Database Write

# Objectives

* Create, update and delete records in a database via a web application.

## 1\_1

## Create a new contact

1. You must have completed the previous lab “Database retrieval” to continue work on this lab. Otherwise, copy the folder contact\ from solution of the previous lab to folder C:\wamp64\www\.
   1. We assume you have installed WAMP in C:\wamp64\.
   2. If you have installed WAMP in another folder, please change accordingly.

## 1\_2

## Add new function create() in ContactDAO

1. Edit ContactDAO to add a new function create that has an input parameter $contact for a Contact object. The code is given below.

|  |
| --- |
| **// Start of new function create()**  function create($contact) {  $result = true;  // connect to database  $connMgr = new ConnectionManager();  $conn = $connMgr->connect();    // prepare insert  $sql = "INSERT INTO `contact`(`email`, `name`, `tel`, `address`) VALUES (:email, :name, :tel, :address)";  $stmt = $conn->prepare($sql);  $stmt->bindParam(":email", $contact->getEmail(), PDO::PARAM\_STR);  $stmt->bindParam(":name", $contact->getName(), PDO::PARAM\_STR);  $stmt->bindParam(":tel", $contact->getTel(), PDO::PARAM\_STR);  $stmt->bindParam(":address", $contact->getAddress(), PDO::PARAM\_STR);  $result = $stmt->execute();  if (! $result ){ // encountered error  $parameters = [ "contact" => $contact, ];  $connMgr->handleError( $stmt, $sql, $parameters );  }    // close connections  $stmt = null;  $conn = null;    return $result;  }  **// End of new function create()** |

Figure 1: New function create().

## 1\_3

## Update the web pages

1. At the top of index.php, add a hyperlink named “New contact” to create.php .
2. Copy the given create.php from resource to your folder contact\. This page has a form that allows user to provide new contact details and posts to process\_create.php.

|  |
| --- |
|  |

Figure 2: create.php allows user to provide a new contact details.

1. Create the file process\_create.php. This will process the submitted form value and create a new record in database using ContactDAO.
2. If the new record is added successfully, process\_create.php will redirect back to index.php.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **S/N** | **Email** | **Name** | **Action** | | 1 | ace@gmail.com | Ace Ang | view | | 2 | bay@live.com | Bay Ming Chun | view | | 3 | chan@smu.edu.sg | Chan Lay Keng | view | | 4 | dan@smu.edu.sg | Danny Fong | view | | 5 | ellen@gmail.com | Ellen Lee | view | |

Figure 3: Redirect back to index.php after a successful adding of new contact [ellen@gmail.com](mailto:ellen@gmail.com)

1. If it fails, print an error message “Failed to create new contact.”
   1. To simulate failure, you can add a typo to the SQL (e.g. wrong table name). Remember to change it back after testing.

## 2\_1

## Server-side validation

Email and name cannot be blank or empty string. Hence, we need to write code to validate the form values submitted for these two fields.

1. Edit process\_create.php to insert the following lines of code to validate email and name fields. If there is any error, it
   1. Add the list of error messages to session variable ‘errors’,
   2. Redirect to create.php to display the errors.

|  |
| --- |
| **// Start of validation code**  // email and name cannot be blank or empty string  $errors = array();  $email = trim( $\_POST["email"] );  // remove leading and trailing whitespaces  if ( $email === "" ) {  $errors[] = "Email cannot be empty or blank.";  }  $name = trim( $\_POST["name"] );  // remove leading and trailing whitespaces  if ( strlen($name) == 0 ) {  $errors[] = "Name cannot be empty or blank.";  }  // If one or more fields have validation error  if ( count($errors) > 0 ) {  // Put the error message in session variable 'errors'  $\_SESSION['errors'] = $errors;    // redirect to create.php which will display the error  header("Location: create.php");  exit();  }  // **End of validation code** |

Figure 4： Server-side validation code to be added to process\_create.php.

1. Edit common.php.
   1. As we are using session variables, we need to tell the web server to start the session by calling the predefined function session\_start().
   2. Secondly, we define a new function printErrors() that
      1. List out the errors based on session variable ‘errors’.
      2. Clear the on session variable ‘errors’.

|  |
| --- |
| **// Start of code to be added to common.php**  /\*\*\*  session related stuff  \*\*\*/  session\_start();  /\*\*\*  print errors based on session variable 'errors'  \*\*\*/  function printErrors() {  if(isset($\_SESSION['errors'])){    echo "<ul style='color:red;'>";    foreach ($\_SESSION['errors'] as $error) {  echo "<li>" . $error . "</li>";  }    echo "</ul>";  // clear the session variable after use.  unset($\_SESSION['errors']);  }  }  **// End of code to be added to common.php** |

Figure 5： Session starting and error printing code added to common.php.

1. Update create.php to call the function printErrors() after the form.
   1. Remember to add require\_once 'common.php' if you have not done so.
2. Test create with empty email and/or name fields.

|  |
| --- |
|  |

Figure 3: Figure 6 Figure 6: Validation error messages displayed in create.php.

## 3\_1

## Edit a contact

1. Rename view.php to edit.php and update it to display the email (which cannot be changed), and a form that allows user to change name, tel and address.
   1. The text fields for name, tel and address should be filled with the existing values for the contact.
   2. Upon submission, post to process\_edit.php.
   3. You will need to use a hidden field for storing the email but display it as HTML text on the “edit” page. See <https://www.w3schools.com/tags/att_input_type.asp>.

|  |
| --- |
|  |

Figure 7: The renamed edit.php display a form with existing values.

1. In class ContactDAO, add a new function update() that takes in $contact which is a Contact object. Complete the function to update to database.
   1. The SQL for updating contact table is as follows:  
      UPDATE `contact` SET `name`=:name,`tel`=:tel,`address`=:address WHERE `email`=:email
   2. Use PDO prepared statement to bind the correct values.
2. Create process\_edit.php. It validates that the submitted name is not empty or blank. If there is validation error, it redirects back to edit.php which will
   1. Display an error message “Name cannot be empty or blank” after the form.
   2. Fill in the text fields with existing values.
   3. Note, edit.php is expecting a GET variable email.

|  |
| --- |
|  |

Figure 8: Validation error message displayed in edit.php.

1. If no validation error, process\_edit.php saves to database and redirect back to index.php which should display the updated values.
2. Update index.php to change all hyperlink ‘view’ to ‘edit’ and link them to the renamed edit.php.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **S/N** | **Email** | **Name** | **Action** | | 1 | ace@gmail.com | Ace Ang | edit | | 2 | bay@live.com | Bay Ming Chun | edit | | 3 | chan@smu.edu.sg | Chan Lay Keng | edit | | 4 | dan@smu.edu.sg | Danny Fong Jing Tao | edit | | 5 | ellen@gmail.com | Ellen Lee | edit | |

Figure 9: Redirect back to index.php after a successful update for [dan@smu.edu.sg](mailto:dan@smu.edu.sg). Name is updated to “Danny Fong Jing Tao”.

1. If you edit the contact again, edit.php should show the updated values.

## 4\_1

## Delete a contact

1. Update index.php to add a new hyperlink “delete” for each contact after the “edit” hyperlink.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **S/N** | **Email** | **Name** | **Action** | | 1 | ace@gmail.com | Ace Ang | edit delete | | 2 | bay@live.com | Bay Ming Chun | edit delete | | 3 | chan@smu.edu.sg | Chan Lay Keng | edit delete | | 4 | dan@smu.edu.sg | Danny Fong Jing Tao | edit delete | | 5 | ellen@gmail.com | Ellen Lee | edit delete | |

Figure 10: New “delete” hyperlink.

1. The new hyperlink “delete” links to delete.php which will display the contact details and a “Confirm delete” button.

|  |
| --- |
|  |

Figure 11: Page delete.php displays contact details and a “Confirm delete” button.

1. Upon clicking the “Confirm delete” button, it will submit to process\_delete.php.  
   Note, the email value to be deleted must be submitted.
2. Page process\_delete.php will delete the contact and redirect back to index.php.
   1. Add a new function delete($email) to ContactDAO to do this.

The SQL for deleting a record in contact table is as follows:  
DELETE from `contact` where `email`=:email  
Use PDO statements to bind the “email” value.

Side note: If a PHP file requires the use of a class definition in another PHP file, you will have to add the necessary “require\_once …” code. We will see how we can reduce this hassle later.

## 5\_1

## Important Note

Session variables require memory and disk space. Hence, avoid storing large amount of data in session variables.

Even if you have very little session variables for a single user (say 10MB), the amount of memory and disk space required will be large when you have a lot of users (like Instagram, Twitter). Hence, it is a good practice to clear session variables when you do not need them anymore

# *References*

1. <https://www.w3schools.com/tags/att_input_type.asp>