An Integrated Approach To Teaching Web Development

Karina Hauser, Utah State University, USA
David Olsen, Utah State University, USA
Kelly Fadel, Utah State University, USA

ABSTRACT

The growing use of the Internet has led to a steep increase in the demand for web developers who possess web design, database, and programming skills. The demand for these skills is reflected in the new IS2007 model curriculum, which suggests that web developers obtain in-depth knowledge of databases and web programming. While these topics are commonly covered in Information Systems (IS) curricula, they are traditionally taught independently in separate courses, leaving students with a fragmented view of how to integrate the various components of a data-driven web application. Research on learning has shown that activation of existing knowledge is an important step in the learning process. This study shows how a common business case can be used to support the activation of existing knowledge in different classes related to web development. The detailed examples show how to use the same business case to teach database, web design, and programming skills.

Keywords: Curriculum Design, Web Development, Case Study

INTRODUCTION

According to the Bureau of Labor Statistics, employment for web developers *is expected to increase much faster than the average as organizations continue to expand their use of technology*. This prediction is supported by studies from industry and academia. Academia has responded to this trend with the inclusion of web technologies and development in the IS2007 model curriculum. Data & Information and Application Development are two of the five core topics in the new IS2007 model curriculum.

Most web applications are data-driven, so web developers need to develop solid SQL and programming skills. Kung surveyed 232 IS programs and found that 204 (88%) included at least one programming course and 214 (92%) included at least one database course. Unfortunately, those courses are often taught independently of each other, leaving students with "islands of exposure" in each area, but little holistic understanding of how to combine the components into an integrated, data-driven web application. This paper shows how use of a single business case can provide a more integrated approach that benefits students' learning and provides a "big picture" approach to web development. The next section provides an overview of instructional principles and how the activation principle can be used to improve learning. Section three describes the common business case used throughout the different classes. Finally, sections four through six provide details on how the business case is used to teach web development, which includes components of a database course, a web graphical user interface (GUI) design course, and a web programming course.[1]

[1] In this paper, we assume that the database course is the first course that students take, followed by web GUI design and, finally, web programming. This seems to be a common sequence for these courses in IS curricula.

ACTIVATION OF EXISTING KNOWLEDGE

Merrill examined a wide variety of instructional theories, both constructivist and objectivist designs, and identified five common principles that he named "First Principles of Instruction". The five principles are shown in Figure 1, followed by a short description of each principle.

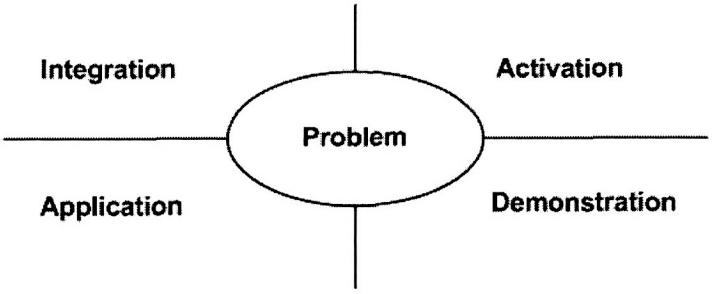

Figure 1: Merrill's First Principles of Instruction

1. Learning is facilitated when learners are engaged in solving real-world problems.
2. *Learning is facilitated when existing knowledge is activated as a foundation for new knowledge.*
3. Learning is facilitated when new knowledge is demonstrated to the learner.
4. Learning is facilitated when new knowledge is applied by the learner.
5. Learning is facilitated when new knowledge is integrated into the learner's world.

The second principle (shown in italics) emphasizes the integration of new knowledge with existing knowledge, and is referred to as the *activation principle*. Studies [see 1for complete literature review] have shown that use of the activation principle in instruction facilitates learning by situating newly acquired knowledge within the context of what learners already know, thus fostering a more complete understanding of the "big picture." However, because instructors generally have little knowledge about students' experiences/knowledge outside school, successfully applying the activation principle can be difficult. One way to overcome this challenge is through the use of integrated curricular materials in courses that teach interdependent skill sets, such as database development, web GUI design, and web programming. Because these courses are often structured in prerequisite fashion (e.g. a web GUI design is a common prerequisite for web programming) and because instructors typically have knowledge of the courses students have taken, the integration of course materials can facilitate application of the activation principle by providing a foundation of existing knowledge familiar to both students and instructors.

To facilitate the activation of existing knowledge in teaching web development, we have developed a detailed business case that can be used in different courses, including database, graphical web design, and web programming courses. While cases are commonly used in information systems classes, they are often geared towards a single course or objective. By using a common business case across courses, instructors know what students have been exposed to, which can help in developing appropriate activation questions to enhance student learning.

CURRENCY TRADING CASE

This section outlines the case study description, the entity-relationship (ER) diagram, and the use case diagram and use case description. These materials serve as a student reference for the development of the database, user interface and back-end programs.

Case Study Description

The Kanab currency trading company is a US-based currency exchange firm that wants to develop a foreign currency trading information system. Kanab has two different types of customers: corporate and person. All customers have an ID, email address, a credit limit, a current cash balance, and a credit rating that is in a range of 0 to 100 with 0 being the worst credit score and 100 being the best. Customers also have an account type which is either 'Unlimited', 'Margin' or 'Basic'. In addition to these common attributes, each type of customer has unique properties. For corporate customers, these include the corporation's name, a contact person's name, and the state of incorporation. For person customers, these include the customer's last name and a first name.

Kanab trades in foreign currencies, so the currency name and the exchange rate for a single US dollar must be captured (i.e. one US dollar might be worth 100 Japanese Yen). There are two types of transactions: buying or selling a foreign currency. For each transaction, Kanab must capture the date and the time of the transaction, the foreign currency being traded, the transaction amount in US dollars, a two percent fee, and a total dollar amount.

Entity-Relationship Diagram

Based on the description above, an ER diagram for the overall database is shown in Figure 2.

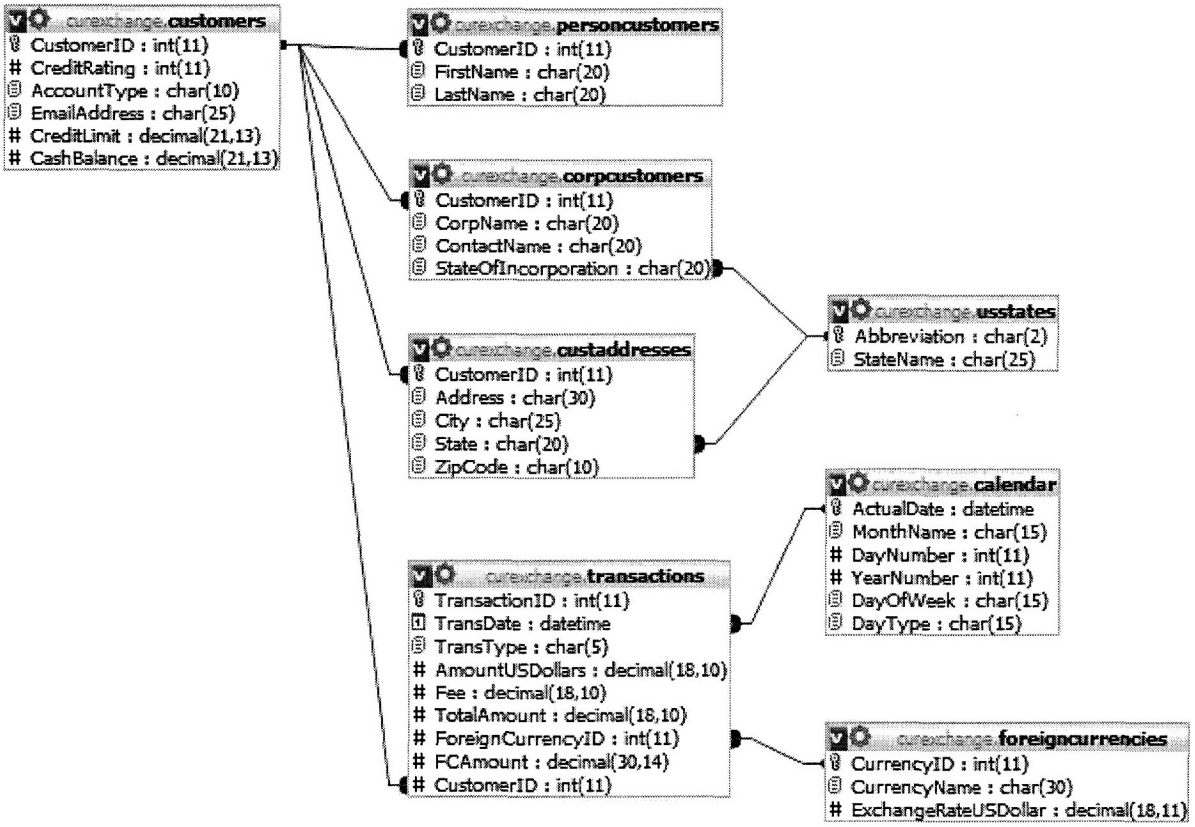

Figure 2: Entity-Relationship Diagram

Use Case Diagram

Use cases describe behavior of an information system and its interaction with users in simple, non-technical terms. The use case diagram (
Figure 3) shows the interaction between the actors (users) and the application. It gives an overview of the currency trading application (CTA) and how the different parts relate to each other.

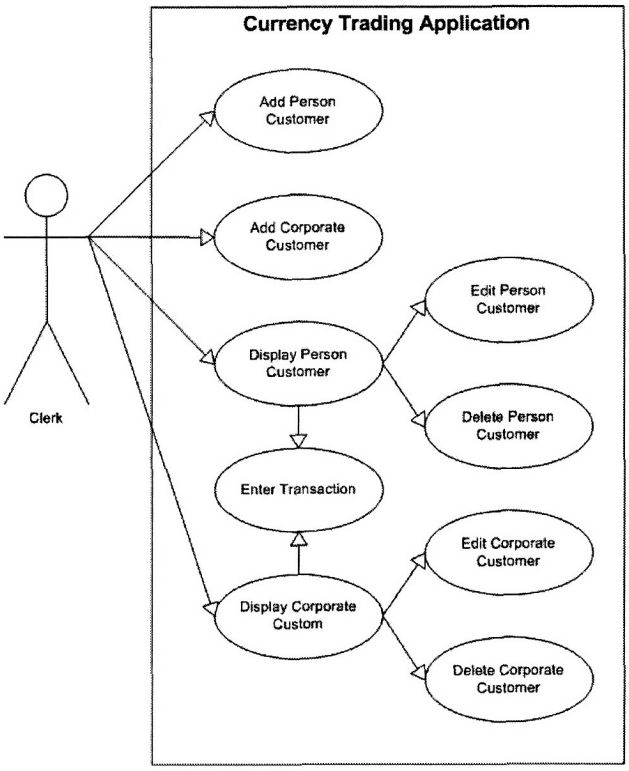

Figure 3: Use Case Diagram

Use Case Descriptions

As shown in the use case diagram nine use cases exist. For brevity reasons only the "Add Person Customer" and "Add Transaction" cases are described in detail.

3.3.1 *USE CASE 1: Add Person Customer*

Primary Actor: Company Clerk

Scope: Currency Trading Application

Level: User goal

Stakeholders and Interests:

Kanab – wants to capture customer information
Customer – wants to register to trade currency

Precondition: None

Main Success Scenario:

1. Clerk asks customer for information
2. Clerk enters information in "Add New Customer" form
3. System verifies information and information is valid
4. CTA stores customer information in database and displays all customers

Extensions:

3a1. Information is either missing or not valid
3a2. CTA sends out error message
3a3. Clerk enters missing or corrects invalid information

3.3.2 *USE CASE 5: Enter Transaction*

Primary Actor: Company Clerk

Scope: Currency Trading Application

Level: User goal

Stakeholders and Interests:

Company – wants add currency trading transaction
Customer – wants to trade currency

Precondition: Customer has been selected

Main Success Scenario:

1. Clerk selects customer to enter transaction
2. CTA displays last 5 transactions for customer
3. Clerk request buying/selling information from customer
4. Clerk enters information in "Add Transaction" form
5. CTA verifies information and information is valid
6. CTA displays foreign currency amount and transaction fees
7. Clerk notifies the customer about amount and transaction fees
8. Clerk finalizes transaction

Extensions:

5a1. Information is not valid
5a2. Clerk request valid information from customer
8a1. Clerk aborts transaction

The case can be used in one or more instructional modules. Each of the following sections defines the objectives for one specific instructional module, provides details on skills taught, and presents sample activation questions and example code.

DATABASE COURSE

The Kanab case is first introduced in the database course to help students learn database design as well as basic SQL statements including INSERT, UPDATE, SELECT, and DELETE. For example, students write SQL statements to insert new customers (person and corporate), update customer information, insert new customer transactions, and delete customers based on specific criteria. Students also use SELECT statements to create various types of reports, such as a summary of the transactions over a certain dollar amount or within a date range.

After basic SQL operations have been taught, the students are introduced to stored procedures as a convenient and robust way to carry out common data manipulation tasks. The Kanab case is again used as a foundation for teaching stored procedures, thus providing a means for activating prior student knowledge. The section below describes a portion of this instructional module focused on creating and executing a stored procedure for inserting a new person customer into the database. Included are sample activation discussion and questions that can be utilized by the instructor.

Objectives

The objectives for this instructional module are:

- Students will be able to write stored procedures
- Students will be able to execute stored procedures
- Students will discover the benefits of stored procedures
- Knowledge/Skills Taught

The following SQL commands demonstrated in example:

CREATE PROCEDURE
EXECUTE procedure

Sample Activation Discussion and Questions:

"Recall how earlier in the course we wrote an SQL INSERT statement to insert a new customer into the database for the Kanab Currency Trading Company:"

- When inserting a new customer in the database, what fields needed to be included? [*All fields in the Customers, PersonCustomers, and CustAddresses tables*]
- How many INSERT statements were required to insert all information for a person customer? [*Three: an INSERT for each of the tables listed above*]
- What problems might this present for the database user who must insert many customers into the database over time? [*Insert statements must be re-written each time, making data entry cumbersome and error-prone*]

Stored procedures provide a convenient way for storing common SQL operations within the DBMS so they can be executed repeatedly. Using stored procedures saves time because statements do not need to be re-written each time they are executed. A stored procedure can include anything from simple SELECT/UPDATE/INSERT/DELETE statements to more complex operations involving advanced DBMS features. In addition, because they are an integrated part of the DBMS, stored procedures are can be optimized by the DBMS to run in the most efficient way possible."

Code

Listings 1 and 2 show the SQL Code for creating the stored procedure that inserts a new person customer and customer address.

```
CREATE Procedure InsertCustomer
        @CustomerID INT,
        @CreditRating INT,
        @AccountType CHAR(10),
        @EmailAddress CHAR(25),
        @CreditLimit DECIMAL(21,13),
                                @CashBalance DECIMAL (21,13)
AS
INSERT INTO Customers
        VALUES (
        @CustomerID,
        @CreditRating,
        @AccountType,
        @EmailAddress,
        @CreditLimit,
        @CashBalance);
```

Listing 1: SQL Code for Creating a Stored Procedure to Insert a Customer

```
CREATE PROCEDURE InsertCustomerAddress
        @CustomerID INT,
                                @Address CHAR(30),
        @City CHAR(25),
        @State CHAR(2),
        @ZipCode CHAR(10)
AS
INSERT INTO CustAddresses
VALUES(
        @CustomerID,
        @Address,
        @City,
        @State,
        @ZipCode);
```

Listing 2: SQL Code for Creating a Stored Procedure to Insert a Customer Address

Listing 4 shows the code for executing this stored procedure by passing the appropriate parameters.

WEB GRAPHICAL USER INTERFACE DESIGN COURSE

In the web GUI design course, students learn to plan, design, develop and maintain Web sites that follow W3C standards for XHTML, CSS and accessibility. In addition, students learn basic skills in image manipulation and how to use mashups (like Google maps) in a Web site. The Kanab case is used to help students develop online forms that will be used as part of a web application.

Objectives

The objectives for this instructional module are:

- Students will understand the purpose of web forms as tools for accepting data from a user
- Students will become familiar with the various types of form elements and how each relates to different types of user input
- Students will be able to create accessible forms using strict XHTML
- Students will be able to apply design elements of forms using valid CSS

```
CREATE Procedure InsertPersonCustomer
        @CreditRating INT,
        @AccountType CHAR(10),
        @EmailAddress CHAR(25),
        @CreditLimit DECIMAL(21,13),
        @CashBalance DECIMAL (21,13),
        @FirstName CHAR(20),
        @LastName CHAR(20),
        @Address CHAR(30),
        @City CHAR(25),
        @State CHAR(2),
       @ZipCode CHAR(10)

AS

DECLARE @NewCustomerID INT

SELECT @NewCustomerID = MAX(CustomerID)+1 FROM Customers

EXEC InsertCustomer
        @NewCustomerID,
        @CreditRating,
        @AccountType,
        @EmailAddress,
        @CreditLimit,
        @CashBalance

EXEC InsertCustomerAddress
        @NewCustomerID,
        @Address,
        @City,
        @State,
        @ZipCode


INSERT INTO PersonCustomers
        VALUES (
        @NewCustomerID,
        @FirstName,
        @LastName);
```

Listing 3: SQL Code for Creating a Stored Procedure to Insert a Person Customer and Execute Other Stored Procedures

```
EXEC InsertPersonCustomer (700, 'Margin','bob@yahoo.com', 25.000, 5.000, 'Bob', 'Miller' 'Marketstreet 24',
                           'Seattle', 'WA', '12345')
```

Listing 4: SQL Code for Executing the InsertPersonCustomer Stored Procedure

Knowledge/Skills Taught

XHTML form elements demonstrated in example:

- form, input, select, option, label, fieldset, legend

XHTML form elements not demonstrated in example:

- textarea, optgroup,

CSS properties demonstrated in example:

- Alignment of fields with float
- Effects of width on fieldset
- Effects of background-color on fieldset (browser differences)

Accessibility principles taught:

- tabindex: importance of making a page easy to navigate without a mouse
- label: making forms accessible for screen readers

Sample Activation Discussion and Questions:

"Recall the Kanab Currency Trading Company case for which you created a database in your database course. Among the various database transactions for which you wrote SQL statements was inserting a new person customer into the database.

- In the Kanab company, who might need to perform the operation of inserting a new customer into the database? [*Someone like a customer account manager or sales rep may need to perform this operation. This person would not necessarily have database expertise.*]
- How can the Web be used to create a user interface for a database application like the Kanab CTA? What are the advantages of creating such an interface? [*Web forms can be developed that present the user with the necessary fields for performing an operation, such as inserting a new customer. The advantage to a web interface is that it can be used by many types of users, including non-technical users, and can be accessed from any computer that is connected to the Internet.*]
- What specific data fields would a user need to enter if s/he wanted to insert a new person customer into the Kanab database? [*List the fields required as parameters in the stored procedure.*]
- What types of form elements could a web designer use in creating a web form to allow a user to insert a person customer in the database? [*Explain various types form elements to students, including radio buttons, checkboxes, text inputs, etc. and how each type of element corresponds to database field types.*]

Code

Figure 3 shows the User Interface for the "Add New Person Customer" form that the students have to develop in class after the instructor has demonstrated the different form elements. The code following the XHTML strict doctype is shown in Listing 5. CSS styling (Listing 6) is kept to a minimum; only the importance of the alignment of the labels and fields is demonstrated.

WEB PROGRAMMING COURSE

The web application programming course teaches students how to develop web applications using PHP. The web applications developed in the class have a GUI frontend and a database backend. This course assumes some background in database development and web design, which will typically have been provided in the two courses described earlier. The web application programming course uses the Kanab case in multiple instructional modules, from simply retrieving data from a database and displaying them to the more complicated issue of structuring programs using include files. The example described in detail below demonstrates a sample instructional module for creating the necessary code to insert a person customer into the Kanab database. The example uses PHP as the server-side programming language, though other platforms (e.g. ASP.NET) could easily be substituted.

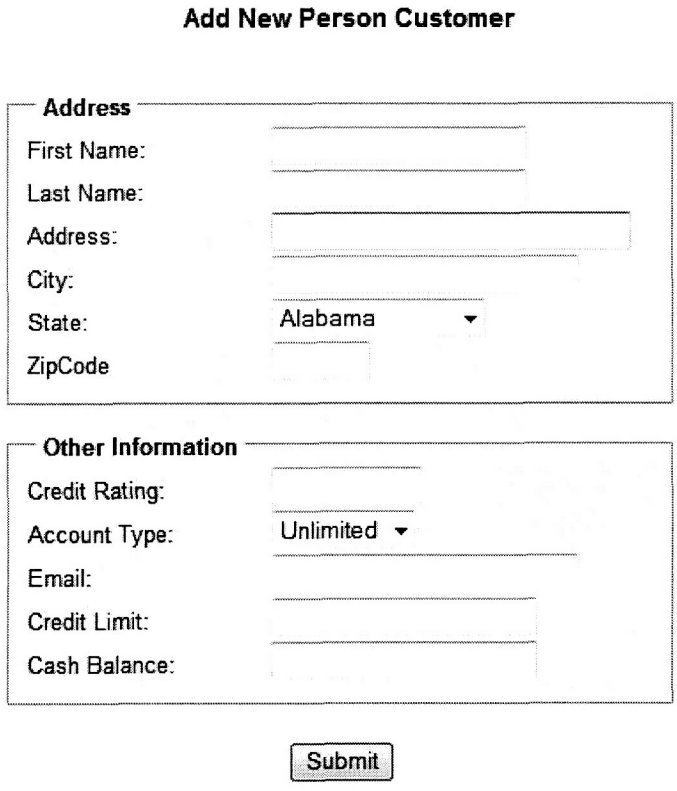

Figure 3: Add New Person Customer Form

Objectives

The objectives for this instructional module are:

- Students will understand the interaction between forms and user, including error handling
- Students will understand the interaction between forms and database tables

Knowledge/Skills Taught

- Integrating forms into PHP code
- Form submissions via POST
- Sending error messages to the user
- Connecting to a database
- Inserting, updating, and deleting data in a table
- How to structure code that handles forms

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<link rel="stylesheet" href="CurExForm.css" type="text/css"  />
<title>Currency Trading Application</title>

</head>
<body>

<h1>Add New Person Customer</h1>
    <form action="test.php" method="post" enctype="text/plain">
    <fieldset>
        <legend> Address </legend>
            <label for="FirstName" >First Name:</label>
                <input type="text" id="FirstName" name="FirstName" size="20" maxlength="20" /> <br />
            <label for="LastName">Last Name: </label>
                <input type="text" id="LastName" name="LastName" size="20" maxlength="20" />  <br />
            <label for="Address">Address: </label>
                <input type="text" id="Address" name="Address" size="30" maxlength="30" /> <br />
            <label for="City">City:</label>
                <input type="text" id="City" name="City" size="25" maxlength="25" /> <br />
            <label for="State">State:</label>
                <select id="State" name="State" size="1">
                    <option value="AL" > Alabama </option>
                    <option value="AK" > Alaska </option>
                    <!-- other states here      -->
                    <option value="WY" > Wyoming </option>
                </select> <br />
            <label for="ZipCode" >ZipCode</label>
                <input type="text" id="ZipCode" name="ZipCode" size="5" maxlength="5" /> <br />
        </fieldset>

        <fieldset>
        <legend> Other Information </legend>
            <label for="CreditRating">Credit Rating:</label>
                <input type="text" id="CreditRating" name="CreditRating" size="10" maxlength="10" /> <br />
            <label for="AccountType">Account Type:</label>
                <select id="AccountType" name="AccountType" size="1">
                    <option value="Unlimited" > Unlimited </option>
                    <option value="Margin" > Margin </option>
                    <option value="Basic" > Basic </option>
                </select><br />
            <label for="EmailAddress" >Email: </label>
                <input type="text" id="EmailAddress" name="EmailAddress" size="25" maxlength="25" /> <br />
            <label for="CreditLimit">Credit Limit:</label>
                <input type="text" id="CreditLimit" name="CreditLimit" size="21" maxlength="21" /> <br />
            <label for="CashBalance">Cash Balance:</label>
                <input type="text" id="CashBalance" name="CashBalance" size="21" maxlength="21" /> <br />
        </fieldset>

        <p><input id="button" type="submit" value="Submit" /></p>
    </form>
</body>
</html>
```

Listing 5: XHTML Code for Customer Form

```
body      {   font:  80% Arial, Helvetica, sans-serif;
              width: 800px;
              margin: 0;
              padding: 0;
              text-align: center;   }

h1        {   margin-top: 20px;
              font-size: 1.5em;
              padding-left: 10px; }

fieldset  {   width: 360px;
              background: #fff8dc;
              padding: 10px;
              line-height: 200%;
              display: block;
              margin: auto;  }

legend    {   font-weight: bold;  }

label     {   display: block;
              width: 140px;
              float: left;
              text-align: left;}

select,
input     {   float: left;}

#button   {   float: none;    }
```

Listing 6: CSS Code for Customer Form

Sample Activation Discussion and Questions:

"Recall the Kanab Currency Trading Company case from your database and web GUI design courses. You wrote a stored procedure that inserts a new person customer into the database, and you also developed a web form that allows an end-user to enter information for a new customer to be inserted. However, for the insert to take place, we must write programming code that can retrieve the data entered by the user in the form and perform the insert operation.

- Conceptually, what must occur for the new customer information entered by the web user to be written to the database? [*Code must be written that retrieves the values from the form, connects to the database, and inserts the values*].
- What should happen if the user enters an invalid value for a certain field, such as an alphabetic entry for the 'credit limit' field? [*The application should check to see that all entries are valid. If one or more entry is invalid, the application should prompt the user to enter a valid value and should not attempt to execute the operation until all entries have been corrected.*
- What might happen if you do not ensure that user entries are valid before attempting to perform the insert operation? [*Data type errors may occur that may halt the operation or cause the application to crash.*]

Code

Figure 4 shows the main page of the CTA that contains the menu and displays either the person customer data or the corporate customer data. Figure 5 illustrates how the "Add New Person Customer" form integrates with the rest of the application.

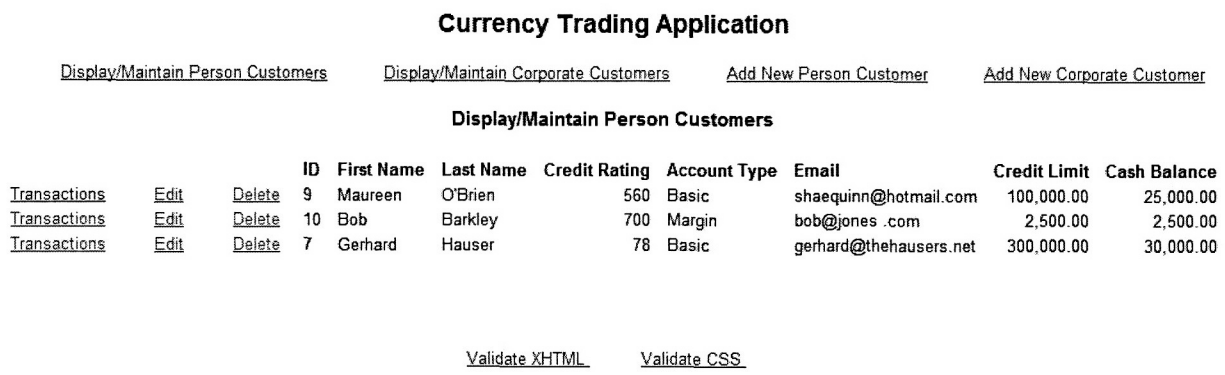

Figure 4: User Interface for Display/Maintain Person Customer

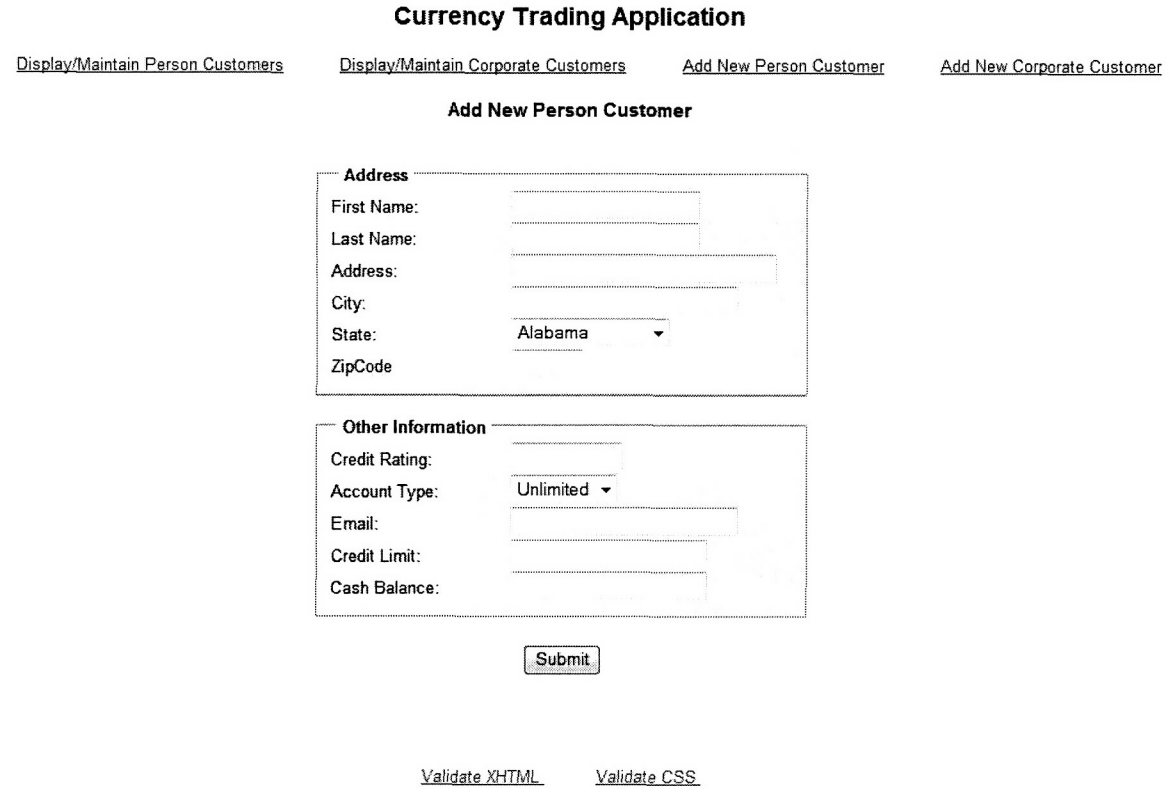

Figure 5: User Interface for Adding New Person Customer

Listings 7 and 8 show the PHP code that handles the form and inserts the data in the database.

One component students typically struggle with is how to send out error messages and how to structure a program for error handling. To help students grasp these concepts, a class discussion is initiated to first come up with all the components that program needs and then determine how to order them. The flow diagram for the components is shown in Figure 6.

```
<?php
include("incConnectDB.php");
$errors = array();

// Check for submission via post
if( $_SERVER['REQUEST_METHOD'] == 'POST' ) {

    // Trim the values
    foreach($_POST as $key => $value) {
        $_POST[$key] = trim($value);
    } // foreach

    // Check for errors
    include("incErrorPerson.php");

    // No errors found --> update database
    if( ! count($errors) ) {
        // Escape special characters
        foreach($_POST as $key => $value) {
        $_POST[key] = mysqli_real_escape_string($link, $_POST[$key] = trim($value));
        } // foreach

        // call stored procedure to insert data

        $sp = "CALL insertPersonCustomer (\"$CreditRating\", \"$AccountType\", \"$EmailAddress\",
                    \"$CreditLimit\", \"$CashBalance\",\"$Address\", \"$City\", \"$State\",
                    \"$ZipCode\",\"$FirstName\",\"$LastName\")";

        $insPersonCustomer = mysqli_query($link, $sp);

        // Redirect to display page
        header("Location:http://$url/displayCustomerPerson.php");
    } // if
} // if

include("incHeader.php");
?>
```

Listing 7: PHP Code for Adding a Person Customer Part 1

The program starts by checking whether the form is displayed for the first time or not (post variable is set). If the form is displayed for the first time (1) the form will be displayed. If the form has been submitted (2), the form data has to be checked for errors. If errors are found the error messages and the form are displayed, making sure that the data entered previously are not erased. If no errors are found, the data are entered in the database and the user is redirected to the display customer screen. It is important to emphasize that the checking for submission, errors and adding data to the database or displaying error messages has to be done before the form is displayed to the user.

```
<h2>Add New Person Customer</h2>

<!-- Display error messages  -->
<?php if($errors[0] != '') { ?>
      <div class="errors">
            <?php
            foreach( $errors AS $message )
            echo '<div id="errors">' . $message . '</div>';
            ?>
      </div>
<?php } ?>


<!-- Display form -->
<form action="<?php echo $_SERVER['PHP_SELF'];?>" method="post">

      <fieldset>
      <legend> Address </legend>
            <label for="FirstName" >First Name:</label>
                  <input type="text" id="FirstName" name="FirstName" size="20" maxlength="20" value="<?php echo
$FirstName; ?>" /> <br />
            <label for="LastName">Last Name: </label>
                  <input type="text" id="LastName" name="LastName" size="20" maxlength="20" value="<?php echo
$LastName; ?>"/>  <br />
            <?php
            // Include fieldset for Customer Address
            include("incAddCustAddress.php");
            ?>
      </fieldset>

      <?php
      // Include fieldset for Other Customer Information
      include("incAddCustOtherInfo.php");
      ?>

      <p>
      <input type="submit" value="Submit" />
      </p>

</form>

<?php
include("incFooter.php");
?>
```

Listing 8: PHP Code for Adding a Person Customer Part 2

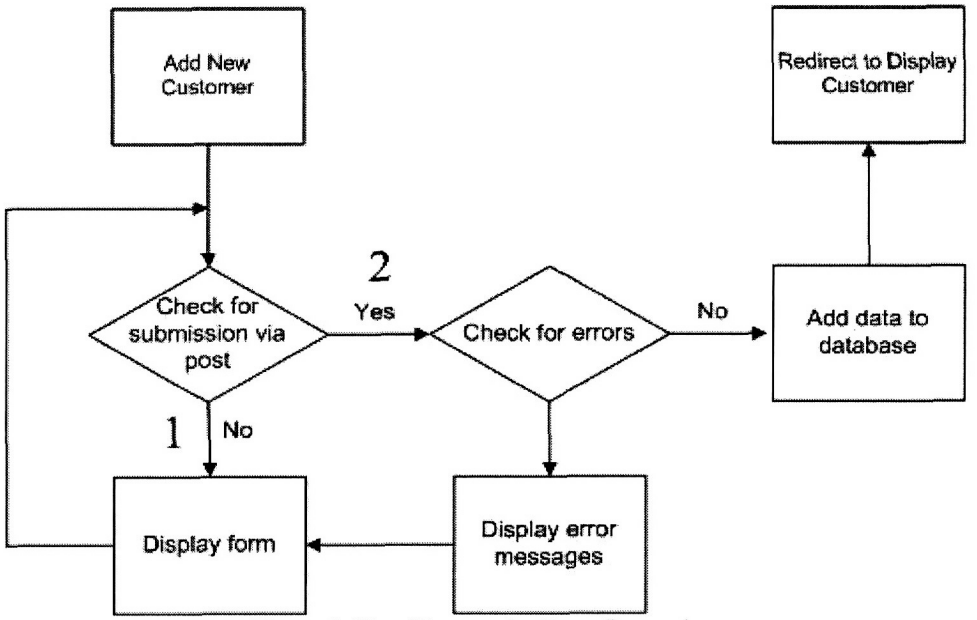

Figure 6: Flow Diagram for Form Processing

CONCLUSIONS

This paper shows how one business case can be used in several different classes related to web development. This integrated approach provides instructors with a meaningful example for their classes and provides a platform for activating prior student knowledge in teaching web development. This case study can also help students see how the different skills acquired during their coursework fit together in a holistic view of web development principles and practices.

AUTHOR INFORMATION

Karina Hauser is an associate professor in the Management Information Systems department at Utah State University. She received her PhD in Decision Science and Information Technology at the University of Kentucky on a Toyota Fellowship. Her research interests are in Web Development, Web Design and Lean Manufacturing. Before going into academia, Karina spent 16 years in industry, first as a programmer and later as a consultant and project manager for Enterprise Resource Planning systems, mainly in the automotive sector. Her research appeared in journals such as the *Journal of Information Systems Education, Journal of Computer Information Systems*, and *International Journal of Production Research.*

David Olsen received a master's degree in accounting from Brigham Young University and a Ph.D. in Management Information Systems from The University of Arizona in 1993. Dr. Olsen joined the MIS department at Utah State University in 1998 and teaches primarily in the database area as well as MBA strategy and management. Dr. Olsen's research interests include database concurrency control, accounting information systems, the integration of SQL, XML and XBRL, and database modeling. He has been published in journals such as *Communications of the ACM*, *Issues in Accounting Education*, and the *Journal of Database Management*

Kelly J. Fadel is an Assistant Professor of Management Information Systems at Utah State University. He received his PhD from The University of Arizona. His research areas include IT change management, knowledge management, end-user learning, and post-adoptive technology use and impact. His research has appeared in journals such as the *Data Base for Advances in Information Systems* and *Review of Business Information Systems* and has received awards at the ACM SIGCPR Conference on Computer Personnel Research and other international IS conferences.

REFERENCES

1. Andre, T., Selected microinstructional methods to facilitate knowledge construction: Implications for instructional design, in *Instructional design: Internation perspective: Theory, research, and model*, R.D. Tennyson, et al., Editors. 1997, Lawrence Erlbaum Associates: Mahwah, NJ. p. 243-267.
2. Bureau of Labor Statistics. *Occupational Outlook Quarterly*. 2006 [cited 12/27/06]; Available from: http://www.bls.gov/oco/ocos042.htm.
3. Koch, G. Opinion: The hottest IT skills survive a cool economy. 2008 [cited 2009 04/15/2009]; Available from: http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=329395.
4. Kung, M., Yang, S.C., and Zhang Y., The Changing Information Systems (IS) Curriculum: A Survey of Undergraduate Programs in the United States. *Journal of Education for Business*, 2006. **81**(6): p. 291-300. 2006
5. Leung, L. 10 Hot Skills for 2009. 2009 [cited 2009 04/15/2009]; Available from: http://www.globalknowledge.com/training/generic.asp?pageid=2321&country=United+States.
6. Litecky, C., B. Prabhakar, and K. Arnett, The IT/IS Job Market: A Longitudinal Perspective., in SIGMIS Computer Personnel Research Conference. . 2006: Claremont, CA.
7. Merrill, M.D., First principals of instruction. *Educational Technology Research and Development*, 2002. **50**(3): p. 43-59. 2002
8. Prabhakar, B., C. , R. Litecky, and , and Arnett K., IT Skills in a Tough Job Market. *Communications of the ACM*, 2005. **48**(10): p. 91-94. 2005
9. Topi, H., et al., Revising the IS Model Curriculum: Rethinking the approach and the Process. *Communications of the AIS*, 2007. **20**: p. 728-740. 2007