

CSCI 241

Writing to Web Databases

Department of CS & QM
Winthrop University
Fall 2011

Prepared by X. Wang

Topics

- Insert a Row into a Table
- Update the Data of a Row in a Table
- Delete a Row from a Table
- Reload Problem
- Use the `header()` function
- How to use GET and POST methods
- An example – the customer [Register](#) operation in the online bookstore

Prepared by X. Wang

2

Modifying Database

- Many web database applications are not only information resources for users but also tools for storing new information
- For example, in an online bookstore, a customer can purchase books by creating orders, register to become a member, and manage his shopping cart
- Writing data in web database applications requires different techniques from reading data
- Issues of transactions and concurrency become important

Prepared by X. Wang

3

Modifying Database

- With the INSERT, UPDATE and DELETE SQL statements, we can modify data in database tables
- Simple data modifications are similar to queries
- But it may suffer from the *reload* problem
- When the primary key of a table has MySQL's `auto_increment` constraint, an INSERT statement in a PHP page will create duplicated rows in a table (when the Refresh button is clicked in the web browser)

Prepared by X. Wang

4

Inserting Data

- Inserting is to create a new row in a table with user's input data
- Inserting data has the following steps:
 1. Data is entered by the user through an HTML `form` element
 2. The data is validated. If the validation fails, an error message is displayed (in browser) and the operation ends
 3. Otherwise, the data is written into the database using an **INSERT** statement. If the insert operation fails, another error message is displayed, otherwise a receipt page (usually the inserted data) is displayed

Prepared by X. Wang

5

Inserting Data

- The following is an HTML page (`addabook.html`) to provide a user interface to enter book information

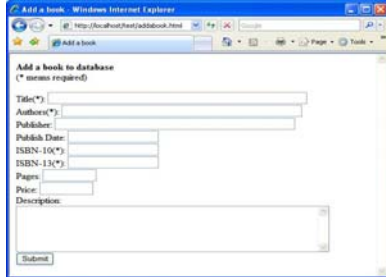
```
<body>
<b>Add a book to database </b><br />
(* means required)<br />
<form name="form1" method="POST" action="addabook.php">
Title(*): <input type="text" name="title" size="65" /><br />
Authors(*): <input type="text" name="authors" size="60" /><br />
Publisher: <input type="text" name="publisher" size="60" /><br />
Publish Date: <input type="text" name="pubdate" size="20" /><br />
ISBN-10(*): <input type="text" name="isbn10" size="20" /><br />
ISBN-13(*): <input type="text" name="isbn13" size="20" /><br />
Pages: <input type="text" name="pages" size="10" /><br />
Price: <input type="text" name="price" size="10" /><br />
Description: <br />
<textarea name="description" rows="5" cols="60"></textarea><br />
<input type="submit" value="Submit" />
</form>
</body>
```

Prepared by X. Wang

6

Inserting Data

- The HTML code creates the following user interface



Prepared by X. Wang

7

Inserting Data

- When clicking the Submit button, the entered data is sent to the page (addabook.php) on the server
- The PHP code in this page will validate the data, finish the insert operation and send a receipt page, or error messages
- The validation can also be done with JavaScript code at the client side
- One important thing is that we need to check whether the book is already in the database

Prepared by X. Wang

8

Inserting Data

```
<body>
<?php
require "db.php";

if (empty($_POST["title"])) die("The Title field cannot be blank");
$title = $_POST["title"];
if (empty($_POST["authors"])) die("The Authors field cannot be blank");
$authors = $_POST["authors"];
$publisher = $_POST["publisher"];
$pubdate = $_POST["pubdate"];
if (empty($_POST["isbn10"]) || empty($_POST["isbn13"]))
    die("Both ISBN fields cannot be blank");
$isbn10 = $_POST["isbn10"];
$isbn13 = $_POST["isbn13"];
$pages1 = $_POST["pages"]; $pages = (int)$pages1;
$price1 = $_POST["price"]; $price = (real)$price1;
$description = $_POST["description"];

if (!($connection = @ mysql_connect($hostname, $dbusername, $dbpassword)))
    die("Could not connect");
```

Prepared by X. Wang

9

Inserting Data

```
if (!(@ mysql_select_db($databasename, $connection)))
    showerror();

$query = "SELECT * FROM books WHERE ISBN='{$isbn10}' OR ISBN13='{$isbn10}'";
if (!($result = @ mysql_query ($query, $connection)))
    showerror();
$num_result = mysql_num_rows($result);
if ($num_result != 0)
    die("The book with ISBN-{$isbn10},{$isbn13} is already in the database.-br />
        The Add operation fails.");

$query = "INSERT INTO books VALUES
(NULL, '{$title}', '{$authors}', '{$isbn10}', '{$isbn13}', '{$publisher}', {$pubdate}, {$pages},
{$price}, {$description})";

if (!($result = @ mysql_query ($query, $connection)))
    showerror();
```

Prepared by X. Wang

10

Inserting Data

```
$book_id = mysql_insert_id();

print("The following items have successfully been written into database ({$book_id}):
<br /><br />");
print("Title: {$title}<br />");
print("Authors: {$authors}<br />");
print("Publisher: {$publisher}<br />");
print("Publish Date: {$pubdate}<br />");
print("ISBN-10: {$isbn10}<br />");
print("ISBN-13: {$isbn13}<br />");
print("Pages: {$pages}<br />");
print("Price: {$price}<br />");
print("Description: {$description}<br />");

?>
</body>
```

Prepared by X. Wang

11

Updating Data

- Updating (editing) data is usually a more complex process than inserting the data
- Updating data has the following steps:
 1. The user gives a key value for the data row to be updated. The key value can be typed, or selected from a pop-up menu by the user
 2. The key value is sent to a PHP page on the server. The PHP page uses the key value to retrieve the data from database, and presents the data to the web browser in an HTML form for the user to edit it

Prepared by X. Wang

12

Updating Data

- When the user finishes the modification and click the Submit button, the data is sent to another PHP page on the server for validation
- If the data passes the validation, the data will be used to update the corresponding data row in database using the **UPDATE** statement, and send a receipt message back to browser
- If the data cannot pass validation, an error message will be sent back to the user

Prepared by X. Wang

13

Updating Data

- The following is a page (`editabook.html`) to enter a book's ISBN

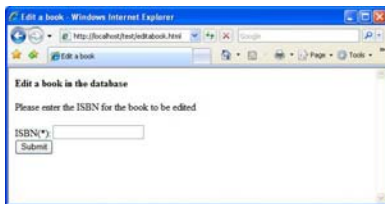
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Edit a book</title>
</head>
<body>
<b>Edit a book in the database </b><br /><br />
Please enter the ISBN for the book to be edited<br />
<form name="form1" method="GET" action="editabook.php">
ISBN(*): <input type="text" name="isbn" size="20" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Prepared by X. Wang

14

Updating Data

- The previous HTML code creates the following user interface



Prepared by X. Wang

15

Updating Data

- After clicking the **Submit** button, the entered ISBN is sent to the page (`editabook.php`) on the server
- The PHP code in this page will check whether the book is already in the database. If not, the code displays an error message.
- Otherwise, the code generates a **form** using the book data from database, and send the form to the browser for user to modify
- Please notice that two ISBN fields are disabled in the form

Prepared by X. Wang

16

Updating Data

- The following is the code for `editabook.php`

```
<body>
<?php
require "db.php";
if (empty($_GET["isbn"])) die("The ISBN must be entered");
$isbn = $_GET["isbn"];
if (!($connection = @ mysql_connect($hostname, $dbusername, $dbpassword)))
die("Could not connect");
if (!(@ mysql_select_db($databasename, $connection)))
showerror();

$query = "SELECT * FROM books WHERE ISBN= '{$isbn}' OR ISBN13= '{$isbn}' ";
if (!($result = @ mysql_query($query, $connection)))
showerror();
$num_result = mysql_num_rows($result);
if ($num_result == 0)
die("The book with ISBN-{$isbn} is not in the database.<br />Please enter a
correct ISBN.");
```

Prepared by X. Wang

17

Updating Data

```
$row = @ mysql_fetch_array($result);

print "Edit the following book data items:<br />";
print "(" means required)<br /><br />";
print "<form name='form1' method='POST' action='updateabook.php'>";
print "Title(*): <input type='text' name='title' size='65' value='{$row['title']}' /><br />";
print "Authors(*): <input type='text' name='authors' size='60'
value='{$row['authors']}' /><br />";
print "Publisher: <input type='text' name='publisher' size='60'
value='{$row['publisher']}' /><br />";
print "Publish Date: <input type='text' name='pubdate' size='20'
value='{$row['publishdate']}' /><br />";
print "ISBN-10(*): <input type='text' name='isbn10' size='20'
value='{$row['ISBN']}' disabled='disabled' /><br />";
print "ISBN-13(*): <input type='text' name='isbn13' size='20'
value='{$row['ISBN13']}' disabled='disabled' /><br />";
print "<input type='hidden' name='isbn' value='{$row['ISBN']}' />";
```

Prepared by X. Wang

18

Updating Data

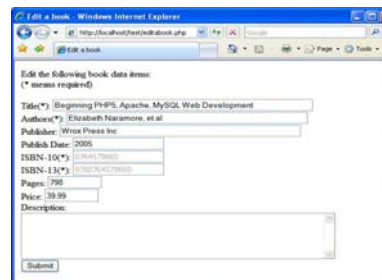
```
print "Pages: <input type='text' name='pages' size='10' value='{$row['pages']}'\n
/><br />";
print "Price: <input type='text' name='price' size='10' value='{$row['price']}' /><br
/>";
print "Description: <br />";
print "<textarea name='description' rows='5'
cols='60'>{$row['description']}</textarea><br />";
print "<input type='submit' value='Submit' />";
print "</form>";
?>
</body>
```

Prepared by X. Wang

19

Updating Data

- The previous HTML code creates the following form



Prepared by X. Wang

20

Updating Data

- Then the user can modify the book data displayed in the form
- After finishing the modification, clicking the Submit button will send modified book data to the page (updatebook.php) on the server
- The PHP code in this page will modify the corresponding row in the books table using the UPDATE statement
- Finally, the page sends a receipt page back to the browser
- The following is the code for updatepage.php

Prepared by X. Wang

21

Updating Data

```
<body>
<?php
require "db.php";
if (empty($_POST["title"])) die("The Title field cannot be blank");
$title = $_POST["title"];
if (empty($_POST["authors"])) die("The Authors field cannot be blank");
$authors = $_POST["authors"];
$publisher = $_POST["publisher"];
$pubdate = $_POST["pubdate"];
if (empty($_POST["isbn"])) die("No ISBN is specified");
$isbn = $_POST["isbn"];
$page1 = $_POST["pages"]; $pages = (int)$page1;
$price1 = $_POST["price"]; $price = (real)$price1;
$description = $_POST["description"];

if (!($connection = @ mysql_connect($hostname, $username, $password)))
die("Could not connect");
if (!(@ mysql_select_db($database, $connection)))
showerror();
```

Prepared by X. Wang

22

Updating Data

```
$query = "UPDATE books SET title='{$title}', authors='{$authors}',
publisher='{$publisher}', publishdate='{$pubdate}', page='{$pages}',
price='{$price}', description='{$description}'
WHERE ISBN='{$isbn}";
if (!($result = @ mysql_query ($query, $connection)))
showerror();

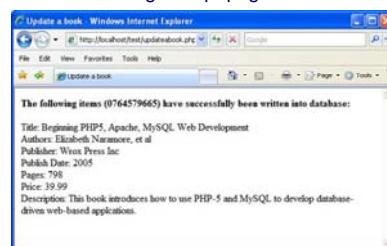
print("<b>The following items ({$insb}) have successfully been written to database:</b>\n
<br /><br />");
print("Title: {$title}<br />");
print("Authors: {$authors}<br />");
print("Publisher: {$publisher}<br />");
print("Publish Date: {$pubdate}<br />");
print("Pages: {$pages}<br />");
print("Price: {$price}<br />");
print("Description: {$description}<br />");
?>
</body>
```

Prepared by X. Wang

23

Updating Data

- When submitting the modification, the previous code sends the following receipt page



Prepared by X. Wang

24

Deleting Data

- Deleting data is a simple operation. It will remove a data row from database
- Deleting data has the following steps:
 1. The user gives a key value for the data row to be deleted. The key value can be typed, or selected from a pop-up menu by the user
 2. The key value is sent to the PHP page on the server. The PHP page uses the key value to check whether the data row is in the database. If yes, the page should send a confirmation page to the browser, and then the user can click **Yes** button to confirm it or click **Cancel** button to cancel the operation

Prepared by X. Wang

25

Deleting Data

- The following is a page ([deleteabook.html](#)) to enter an ISBN for the book to be deleted

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Delete a book</title>
</head>
<body>
<b>Delete a book from the database </b><br /><br />
Please enter the ISBN for the book to be deleted<br />
<form name="form1" method="POST" action="deleteabook.php">
ISBN(*): <input type="text" name="isbn" size="20" /><br />
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

Prepared by X. Wang

26

Deleting Data

- The previous HTML code creates the following user interface



Prepared by X. Wang

27

Deleting Data

- When clicking the **Submit** button, the entered ISBN is sent to the page ([deleteabook.php](#)) on the server
- The PHP code in the page will check whether the book is already in the database. If not, the code displays an error message.
- Otherwise, the code generates a form with the two radio buttons (**YES** and **Cancel**), and a **Submit** button, and send the form to the browser for confirmation
- The following is the code for [deleteabook.php](#)

Prepared by X. Wang

28

Deleting Data

```
<?php
require "db.php";

if (empty($_POST["isbn"])) die("The ISBN must be entered");
$isbn = $_POST["isbn"];

if (!($connection = @ mysql_connect($hostname, $dbusername, $dbpassword)))
die("Could not connect");
if (!(@ mysql_select_db($database, $connection)))
showerror();

$query = "SELECT * FROM books WHERE ISBN=('$isbn') OR ISBN13=('$isbn)";
if (!($result = @ mysql_query ($query, $connection)))
showerror();
$num_result = mysql_num_rows($result);
if ($num_result == 0)
die("The book with ISBN-{$isbn} is not in the database.<br /> Please enter a correct ISBN.");
```

Prepared by X. Wang

29

Deleting Data

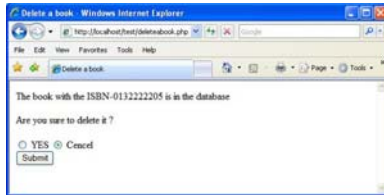
```
print "The book with the ISBN-{$isbn} is in the database.<br /><br />";
print "Are you sure to delete it ?<br />";
print "<form name='form1' method='POST' action='confirmdelete.php'>";
print "<input type='radio' name='delete' value='yes' /> YES ";
print "<input type='radio' name='delete' value='cancel' checked='checked' />";
print "Cancel<br />";
print "<input type='hidden' name='isbn' value='{$isbn}' /> ";
print "<input type='submit' value='Submit' /> ";
print "</form>";
?>
```

Prepared by X. Wang

30

Deleting Data

- The previous PHP code creates the following form



Prepared by X. Wang

31

Deleting Data

- Then the user can choose to delete the book or cancel the operation by checking different radio button
- When clicking the **Submit** button, the confirmation is send to the page ([confirmadelete.php](#)) on the server
- The PHP code in this page will delete the corresponding row from the books table using the **DELETE** statement
- Finally, the page sends a receipt page back to the browser
- The following is the code for [confirmadelete.php](#)

Prepared by X. Wang

32

Deleting Data

```
<body>
<?php
require "db.php";
if (empty($_POST["isbn"])) die("The ISBN must be entered");
$isbn = $_POST["isbn"];
if (isset($_POST["delete"]) || $_POST["delete"]=="yes") {
    print "The DELETE operation has been cancelled!";
    exit;
}
if (!($connection = @ mysql_connect($hostname, $dbusername, $dbpassword)))
    die("Could not connect");
if (!(@ mysql_select_db($databasename, $connection)))
    showerror();
$query = "DELETE FROM books WHERE ISBN=('$isbn') OR ISBN13=('$isbn')";
if (!($result = @ mysql_query($query, $connection))) showerror();
$num_affected = mysql_affected_rows();
print "$num_affected book with the ISBN-{$isbn} has been deleted from the database";
?>
</body>
```

Prepared by X. Wang

33

Reload Problem

- We have had an example to show how to **insert** a new book into the **books** table
- The slide for [addabook.php](#) shows that we check **ISBN** to see whether **the new book is already in the table before we execute the INSERT statement**
- We know that this check can avoid creating duplicated books in the books table
- Now, let's **assume** that we **remove ISBN** attribute from the **books** table
- Then we **don't have an attribute** that can be entered by a user to **explicitly** identify a row in a table

Prepared by X. Wang

34

Reload Problem

- Instead, we only have the primary key (**book_id**) to identify a book
- We know that **book_id** is an **auto_increment** attribute
- If a user clicks the **reload** (or **refresh**) button of a web browser, the same URL will be sent to the server to execute the [addabook.php](#) page again
- Because we don't have the **ISBN**, the **INSERT** statement will be executed, which creates duplicated rows in the table
- If a user keeps clicking the reload button, the same row (book) will be inserted into the table for every clicking

Prepared by X. Wang

35

Reload Problem

- This problem is called **reload problem**
- It appears in both **INSERT** and **UPDATE** operations
- For the **UPDATE** operation, it doesn't hurt the data in the database except occupying more network and database capacities
- However, for the **INSERT** operation, it will damage the consistency of a database
- We must solve or avoid this problem for the **INSERT** operation

Prepared by X. Wang

36

Reload Problem - Solution

- One way that can solve the problem is to **design an explicit key** (an attribute that is entered by the user) for each table whose data rows are created from user's input
- For example, in the **books** table, we have **ISBN** attribute. In the **customers** table, we have the **email** attribute. In the **employee** table, we have **SSN** attribute
- For the tables such as **orders** and **items**, it is fine without the explicit key since the rows in these tables are created by scripts (once a row is created, its data source will be deleted), not directly from user's input

Prepared by X. Wang

37

Reload Problem - Solution

- With an explicit key, we can **check** whether a row with the same key is already in the table **before** we do an INSERT operation
- If there is a row with the same key, we can **reject** the INSERT operation. This can completely **avoid** the reload problem
- Usually, we **can find** an explicit key attribute for the tables whose data rows are created from user's input

Prepared by X. Wang

38

Reload Problem - Solution

- If we cannot find an explicit key attribute for a table, and the data rows for the table are from user's input, then we need to use the **Relocation Technique** to **try to avoid** the duplicated insert operations
- Furthermore, even if we can find an explicit key for a table, we can still use Relocation Technique to reduce network traffic and server load
- A **Relocation Technique** is to **split the PHP page that inserts (or updates) a row and generates a receipt page into two separate PHP pages**

Prepared by X. Wang

39

Reload Problem - Solution

- The **first PHP page** does the following steps:
 - Accepts the user's input data
 - Validates the input data, and sends error message back to browser if the input data has problems
 - Inserts the data into the database table if passing the validation
 - **Jumps** to the second PHP page by sending an HTTP header back to the browser using the **header()** function

Prepared by X. Wang

40

Reload Problem - Solution

- The **second PHP page** only generates the receipt information. It doesn't modify database
- In the first page, the **header()** function call in the last step actually changed the **Location** object in the web browser
- When the user **clicks the Reload button**, the URL in the **Location** object is already changed and **points to the second page**
- This can avoid that the **Reload clicking** directly causes a duplicated insert (or update) operation

Prepared by X. Wang

41

Reload Problem - Solution

- Of course, if the user clicks the **Back** button of the browser and **re-submits** the user data, it will **still cause the Reload problem**
- So using the **header()** function call cannot completely solve the Reload problem, but it mostly reduces the possibility that Reload problem occurs
- Even if a table has an explicit key attribute, it is **better** to design an **insert or update** operation into two separate pages **using the header() function (to reduce network traffic and server load)**

Prepared by X. Wang

42

The header () Function

- The `header()` function in PHP is used to send raw HTTP header back to the browser
- A PHP script can use a header function call to jump to another web page from the current page
- The syntax to do this is

```
header("Location: URL");
```
- URL can be an absolute or relative HTTP address
- For example: `header("Location: welcome.php");` in the following code example jumps to the receipt page `welcome.php`

Prepared by X. Wang

43

The header () Function

- A `header()` function must be called **before any actual output is sent**
- If this can not be guaranteed, we need to call the `ob_start()` function to buffer the output information
- The `ob_end_flush()` function is used to send the output buffer and turn off output buffering

Prepared by X. Wang

44

How to Use GET and POST

- As we know that the user data can be sent from the browser to the server using either GET or POST method
- There are some cases that we must use GET or POST
- If a PHP page allows to be accessed using an **active hyperlink or from the address field**, the page must be designed to **use the GET method** to accept user's data
- If a PHP page accepts a **user name and a password**, we should **use the POST method** for security reason

Prepared by X. Wang

45

How to Use GET and POST

- If a PHP page accepts user's data to **insert or update a record in database**, we **usually use the POST method**
- If a PHP page accepts many data items from the user, we **usually use the POST method**
- If a PHP page accepts user's data to **conduct a query**, we can use **either GET or POST method**

Prepared by X. Wang

46

An Example to Do Insert Operation with header () function call

- The following is an example that use three PHP pages to complete the customer registration operation
- The first PHP page (`register.php`) displays an HTML form to collect a customer's information
- If we don't consider the session state (introduced in the future), this page can be written in only HTML code
- We use the PHP code to create the list for selecting a state. The names of 50 states are stored in an array in the `db.php` file

Prepared by X. Wang

47

An Example to Do Insert Operation with header () function call

- Here is the new `db.php` file

```
<?php
$hostname = "localhost";
$databasename = "wangx";
$dbusername = "wangx";
$dbpassword = "wang123";

function showerror() {
    print("Error: " . mysql_errno() . " - " . mysql_error());
}

$states = array("Alabama","Alaska","Arizona","Arkansas","California","Colorado",
"Connecticut","Delaware","Florida","Georgia","Hawaii","Idaho","Illinois","Indiana",
"Iowa","Kansas","Kentucky","Louisiana","Maine","Maryland","Massachusetts",
"Michigan","Minnesota","Mississippi","Missouri","Montana","Nebraska","Nevada",
"New Hampshire","New Jersey","New Mexico","New York","North Carolina",
"North Dakota","Ohio","Oklahoma","Oregon","Pennsylvania","Rhode Island",
"South Carolina","South Dakota","Tennessee","Texas","Utah","Vermont",
"Virginia","Washington","Washington D.C.,""West Virginia","Wisconsin",
"Wyoming");

?>
```

Prepared by X. Wang

48

An Example to Do Insert Operation with header() function call

- The register.php page generates the user interface:

Prepared by X. Wang

49

An Example to Do Insert Operation with header() function call

- Since we ask the user to enter password in the user interface, we need to use the POST method in the register.php page
- After entering a customer's information, clicking the Submit button, the customer information will be sent to the next page (suppose called saveregister.php) using the HTTP POST method

Prepared by X. Wang

50

An Example to Do Insert Operation with header() function call

- The saveregister.php page validates user data first. If a data item misses, the page needs to send an error message back to the browser
- Otherwise, the PHP code will query the customers table (via the email) and check whether the customer exists there


```
$query = "SELECT * FROM customers WHERE email={\$email}";
```
- If the customer exists in the customers table, the page should report the customer that he has an account in the system

Prepared by X. Wang

51

An Example to Do Insert Operation with header() function call

- Otherwise, the PHP code constructs an insert statement as follows to insert the customer record into the customers table


```
$query = "INSERT INTO customers VALUES (NULL, {\$lastname}, {\$firstname}, {\$email}, {\$password1}, {\$address}, {\$city}, {\$state}, {\$zipcode}, {\$phone})";
```
- If insert is successful, call mysql_insert_id(); function to read the cust_id (primary key)


```
$cust_id = mysql_insert_id();
```

Prepared by X. Wang

52

An Example to Do Insert Operation with header() function call

- To avoid the Reload problem, the saveregister.php page finally needs to call the header() function with the cust_id (implicitly use the GET method) to redirect to the third page welcome.php


```
header("Location: welcome.php?cust_id={\$cust_id}");
```
- Notice, we need to use ob_start(); at the beginning of the page to buffer the output since we call the header() function in this page

Prepared by X. Wang

53

An Example to Do Insert Operation with header() function call

- The welcome.php page accepts the cust_id from the GET array, queries the customer table to get the customer name, and generates a welcome page
- After running the pages, we can see that the page saveregister.php will never appear in the address field
- When user clicks the Reload button, the page with the insert operation (saveregister.php) will not be executed again

Prepared by X. Wang

54

