraville |

Book-O-Rama's customer details are stored in a table.

➤ Columns

- Each column in the table has a unique name and contains different data. Columns are sometimes **called fields or attributes**.

➤ Rows

- Each row in the table represents a different customer. Rows are also **called records or tuples**.

➤ Values

- Each row consists a set of individual values that correspond to columns. Each value must have the **data type specified by its column**.

Relational Database Concepts - Keys

➤ Keys

- A way of identifying each specific customer. The **identifying column** in a table is called the **key** or the **primary key**.
- A key can also consist of multiple columns.
- Databases usually consist of multiple tables and **use a key as a reference from one table to another**.
- The primary key when appeared in another table is called the **foreign key**.
- Below shows a second table added to the database.

CUSTOMERS

| Customer ID | Name | Address | City |
|-------------|-----------------|--------------------|--------------|
| 1 | Julie Smith | 25 Oak Street | Airport West |
| 2 | Alan Wong | 1/47 Haines Avenue | Box Hill |
| 3 | Michelle Arthur | 357 North Road | Yarraville |

Primary key

ORDERS

| OrderID | CustomerID | Amount | Date |
|---------|------------|--------|-------------|
| 1 | 3 | 27.50 | 02-Apr-2007 |
| 2 | 1 | 12.99 | 15-Apr-2007 |
| 3 | 2 | 74.00 | 19-Apr-2007 |
| 4 | 3 | 6.99 | 01-May-2007 |

Foreign key

Each order in the Orders table refers to a customer from the Customers table.

Relational Database Concepts - Schemas

- The database schema is a complete set of table designs for a database – a blueprint for the database.
- It shows the tables along with their columns, and the **primary** and any **foreign** keys
 - Underlined terms are primary keys and italic terms are foreign keys
- Foreign keys represent **relationship between tables**.
 - Can have one-to-one, one-to-many, and many-to-many relationship types
- Before using a database, need to select the database and create the tables

```
Customers(CustomerID, Name, Address, City)

Orders(OrderID, CustomerID, Amount, Date)

Books(ISBN, Author, Title, Price)

Order_Items(OrderID, ISBN, Quantity)

Book_Reviews(ISBN, Reviews)
```


Designing Your Web Database

- Create a database that model real-world items and relationships
 - each class of real-world objects generally needs its own table.
- In Book-O-Rama example, at least three such tables:
Customers, Orders, Books

CUSTOMERS

| CustomerID | Name | Address | City |
|------------|-----------------|--------------------|--------------|
| 1 | Julie Smith | 25 Oak Street | Airport West |
| 2 | Alan Wong | 1/47 Haines Avenue | Box Hill |
| 3 | Michelle Arthur | 357 North Road | Yarraville |

ORDERS

| OrderID | CustomerID | Amount | Date |
|---------|------------|--------|-------------|
| 1 | 3 | 27.50 | 02-Apr-2007 |
| 2 | 1 | 12.99 | 15-Apr-2007 |
| 3 | 2 | 74.00 | 19-Apr-2007 |
| 4 | 3 | 6.99 | 01-May-2007 |

BOOKS

| ISBN | Author | Title | Price |
|---------------|----------------|------------------------------------|-------|
| 0-672-31697-8 | Michael Morgan | Java 2 for Professional Developers | 34.99 |
| 0-672-31745-1 | Thomas Down | Installing GNU/Linux | 24.99 |
| 0-672-31509-2 | Pruitt.et al. | Teach Yourself GIMP in 24 Hours | 24.99 |

Design Considerations(1)

- Avoid storing redundant data
 - Redundant data takes up extra space and can cause anomalies in the data (e.g. inconsistent data updates)

| OrderID | Amount | Date | CustomerID | Name | Address | City |
|---------|--------|-------------|------------|-------------|---------------|--------------|
| 12 | 199.50 | 25-Apr-2007 | 1 | Julie Smith | 25 Oak Street | Airport West |
| 13 | 43.00 | 29-Apr-2007 | 1 | Julie Smith | 25 Oak Street | Airport West |
| 14 | 15.99 | 30-Apr-2007 | 1 | Julie Smith | 25 Oak Street | Airport West |
| 15 | 23.75 | 01-May-2007 | 1 | Julie Smith | 25 Oak Street | Airport West |

- Use Atomic column values
 - Each attribute in each row stores only one thing

ORDERS

| OrderID | CustomerID | Amount | Date | Books Ordered |
|---------|------------|--------|-------------|---|
| 1 | 3 | 27.50 | 02-Apr-2007 | 0-672-31697-8 |
| 2 | 1 | 12.99 | 15-Apr-2007 | 0-672-31745-1. 0-672-31509-2 |
| 3 | 2 | 74.00 | 19-Apr-2007 | 0-672-31697-8 |
| 4 | 3 | 6.99 | 01-May-2007 | 0-672-31745-1. 0-672-31509-2. 0-672-31697-8 |

ORDER_ITEMS

| OrderID | ISBN | Quantity |
|---------|---------------|----------|
| 1 | 0-672-31697-8 | 1 |
| 2 | 0-672-31745-1 | 2 |
| 2 | 0-672-31509-2 | 1 |
| 3 | 0-672-31697-8 | 1 |
| 4 | 0-672-31745-1 | 1 |
| 4 | 0-672-31509-2 | 2 |
| 4 | 0-672-31697-8 | 1 |

use a separate table instead

Design Considerations(2)

- Choose sensible keys
 - Make sure keys chosen are unique.
- Anticipate the use of the database
 - Make sure that the database contains all the data required and appropriate links exist between tables to answer to database requests.
- Avoid designs with many empty attributes
 - It wastes storage space and causes problems dealing with null values; whether they were truly empty or were mistakes

BOOKS

| ISBN | Author | Title | Price | Review |
|---------------|----------------|------------------------------------|-------|--------|
| 0-672-31697-8 | Michael Morgan | Java 2 for Professional Developers | 34.99 | |
| 0-672-31745-1 | Thomas Down | Installing GNU/Linux | 24.99 | |
| 0-672-31509-2 | Pruitt,et al. | Teach Yourself GIMP in 24 Hours | 24.99 | |

BOOKS_REVIEWS

| ISBN | Review |
|------|--------|
| | |

use a separate table instead



Working with MySQL

➤ Two ways of working with MySQL

- Command-line interface

```
> mysql -h hostname -u username -p
```

- Web-based MySQL administration system by logging in to phpMyAdmin at

<http://192.168.56.2/phpmyadmin/>



The screenshot shows the phpMyAdmin login interface. At the top, there is a logo for phpMyAdmin featuring a stylized sailboat. Below the logo, the text "Welcome to phpMyAdmin" is displayed. The interface includes a "Language" section with a dropdown menu currently set to "English". Below this is a "Log in" section with a "Log in" button and a lock icon. The "Log in" section contains two input fields: "Username:" with the value "f35im34" and "Password:" with an empty field. A "Go" button is located at the bottom right of the login section.

Using MySQL

- MySQL command ends with a semicolon (;)
- SQL statements are not case sensitive
 - however, **database and table names are case sensitive** in MySQL
- Data Types in MySQL
 - AUTO-INCREMENT, an unique identifier value will be automatically generated if field is left blank
 - NOT NULL, attribute in the table must have a value
 - UNSIGNED, integer type with a zero or positive value
- Strings
 - Strings are single-quoted for literals and double-quoted for expression.
 - As in PHP, quotation marks are prime characters (‘ , “)
- Common **CRUD functions** of databases
 - Create, Retrieve, Update, Delete

Commonly used SQL Commands

- Some of the most important SQL commands
 - **CREATE DATABASE** - creates a new database
 - **CREATE TABLE** - creates a new table
 - **INSERT INTO** – insert (Create) new records (rows) into a table
 - **SELECT** – Retrieve a record with selected columns from a table
 - **UPDATE** – Updates (modify) values of an existing record in a table
 - **DELETE** - Deletes data from a table
 - **DROP TABLE** – drop (delete) an existing table
 - **DROP DATABASE** – drop (delte) an existing database
 - **ALTER DATABASE** - modifies a database (e.g. character set)
 - **ALTER TABLE** - modifies a table (to add, delete, or modify columns)
 - **CREATE INDEX** - creates an index (search key for fast search)
 - **DROP INDEX** - deletes an index

Creating Database using phpMyAdmin

1. Create a new database called “myuser”
 - Enter database name “myuser” under the Databases tab
2. Create the tables in “myuser” database
 - Two ways of creating tables using phpMyAdmin
 - Interactive inputs
 - Create tables (e.g. “customers”) in the database and specify the data types
 - Then, insert the data record for each table element
 - Import from text files
 - Import file containing sql CREATE TABLES commands
 - Then, Import file containing sql INSERT INTO commands

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(50) not null,
  address char(100) not null,
  city char(30) not null
);
```

```
use myuser;
insert into customers values
(3, "Julie Smith", "25 Oak Street", "Airport West"),
(4, "Alan Wong", "1/47 Haines Avenue", "Box Hill"),
(5, "Michelle Arthur", "357 North Road", "Yarraville");
```

Importing Database Tables for Book-O-Rama

```
create table customers
( customerid int unsigned not null auto_increment primary key,
  name char(50) not null,
  address char(100) not null,
  city char(30) not null
);

create table orders
( orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null,
  amount float(6,2),
  date date not null
);

create table books
( isbn char(13) not null primary key,
  author char(50),
  title char(100),
  price float(4,2)
);
```

Do it!

```
create table order_items
( orderid int unsigned not null,
  isbn char(13) not null,
  quantity tinyint unsigned,

  primary key (orderid, isbn)
);

create table book_reviews
(
  isbn char(13) not null primary key,
  review text
);
```

bookorama.sql - sql CREATE
TABLE commands for Book-O-
Rama

Customers(CustomerID, Name, Address, City)

Orders(OrderID, CustomerID, Amount, Date)

Books(ISBN, Author, Title, Price)

Order_Items(OrderID, ISBN, Quantity)

Book_Reviews(ISBN, Reviews)

Book-O-Rama
database
schema

Importing Data into Tables for Book-O-Rama

- **book_insert.sql** – sql INSERT INTO tables commands in “myuser” database

use myuser;

insert into customers values

```
(3, "Julie Smith", "25 Oak Street", "Airport West"),  
(4, "Alan Wong", "1/47 Haines Avenue", "Box Hill"),  
(5, "Michelle Arthur", "357 North Road", "Yarraville");
```

insert into orders values

```
(NULL, 3, 69.98, "2007-04-02"),  
(NULL, 1, 49.99, "2007-04-15"),  
(NULL, 2, 74.98, "2007-04-19"),  
(NULL, 3, 24.99, "2007-05-01");
```

Do it!

insert into books values

```
("0-672-31697-8", "Michael Morgan", "Java 2 for Professional Developers", 34.99),  
("0-672-31745-1", "Thomas Down", "Installing Debian GNU/Linux", 24.99),  
("0-672-31509-2", "Pruitt, et al.", "Teach Yourself GIMP in 24 Hours", 24.99),  
("0-672-31769-9", "Thomas Schenk", "Caldera OpenLinux System Administration Unleashed", 49.99);
```

insert into order_items values

```
(1, "0-672-31697-8", 2),  
(2, "0-672-31769-9", 1),  
(3, "0-672-31769-9", 1),  
(3, "0-672-31509-2", 1),  
(4, "0-672-31745-1", 3);
```

insert into book_reviews values

```
("0-672-31697-8", "Morgan's book is clearly written and goes well beyond most of the basic Java books out there.")
```

Inserting Data into Database Tables

- The usual form of an INSERT statement is

```
INSERT [INTO] table [(column1, column2, column3,...)] VALUES  
    (value1, value2, value3,...);
```

- E.g, to insert a record into Book-O-Rama's customer table, you could type
INSERT INTO customers VALUES
 (NULL, 'Julie Smith', '25 Oak Street', 'Airport West');
- To fill in only some of the columns, or to specify them in a different order, you can list the specific columns in the columns part of the statement.

```
INSERT INTO customers (name, city) VALUES  
    ('Melissa Jones', 'Nar Nar Goon North');
```

- Alternatively, the same effect with the following syntax:

```
INSERT INTO customers  
set name = 'Michael Archer',  
    address = '12 Adderley Avenue',  
    city = 'Leeton';
```

Try it!

Retrieving Data from Database Tables

- The basic form of a SELECT is

```
SELECT [options] items [INTO file_details]
FROM tables [ WHERE conditions ]
[ GROUP BY group_type ][HAVING where_definition ]
[ ORDER BY order_type ][LIMIT limit_criteria]
[PROCEDURE procname(arguments)][lock_options]
;
```

- The following query lists the contents of the name and city columns from the customers table:

```
SELECT name, city
from customers;
```

Try it!

- The next query lists all the columns from the order_items table:

```
SELECT *
from order_items;
```

Try it!

Retrieving Data with Specific Criteria

➤ Retrieving Data with Specific Criteria

```
SELECT *
```

```
from orders
```

```
where customerid = 3;
```

Try it!

- It selects all the columns from the `orders` table, but only the rows with a customer id of 3.

➤ You can test multiple criteria using the simple operators and the pattern matching syntax and combine them into more complex criteria with AND and OR. For example,

```
SELECT *
```

```
from orders
```

```
where customerid = 3 or customerid = 4;
```

Try it!

Condition Operators for WHERE Clauses

| Operator | Name (If Applicable) | Example | Description |
|-------------|-----------------------|--------------------------------|---|
| = | Equality | Customerid = 3 | Tests whether two values are equal |
| > | Greater than | amount > 60.00 | Tests whether one value is greater than another |
| < | Less than | amount < 60.00 | Tests whether one value is less than another |
| >= | Greater than or equal | amount => 60.00 | Tests whether is greater than or equal to another |
| <= | Less than or equal | amount <= 60.00 | Tests whether is less than or equal to another |
| != or <> | Not equal | quantity != 0 | Tests whether two values are not equal |
| IS NOT NULL | n/a | address is not null | Tests whether a field actually contains a value |
| IS NULL | n/a | address is null | Tests whether a field does not contain a value |
| BETWEEN | n/a | amount between 0 and 60.00 | Tests whether a value is greater than or equal to a minimum value and less than or equal to a maximum value |
| IN | n/a | city in ("Carlton", "Moe") | Tests whether a value is in a particular set |
| NOT IN | n/a | city not in ("Carlton", "Moe") | Tests whether a value is not in a set |
| LIKE | Pattern match | name like("Fred %") | Checks whether a value matches a pattern using simple SQL pattern matching |
| NOT LIKE | Pattern match | name not like | Checks whether a value doesn't match a pattern |
| REGEXP | Pattern match | name regexp | Checks whether a value match a regular expression |

Retrieving Data from Joining Two Tables

- A *join* operation is required to retrieve data from joining two tables

- To see the orders that customer, Julie Smith has placed

```
select orders.orderid, orders.amount, orders.date  
from customers, orders  
where customers.name = 'Julie Smith'  
and customers.customerid = orders.customerid;
```

Try it!

- the *equi-join condition* in the WHERE clause matches up the rows in both tables is

```
customers.customerid = orders.customerid
```

- The **dot notation** used to make it clear which table a particular column comes from
 - **dot notation is required only** if the name of a column is ambiguous — that is, **if it occurs in more than one table**.

Try it!

Retrieving Data in a Particular Order

- The ORDER BY clause sorts the rows on one or more of the columns listed in the SELECT clause. For example,

```
SELECT name, address  
from customers  
order by name
```

- The default ordering is ascending (a to z or numerically upward). Similar effect using the ABC keyword:

```
SELECT name, address  
from customers  
order by name asc;
```

Try it!

- You can also do it in the opposite order by using the DESC (descending) keyword:

```
SELECT name, address  
from customers  
order by name desc;
```

Try it!

Grouping and Aggregating Data

➤ Aggregate Functions in MySQL

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

Aggregating Data in a Specific Column

- To calculate the average value of the amount column in the orders table:

```
select avg(amount) from orders;
```

Try it!

- The aggregate functions can also be applied to a column or to groups of data within a table.

```
select customerid, avg(amount) from orders  
group by customerid;
```

Try it!

- ❖ The GROUP BY clause changes the behavior of the aggregate function.
- ❖ Instead of giving the average value of an entire column across the table, it gives the average value for each customer (customerid) within a table

Updating Records in the Database

- The usual form of an UPDATE statement is

```
UPDATE [LOW_PRIORITY] [IGNORE] tablename  
SET column1=expression1, column2=expression2,...  
[WHERE condition]  
[ORDER BY order_criteria]  
[LIMIT number]
```

- Examples:

```
UPDATE books  
set price = price*1.1;
```

Try it!

```
UPDATE customers  
set address = '250 Olsens Road'  
where customerid = 4;
```

Try it!

Deleting Records from the Database

- The usual form of an DELETE statement is

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM table  
[WHERE condition]  
[ORDER_BY order_cols]  
[LIMIT number]
```

- Example:

```
DELETE from customers  
where customerid=6;
```

Try it!

Querying a Database from the Web

- In any script used to access a database from the Web, you follow some basic steps:
 1. Check the filter data coming from the user.
 2. Set up a connection to the appropriate database.
 3. Query the database.
 4. Retrieve the results
 5. Present the results back to the user

Steps in PHP with MySQL for insert

➤ Form in html:

```
<form action="insert_book.php" method="post">
```

```
.....
```

```
...
```

```
<input type="submit" value="Register"> ...
```

```
</form>
```

Steps in PHP with MySQL for insert

➤ In PHP server script

```
<?php // insert_book.php
```

```
    $isbn=$_POST['isbn']; // create short variable names
```

```
    .....
```

```
    if (!$isbn || !$author || !$title || !$price) {.....exit; } // check inputs
```

```
    if (!get_magic_quotes_gpc()) { .... } // filter input data
```

```
    @ $db = new mysqli('localhost', 'root', '', 'books'); // connect to db
```

```
    if (mysqli_connect_errno()) {..... exit; }
```

```
    $query = "insert into books values (' ".$isbn."', ".$author.",  
        ".$title.", ".$price.")"; // query formulation
```

```
    $result = $db->query($query); // query submission
```

```
    if ($result) { } else { ...} // retrieve query results
```

```
    $db->close();
```

```
?>
```

Steps in PHP with MySQL for search

➤ Form in HTML

```
<form action="results.php" method="post">
```

```
...
```

```
<input type="submit" name="submit" value="Search">
```

```
</form>
```

Steps in PHP with MySQL for search(1)

➤ In PHP server script

```
<?php // results.php
```

```
$searchtype=$_POST['searchtype']; // create short variable names
```

```
.....
```

```
if (!$searchtype || !$searchterm) { .... exit; } // check inputs
```

```
if (!get_magic_quotes_gpc()) {..... } // filter input data
```

```
@ $db = new mysqli('localhost', 'root', ' ', 'books'); // connect to db
```

```
if (mysqli_connect_errno()) { ... exit; }
```

```
$query = "select * from books where ".$searchtype." like
```

```
    "%$searchterm%"; // query formulation
```

```
$result = $db->query($query); // query submission
```

```
$num_results = $result->num_rows; // retrieve query results
```


Steps in PHP with MySQL for search(2)

```
echo "<p>Number of books found: ".$num_results."</p>";
```

// presenting the results

```
for ($i=0; $i <$num_results; $i++) {  
    $row = $result->fetch_assoc();  
    echo "<p><strong>".($i+1).". Title: ";  
    echo htmlspecialchars(stripslashes($row['title']));  
    echo "</strong><br />Author: ";  
    echo stripslashes($row['author']);  
    echo "<br />ISBN: "; .....  
    echo "</p>";  
}  
$result->free();  
$db->close();  
?>
```

How it Works @ Book-O-Rama

Processing `search.html`

➤ Parsing Form in `search.html`

- Parse values for “searchtype” and “searchterm”.
- Pass action to “results.php” and \$_GET variables for “searchtype” and “searchterm”

```
<html>
<head>
  <title>Book-O-Rama Catalog Search</title>
</head>

<body>
  <h1>Book-O-Rama Catalog Search</h1>

  <form action="results.php" method="get">
    Choose Search Type:<br />
    <select name="searchtype">
      <option value="author">Author
      <option value="title">Title
      <option value="isbn">ISBN
    </select>
    <br />
    Enter Search Term:<br />
    <input name="searchterm" type="text" size="40">
    <br />
    <input type="submit" name="submit" value="Search">
  </form>

</body>
</html>
```

Try it!

Step 1: in PHP with MySQL for Search

Dissecting results.php

➤ Checking and Filtering Input Data

- Stripping whitespace at beginning or end of user input, if any.
- This is done by applying the function trim() to \$searchterm
- Verify that the user has entered a search term and selected a search type
- Test get_magic_quotes_gpc() to ensure back slashes are automatically done to escape user input data using addslashes()

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  // create short variable names
  $searchtype=$_GET['searchtype'];
  $searchterm=trim($_GET['searchterm']);

  if (!$searchtype || !$searchterm) {
    echo 'You have not entered search details. Please go back and try again.';
    exit;
  }

  if (!get_magic_quotes_gpc()){ //default is add slashes to get, post, cookie
    $searchtype = addslashes($searchtype);
    $searchterm = addslashes($searchterm);
  }
```

Step 2

➤ Setting up a Connection

- Connect to MySQL server as configured; e.g. a root user on a localhost with no password set
- The connection is setup to use the database called “myuser”

```
@ $db = new mysqli('localhost','root',' ','myuser');
```

- Or, a procedural approach...

```
@ $db = mysqli_connect('localhost','root','','myuser');
```

- `mysqli_connect_errno()` returns an error number on error, or zero on success.

➤ Choosing a Database to Use

- May change the default with

```
$db->select_db(dbname)
```

- Or as.....

```
mysqli_select_db(db_resource, db_name)
```

```
@ $db = new mysqli('localhost', 'root', '', 'myuser');

if (mysqli_connect_errno()) {
    echo 'Error: Could not connect to database.
    Please try again later.';
    exit;
}

$query = "select * from books where ".$searchtype.
        " like '%$searchterm%'";

$result = $db->query($query);

$num_results = $result->num_rows;
```

Step 3 and 4

➤ Querying the Database

- Formulating the query statement:
\$query = "select * from books where ".\$searchtype." like '%\$searchterm%'";
- Querying the database using class membership approach:
\$result = \$db->query(\$query);
- Or, a procedural approach...
\$result = mysqli_query(\$db, \$query);

➤ Retrieving the Query Results

- Number of rows returned is stored in num_rows of the result object
\$num_results = \$result->num_rows;
- Or, procedurally mysqli_num_rows() returns the number of rows...
\$num_results = mysqli_num_rows(\$result);

```
@ $db = new mysqli('localhost', 'root', '', 'myuser');

if (mysqli_connect_errno()) {
    echo 'Error: Could not connect to database.
    Please try again later.';
    exit;
}

$query = "select * from books where ".$searchtype."
        " like '%$searchterm%'";

$result = $db->query($query);

$num_results = $result->num_rows;
```

Step 5

➤ Presenting the Query Results

- To process the query results, extract the elements of each row in the \$result object using associative keys

```
For ($i=0;$i<$num_results;$i++) {
    $row = $result->fetch_assoc();
    Or, procedurally...
    $row = mysqli_fetch_assoc($result);
}
```

➤ Disconnecting from the database

- Free up the result set by either
\$result->free();
Or....
mysqli_free_result(\$result);
- Close connection, if necessary
\$db->close();
Or....
mysqli_close(\$db);

```
echo "<p>Number of books found: ".$num_results."</p>";

for ($i=0; $i <$num_results; $i++) {
    $row = $result->fetch_assoc();
    echo "<p><strong>".($i+1).". Title: ";
    echo htmlspecialchars(stripslashes($row['title']));
    echo "</strong><br />Author: ";
    echo stripslashes($row['author']);
    echo "<br />ISBN: ";
    echo stripslashes($row['isbn']);
    echo "<br />Price: ";
    echo stripslashes($row['price']);
    echo "</p>";
}

$result->free();
$db->close();
```

Putting New Information in the Database

Processing `newbook.html`

- A basic HTML form for putting new books into the database

Try it!

```
<html>
<head>
  <title>Book-O-Rama - New Book Entry</title>
</head>

<body>
  <h1>Book-O-Rama - New Book Entry</h1>

  <form action="insert_book.php" method="post">
    <table border="0">
      <tr>
        <td>ISBN</td>
        <td><input type="text" name="isbn" maxlength="13" size="13"></td>
      </tr>
      <tr>
        <td>Author</td>
        <td><input type="text" name="author" maxlength="30" size="30"></td>
      </tr>
      <tr>
        <td>Title</td>
        <td><input type="text" name="title" maxlength="60" size="30"></td>
      </tr>
      <tr>
        <td>Price $</td>
        <td><input type="text" name="price" maxlength="7" size="7"></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" value="Register"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Step 1: in PHP with MySQL for Insert

Dissecting `insert_book.php`

- PHP for inserting new books into the database
- **Checking and Filtering Input Data**
 - Creating of local short variables using the POST method
 - Verify that the user has entered a search term and selected a search type
 - Test `get_magic_quotes_gpc()` to ensure back slashes are automatically done to escape user input data using `addslashes()`

```
<html>
<head>
  <title>Book-O-Rama Book Entry Results</title>
</head>
<body>
<h1>Book-O-Rama Book Entry Results</h1>
<?php
  // create short variable names
  $isbn=$_POST['isbn'];
  $author=$_POST['author'];
  $title=$_POST['title'];
  $price=$_POST['price'];

  if (!$isbn || !$author || !$title || !$price) {
    echo "You have not entered all the required details.<br />"
      . "Please go back and try again.";
    exit;
  }

  if (!get_magic_quotes_gpc()) {
    $isbn = addslashes($isbn);
    $author = addslashes($author);
    $title = addslashes($title);
    $price = doubleval($price);
  }
```


Step 2, 3 and 4

➤ Setting up a Connection

- Connect to MySQL server:

➤ Querying the database

- Formulating the query to insert values into the “books” table
- Querying the database using:
\$result = \$db->query(\$query);
- Or, procedurally...
\$result = mysql_query(\$db, \$query);

➤ Verify the result of the insert

- The number of records inserted
\$db->affected_rows
- Or procedurally...
mysql_affected_rows(\$result);

```
@ $db = new mysqli('localhost', 'myuser', 'xxxx', 'myuser');

if (mysqli_connect_errno()) {
    echo "Error: Could not connect to database. Please try again later.";
    exit;
}

$query = "insert into books values
        ('".$isbn."', '".$author."', '".$title."', '".$price."')";
$result = $db->query($query);

if ($result) {
    echo $db->affected_rows." book inserted into database.";
} else {
    echo "An error has occurred. The item was not added.";
}

$db->close();
?>
</body>
</html>
```

Exercise – eBookshop

- Add an input box and a submit button for updating the price of the books among the search results.
- The input box and submit button should be on the same page of Book-O-Rama search results page
- Upon submission, the new price of the book should be updated onto Book-O-Rama database and re-displayed onto the search results.
- May make use of multipurpose page.

Book-O-Rama Catalog Search

Choose Search Type:

Title ▾

Enter Search Term:

java

Search

Book-O-Rama Search Results

Number of books found: 1

1. Title: Java 2 for Professional Developers

Author: Michael Morgan

ISBN: 0-672-31697-8

Price: 34.99

Update price