CS381
Web Application Development

MySQL

These class notes are based on the material from our textbook,
Learning PHP, MySQL & JavaScript, 5th ed., by Robin Nixon

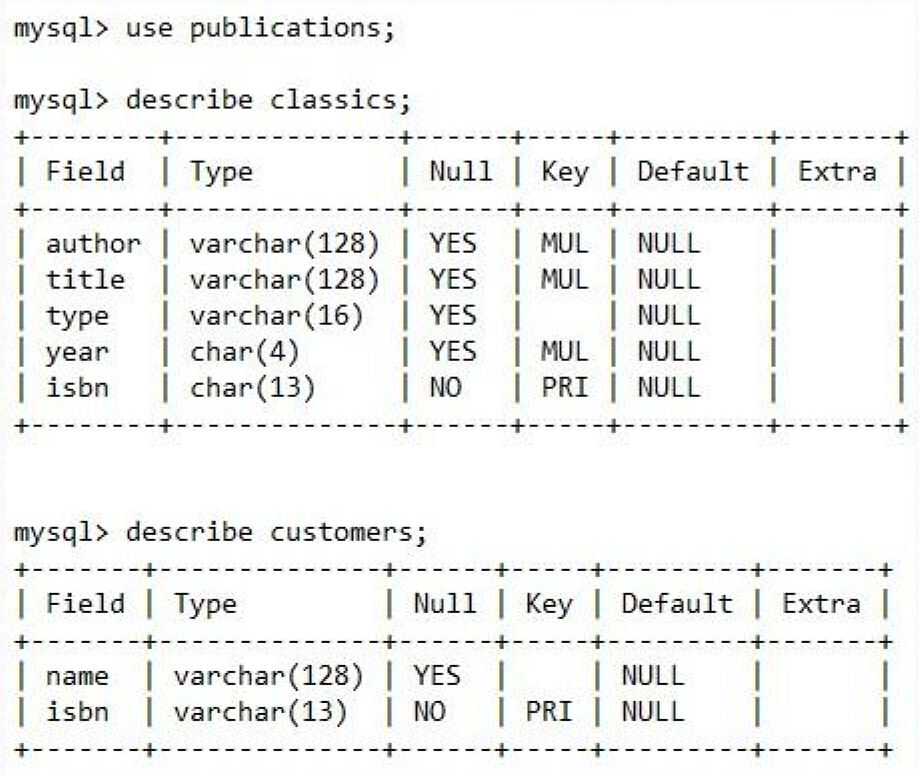
Preventing html and database Injection

Protect users' privacy from **cross-site scripting**, also referred to as an **XSS attack**., in which malicious scripts are injected into otherwise benign and trusted websites.

1. Call one of the functions, which strips out all HTML markups and replaces them with a form that displays the chars:
   a) **htmlspecialchars()** convert the special characters to HTML entities. **&, ', ", <, >**
   b) **htmlentities()** convert all applicable characters to HTML entities.

2. Use one of the following:
   a) **mysqli::real_escape_string()** protect the database so it wont run any injected code by removing special characters (**NUL, \n, \r, \, ', ", ^Z**) that may interfere with the query operations.
   b) **Placeholders** are positions within prepared statements in which data is transferred directly to the database, without the possibility of user-submitted data being interpreted as MySQL statements.

Database Used



```
mysql> use publications;

mysql> describe classics;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| author | varchar(128) | YES  | MUL | NULL    |       |
| title  | varchar(128) | YES  | MUL | NULL    |       |
| type   | varchar(16)  | YES  |     | NULL    |       |
| year   | char(4)      | YES  | MUL | NULL    |       |
| isbn   | char(13)     | NO   | PRI | NULL    |       |
+--------+--------------+------+-----+---------+-------+


mysql> describe customers;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| name  | varchar(128) | YES  |     | NULL    |       |
| isbn  | varchar(13)  | NO   | PRI | NULL    |       |
+-------+--------------+------+-----+---------+-------+
```

ConnectToMySQL.php

```php
<?php // ConnectToMySQL.php

  require_once '00 login.php';

  $conn = new mysqli($hn, $un, $pw, $db);    // MySQL Improved Extension
  if ($conn->connect_error) die("Fatal Error");


  $query  = "SELECT * FROM classics";
  $result = $conn->query($query);          if (!$result) die("Fatal Error");


  $rows = $result->num_rows;


  for ($j = 0 ; $j < $rows ; ++$j)
  {        $row = $result->fetch_array(MYSQLI_ASSOC);    // ==   fetch_assoc();      != fetch_array(MYSQLI_NUM);

           echo 'Author: '   . htmlspecialchars($row['author'])   . '<br>';
           echo 'Title: '    . htmlspecialchars($row['title'])    . '<br>';
           echo 'Category: ' . htmlspecialchars($row['type'])     . '<br>';
           echo 'Year: '     . htmlspecialchars($row['year'])     . '<br>';
           echo 'ISBN: '     . htmlspecialchars($row['isbn'])     . '<br><br>';
  }

  $result->close();                                    // close and free record set
  $conn->close();
?>
```

```php
<?php // 00 login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'jim';
  $pw = 'mypasswd';
?>
```

ConnectToMySQLAll.php

```php
<?php // ConnectToMySQLAll.php

  require_once 'login.php';

  $conn = new mysqli($hn, $un, $pw, $db);     if ($conn->connect_error) die("Fatal Error");

  $query  = "SELECT * FROM classics";
  $result = $conn->query($query);             if (!$result) die("Fatal Error");

  $rows = $result->fetch_all(MYSQLI_ASSOC);

  foreach($rows as $row)
  {
        echo 'Author: '   . htmlspecialchars($row['author'])   . '<br>';
        echo 'Title: '    . htmlspecialchars($row['title'])    . '<br>';
        echo 'Category: ' . htmlspecialchars($row['type'])     . '<br>';
        echo 'Year: '     . htmlspecialchars($row['year'])     . '<br>';
        echo 'ISBN: '     . htmlspecialchars($row['isbn'])     . '<br><br>';
  }

  $result->close();
  $conn->close();
?>
```

AddDeleteRecSql.php

```php
<?php // AddDeleteRecSql.php

  require_once '00 login.php';
  $conn = new mysqli($hn, $un, $pw, $db);            if ($conn->connect_error) die("Fatal Error");


  if (isset($_POST['delete']) && isset($_POST['isbn']))  //  isset Determines if a variable is declared and is different than NULL
  {
          $isbn   = get_post('isbn');
          $query  = "DELETE FROM classics WHERE isbn='$isbn'";

          $result = $conn->query($query);            if (!$result) echo "DELETE failed<br><br>";
  }


  if ( isset($_POST['author']) && isset($_POST['title'])  && isset($_POST['type']) && isset($_POST['year'])   && isset($_POST['isbn'])  )
  {
          $author   = get_post('author');
          $title    = get_post('title');
          $type     = get_post('type');
          $year     = get_post('year');
          $isbn     = get_post('isbn');

          $query    = "INSERT INTO classics VALUES" . "('$author', '$title', '$type', '$year', '$isbn')";

          $result   = $conn->query($query);          if (!$result) echo "INSERT failed<br><br>";
  }
```

AddDeleteRecSql.php

```
  echo <<<_END
  <form action="/PHP/AddDeleteRecSql.php" method="post"><pre>
    Author <input type="text" name="author">
     Title <input type="text" name="title">
     Type  <input type="text" name="type">
      Year <input type="text" name="year">
      ISBN <input type="text" name="isbn">
           <input type="submit" value="ADD RECORD">
  </pre></form>
_END;

  $query  = "SELECT * FROM classics";
  $result = $conn->query($query);
  if (!$result) die ("Database access failed");
  $rows = $result->num_rows;
```

```
for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = $result->fetch_array(MYSQLI_NUM);
    $r0 = htmlspecialchars($row[0]);
    $r1 = htmlspecialchars($row[1]);
    $r2 = htmlspecialchars($row[2]);
    $r3 = htmlspecialchars($row[3]);
    $r4 = htmlspecialchars($row[4]);

    echo <<<_END
  <pre>
    Author $r0
     Title $r1
      Type $r2
      Year $r3
      ISBN $r4
  </pre>
  <form action='/PHP/AddDeleteRecSql.php' method='post'>
  <input type='hidden' name='delete' value='yes'>
  <input type='hidden' name='isbn' value='$r4'>
  <input type='submit' value='DELETE RECORD'></form>
   _END;
}
```

```
  $result->close();
  $conn->close();

  function get_post($var)
  { global $conn;
    $var = $conn->real_escape_string($_POST[$var]); // escapes special characters to create a legal SQL for use in an SQL query.
    $var = strip_tags($var);                        //  Removes html and php tags
    $var = htmlentities($var);
    return $var;
  }?>
```

Placeholders

are **positions** within prepared statements in which data is transferred directly to the database, without the possibility of user-submitted data being interpreted as MySQL statements.

1. Prepare the statement.	**$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');**
2. Bind the variables.	**$stmt->bind_param('sssss', $author, $title, $type, $year, $isbn);**
3. Populate the variables.	$author='Emily';$title=Heights';$Type='Fiction';$year='1847'; $isbn= 9780553212587';
4. Execute the statement	**$stmt->execute();**
	i: integer, d: double, s: string, b: BLOB (and will be sent in packets).

Using Placeholders.php

```
<?php  // UsingPlaceholders.php
  require_once '00 login.php';
  $conn = new mysqli($hn, $un, $pw, $db);      if ($conn->connect_error) die("Fatal Error");

  $stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');
  $stmt->bind_param('sssss', $author, $title, $category, $year, $isbn);

  $author   = 'Emily Brontë';
  $title    = 'Wuthering Heights';
  $category = 'Classic Fiction';
  $year     = '1847';
  $isbn     = '9780553212587';

  $stmt->execute();
  printf("%d Row inserted.\n", $stmt->affected_rows);
  $stmt->close();
  $conn->close();
?>
```

HTTP Authentication

- uses the web server to manage users and passwords for the application.
- adequate for simple applications that ask users to log in.
- values returned in the $_SERVER should be first processed through htmlspecialchars.
- realm used for defining protection spaces (a page or more) for which the credentials are used.
- WWW-Authenticate header defines the authentication method that should be used.
- WWW-Authenticate header is sent along with a 401 Unauthorized response header.
  - if the user fills out the fields, the PHP program runs again from the top.
  - if the user clicks Cancel, the program proceeds to send the header: HTTP/1.0 401 Unauthorized.

HttpAuthentication.php

```php
<?php    // Send an HTTP header to redirect
  header("Location: http://www.google.com/");
  exit;
?>
```

```php
<?php // HttpAuthentication.php
  $username = 'jim';
  $password = 'mypasswd';
  $realm = 'My Restricted area';

  if (isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER['PHP_AUTH_PW']))
  {
    if ($_SERVER['PHP_AUTH_USER'] === $username && $_SERVER['PHP_AUTH_PW']   === $password)
          echo "You are now logged in";
    else die("Invalid username/password combination");
  }
  else
  {
    header('WWW-Authenticate: Basic realm="My Restricted Area"');
    header('HTTP/1.0 401 Unauthorized');
    die ("Please enter your username and password");
  }
?>
```

Storing Passwords

password_hash() selects a random salt for every password. It is better than using MD5 or SHA-1 hashing algorithm.

```
echo password_hash("mypassword", PASSWORD_DEFAULT);
```

PASSWORD_DEFAULT

- Selects the most secure hashing function currently available.
- It is recommend to store hashes in a database field that can expand to at least 255 characters, because the returned hash will expand in size over time as better security is implemented.

PASSWORD_BCRYPT

- Uses the BlowFish algorithm; to guarantee a hash string of only 60 characters.

password_verify() verifies that a password matches a hash.

```
if ( password_verify("mypassword", $hash) )   echo "Valid";
```

SetupUsers.php

```
<?php           // SetupUsers.php
 require_once '00 login.php';
 $conn = new mysqli($hn, $un, $pw, $db);              if ($conn->connect_error)  die("Fatal Error");


 $query = "CREATE TABLE users (  forename VARCHAR(32) NOT NULL, surname   VARCHAR(32) NOT NULL,
                                 username VARCHAR(32) NOT NULL UNIQUE,   password VARCHAR(255) NOT NULL  )";


 $result = $conn->query($query);            if (!$result) die("Could not create table");


 add_user('Bill', 'Smith', 'bsmith', 'mypasswd');

 add_user('Pauline', 'Jones', 'pjones', 'acrobat');


 function add_user($fn, $sn, $un, $pw)
 {               global $conn;

                 $hash = password_hash($pw, PASSWORD_DEFAULT);
                 $stmt = $conn->prepare('INSERT INTO users VALUES(?,?,?,?)');
                 $stmt->bind_param('ssss', $fn, $sn, $un, $hash);
                 $stmt->execute();
                 $stmt->close();
 }

?>
```

UserAuthenticate.php

```
<?php  // UserAuthenticate.php
  require_once 'login.php';
  $realm = 'Restricted area';
  $conn = new mysqli($hn,  $un,  $pw, $db);                    if ($conn->connect_error)  die("Fatal Error");

  if (   isset($_SERVER['PHP_AUTH_USER'])   ) {
                 $un_temp = fix($_SERVER['PHP_AUTH_USER']);
                 $pw_temp = fix($_SERVER['PHP_AUTH_PW']);
                 $query   = "SELECT * FROM users WHERE username='$un_temp'";        $result  = $conn->query($query);

                 if (!$result)   die("User not found");
                 elseif ( $result->num_rows ) {
                               $row = $result->fetch_array(MYSQLI_NUM);
                               $result->close();
                               if ( password_verify($pw_temp, $row[3]))              echo htmlspecialchars("Hi $row[0] $row[1]");
                               else                                                  die("Invalid  username/password combination");
                 }
                 else die("Invalid  username/password combination");
  }
  else {
                 header('WWW-Authenticate:  Basic realm="Restricted Area"');
                 header('HTTP/1.0  401 Unauthorized');     die ("Please enter  your  username  and password");
  }

  $conn>close();

  function  fix($string)  {   global $conn;       return  htmlentities(  $conn->real_escape_string($string)  );  }

?>
```

Cookies

Cookies are exchanged during the transfer of headers, before the actual HTML of a web page is sent.

setcookie(name, value, expire, path, domain, secure, httponly);

name	The name of the cookie	**location**
value	cookie's contents. This can contain up to 4 KB	**USA**
expire	(Optional) If not set, the cookie expires when the browser closes	**time() + 60*60*24**
path	(Optional) path of the cookie on the server.	**/**
domain	(Optional) The internet domain of the cookie.	**webserver.com**
secure	(Optional) Whether the cookie must use a secure connection (https://)	**TRUE**
Httponly	(Optional) If TRUE then JavaScript cannot access the cookie.	**FALSE**

Create a cookie with the name *location* and the value *USA* that is accessible across the entire web server on the current domain, and will be removed from the browser's cache in seven days

Issuing	`setcookie('location', 'USA', time() +60*60*24*7, '/');`
Accessing	`if (isset($_COOKIE['location'])) $location = $_COOKIE['location'];`
Destroying	`setcookie('location', '', time() - 2000, '/');          // or pass FALSE as a value with no time`

/ cookie is available over the entire domain, such as webserver.com. If it is a subdirectory, the cookie is available only within that subdirectory.

AuthenticateWithSession.php

```php
<?php // AuthenticateWithSession.php
   require_once '00 login.php';
   require_once 'Sessions.php';

   $conn = new mysqli($hn, $un, $pw, $db);            if ($conn->connect_error) die("Fatal Error");            $message="";

   if(count($_POST)>1)
   {             $un_temp = fix($_POST['userName']);
                 $pw_temp = fix($_POST['password']);
                 $query   = "SELECT * FROM users WHERE username='$un_temp'";             $result  = $conn->query($query);

                 if (!$result)           $message = "User not found";
                 elseif ($result->num_rows)
                 {                $row = $result->fetch_array(MYSQLI_NUM);                       $result->close();

                                  if (password_verify($pw_temp, $row[3]))
                                  {               $_SESSION['forename']  = $row[0];
                                                  $_SESSION['surname']   = $row[1];
                                                  $_SESSION['ip'] = $_SERVER['REMOTE_ADDR'];
                                                  echo htmlspecialchars("Hi $row[0] $row[1]");
                                                  header("Location: SessionActive.php");
                                  }
                                  $message = "Invalid username/password combination";
                 }
                 $message = "Invalid username/password combination";
   }
   $conn->close();
   function fix($string) {    global $conn;             return htmlentities( $conn->real_escape_string($string) ); }
?>
```

AuthenticateWithSession.php

```
<html><head><title>User  Login</title>
<style>
                .tlogin { border: 2px solid #6388ad; background: #c9d6e4; border-radius: 15px; }
                 td { text-align:  center; }
                .theader  { font-size:  30px; }
                .message  { color: #FF0000;  font-weight:  bold; }
                .btnSubmit  { border: 1px solid  #daeafa; background:  #22598b; padding:  10px  20px; color: #FFF; }
</style>
</head>

<body>
<form action="" method="post"  >
     <table  cellpadding="10"  cellspacing="1"  width="500"  align="center"  class="tlogin">
                <tr class="theader"><td>Login</td></tr>
                <tr><td><input type="text"  name="userName"  placeholder="User  Name"></td></tr>
                <tr><td><input type="password"  name="password"  placeholder="Password"></td></tr>
                <tr><td><input type="submit"  name="submit"  value="Submit"  class="btnSubmit"></td></tr>
                <tr><td class="message">  <?php  if($message!="")  { echo $message;  } ?>  </td></tr>
     </table>
</form>
</body></html>
```

SessionActive.php

```
<?php    // SessionActive.php


    require_once  'Sessions.php';

    if ( isset($_SESSION['ip']  )  )
    {
            if ($_SESSION['ip']  != $_SERVER['REMOTE_ADDR'])      endSession();

            $forename  = $_SESSION['forename'];
            $surname   = $_SESSION['surname'];

            echo "Welcome  back<br>";
            echo htmlspecialchars("$forename  $surname.");

            $rtime  = $timeout  + 5;
            $page  = htmlentities($_SERVER['PHP_SELF']);                  // returns  the  name  and  path  of the  current  file
            header("Refresh:  $rtime;  url=$page");
    }
    else
    {
             header("Location:  AuthenticateWithSession.php");
    }

?>
```

Sessions.php

```
<?php    // Sessions.php                    PHP allows the user to modify some of its settings in php.ini                    by    ini_set().


   $timeout  = 10;                                         // in sec
   ini_set('session.gc_maxlifetime',   $timeout);          // Sets the value of a configuration option - max life time for the session
   ini_set('session.use_only_cookies',   $timeout);        // Forcing cookie-only sessions to prevent session fixation
   session_start();                                        // Starts or resumes (read) existing session; must be called before any output

   if (isset($_SESSION['LAST_ACTIVITY'])  && ($_SERVER['REQUEST_TIME']  - $_SESSION['LAST_ACTIVITY'])  > $timeout)     endSession();


   $_SESSION['LAST_ACTIVITY']  = $_SERVER['REQUEST_TIME'];


    function  endSession()
    {      setcookie(session_name(), '', time() - 2592000, '/');
           session_unset();                                // frees all session variables currently  registered   or  $_SESSION = array();
           session_destroy();                              // destroys all of the data associated with the current session. It does not unset any of the
           session_start();                                // global variables associated with the session, or unset the session cookie.
    }


?>
```

Session fixation happens when a malicious third party obtains a valid session ID and makes the user authenticate themselves with that session ID, instead of authenticating with their own.