DESIGN PATTERNS FOR DATABASE APPLICATION INTERFACES

Yu Fan
Joline Morrison
Department of Computer Science
University of Wisconsin – Eau Claire
Eau Claire, WI 54702
fany@uwec.edu, morrisjp@uwec.edu

ABSTRACT

Students often struggle with visualizing database application interfaces and designing appropriate interfaces for specific application tasks. To address these issues, this paper describes a research project that demonstrates design patterns for a common database application through implementing a website.

The website (http://irv.cs.uwec.edu:8080/CS321/SampleJSPSite/) supports an online shopping process. A customer is able to view products by category or by price, sign in, login/logoff, create/update profile, view previous orders, add products into a shopping cart, create/update shipping information, and proceed to check out. The system will then insert, modify or delete from the database according to the customer's request.

The purpose of this project is to help student with interface design and other common issues in database application development, such as designing data models, using JavaScript and CSSs within JSPs, creating cookies to save temporary information, validating data before accessing the database, and resolving transaction conflicts. Students can use the web pages as templates to meet their specific needs.

1. Introduction

Database courses usually emphasize back-end topics such as referential integrity, schema design, normalization, transaction management, and system administration. These concepts often seem abstract for students when they try to create their own database designs and then implement corresponding applications. To help students create Web-based database applications, we developed a website with an associated database and application-specific patterns to illustrate topics such as designing data models, connecting to the database, inserting/updating/deleting database data, and resolving transaction conflicts. It helps students visualize user interfaces for database applications, and provides a template that they can use and apply to other similar problems

The first step in this research was to develop a hierarchy of design patterns for database interfaces. These patterns define elements to represent specific data items, and element combinations to display related data. The final step was to develop domain-specific interface patterns to serve as templates for solving real world problems in web/database development.

The paper first explores the definition of general and interface design patterns, and then describes the design patterns used in the website. Finally it demonstrates some common issues in database application development.

2. Background

Gamma et. al [1] define a *design pattern* as an object class that provides a reusable solution to a problem. It is domain-independent, and can be expressed at different abstraction levels. Gamma et. al. propose that a design pattern has the following four essential elements:

- *Pattern name,* which is a short (1-3 word) descriptor that describes the problem, its solutions, and its consequences;
- *Problem,* which describes when to apply the pattern;
- *Solution,* which describes the design elements and their underlying relationships and collaborations;
- *Consequences*, which describes the pattern's results and limitations.

Tidwell [2] applies design patterns to user interface design for creating interfaces in GUI and hypermedia environments. She specifies that interface design patterns can serve as learning tools for novice designers, and provide a common interface design language to facilitate communication among designers, programmers, and users. She notes that interface designers should use the patterns with some caution, because they do not replace standard analysis, design, and usability testing.

Tidwell's most recent library [3] contains over 50 general design patterns for common interface operations ranging from very high-level items that design the overall navigation architecture of the application to specific artifacts such as textual data entry hints or ToolTips. She defines each pattern with a descriptive name (such as "Alternating Row Colors,"). She also specifies when, why, and how to use each pattern, and provides one or more visual examples. This description approach is similar to Gamma et. al's design pattern element requirements, except that it omits an explicit discussion of pattern consequences and limitations.

The next section uses these definitions and description approaches to develop and describe a series of design patterns for database application interfaces.

3. Database Interface Patterns

Table 1 describes a hierarchy of database interface design patterns, and provides definitions for each pattern.

Level	**Description**	**Pattern Name**
Atomic	Single data item	Text Item
		Text Box
		Radio Button Group
		Check Box
		List
		Image
Single-Surface	Multiple related data items that appear on a single surface	Single-record
		Tabular
		Master-detail
Multiple-Surface	Multiple related data items that appear on different application surfaces	Master-detail
		Hierarchy
		Network
		Sequence
		Tab surface
Application	Combination of single- and/or multiple-form patterns that commonly appear in applications	Logon form
		Shopping cart
		Reservation management form
		Appointment management form

Table 1: Design pattern hierarchy

Table 1's Level column describes each pattern in terms of its complexity level, from the most basic (atomic) elements to the more complex single-surface and multiple-surface patterns. An application pattern represents a single- or multiple-surface pattern that corresponds to a common database application domain. While this hierarchy is not comprehensive, it represents a large number of common database interface domains.

The sections that follow describe each level in detail through the implementation of patterns in the website. The pattern definitions define each pattern in terms of what it does, when to use it, and how to implement the pattern. The limitations section provides guidelines as to when you should not use a specific pattern.

3.1 Atomic Elements

An *atomic* element represents a single data item that a database stores, such as a customer name, or product image. These elements correspond to GUI controls (text box, radio button, and so forth) that developers commonly use represent these data items, and describe a standard set of interface elements for most relational database items.

Table 2 summarizes the pattern name, database data type, and usage definitions for the atomic elements.

3.2 Single-Surface Patterns

A *single-surface* pattern is a combination of multiple atomic elements that appear on a single computer screen. The following subsections describe the single-record, tabular, and master-detail single-surface patterns.

3.2.1 Single-Record

What: Shows data associated with a single record; data may come from a single table or from multiple tables through a join query.

When: Inserting or updating data associated with a single record.

How: Display non-editable items using text items and editable items using other elements. Retrieve existing values through a sequential or text search function. Enable foreign key selections using radio buttons or a list.

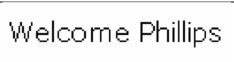
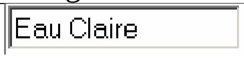
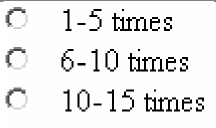
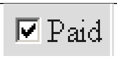
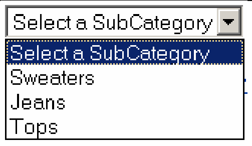
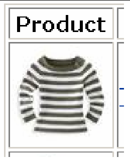
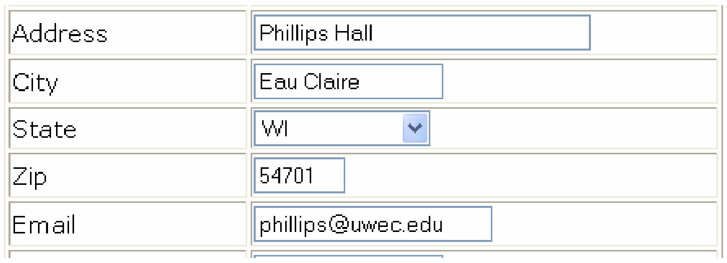
Element Name	**Data Type/Operation**	**Example & Description**
Text Item	Text/Number/Date	 Welcome Phillips Text background color is the same as the background color
	View only	
Text Box	Text/Number/Date	 Eau Claire Outlined text on white background
	View, Add, Update	
Radio Button Group	Text/Number value with five or less related, mutually-exclusive selections whose values do not change	 1-5 times 6-10 times 10-15 times Labels correspond to but don't necessary mirror associated data values
	View, Add, Update	
Check Box	Boolean	 ✓ Paid Label describes data representation; value is true if checked, false if unchecked
	View, Add, Update	
List	Text/Number data with restricted values that change over time; used to represent foreign keys	 Select a SubCategory Select a SubCategory Sweaters Jeans Tops Dropdown list that displays current selection
	View, Add, Update	
Image	Binary image data or text-based file reference	 Product Image can optionally have associated text box that displays associated folder path/filename
	View, Add, Update	

Table 2: Atomic elements

Limitations: Shows data in isolation and not within the context of other related data. It is difficult to use this pattern to retrieve a specific record unless the application provides a text search function.

Example: Figure 1 shows a single-record pattern comprised of text boxes and a list.



Address	Phillips Hall
City	Eau Claire
State	WI
Zip	54701
Email	phillips@uwec.edu

Figure 1: Single-record pattern.

3.2.2 Tabular

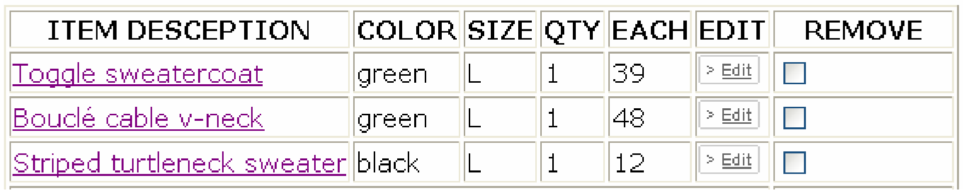
What: Shows data associated with a multiple related records; data may come from a single table or from multiple tables through a join query.

When: Viewing multiple related records; inserting, and updating a record within a set of records.

How: Display non-editable items using text items, and editable items using alternate elements such as text boxes, check boxes, or lists. You should provide vertical and/or horizontal scrollbars to show additional data, and enable foreign key selections using radio buttons or a list.

Limitations: It is difficult to display multiple columns more than five or six columns without running out of horizontal screen space. It is difficult to find a specific record in this display unless the application provides a link or a text search function. This display does not readily display data using a radio button group.

Example: Figure 2 shows a tabular pattern comprised of text items, a link, a button, and a check box.



ITEM DESCEPTION	COLOR	SIZE	QTY	EACH	EDIT	REMOVE
Toggle sweatercoat	green	L	1	39	> Edit	☐
Bouclé cable v-neck	green	L	1	48	> Edit	☐
Striped turtleneck sweater	black	L	1	12	> Edit	☐

Figure 2: Tabular pattern.

3.2.3 Master-Detail (Single-surface)

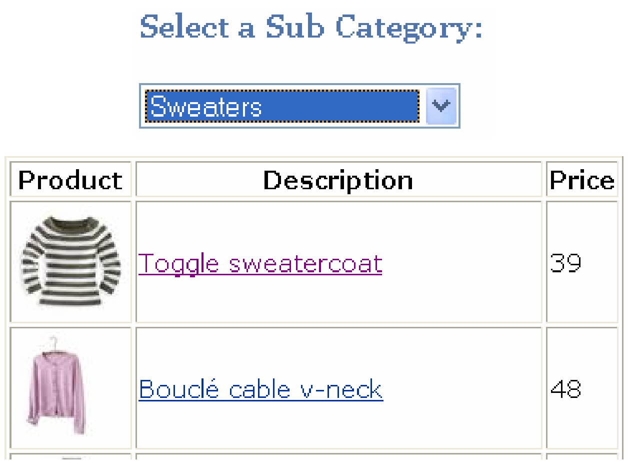
What: Shows data associated with a master-detail (one-to-many) relationship. This pattern usually uses a single-record pattern for the master records and a tabular pattern for the detail records.

When: Enabling the user to selecting a master record and then view/insert/update a detail record.

How: Display master records using a list or option buttons, and detail records using a tabular display.

Limitations: This display can become confusing if you allow users to edit master and detail records simultaneously. You usually do not use this pattern in a Web application because it requires re-populating the master display each time the server generates the page.

Example: Figure 3's single-surface master-detail pattern allows the user to select a master item (which in this case is a Sub Category name, such as "Sweater") from a list, and displays the related detail items (product, description and price) in a tabular pattern. The user can click the Description link to open the item for viewing.


Select a Sub Category:

Sweaters

Product	Description	Price
	Toggle sweatercoat	39
	Bouclé cable v-neck	48

Figure 3: Single-surface master-detail pattern.

3.3 Multiple-Surface Patterns

A *multiple-surface* pattern combines single-surface patterns that perform one or more application tasks across multiple computer screens.

3.3.1 Master-Detail (Multiple-surface)

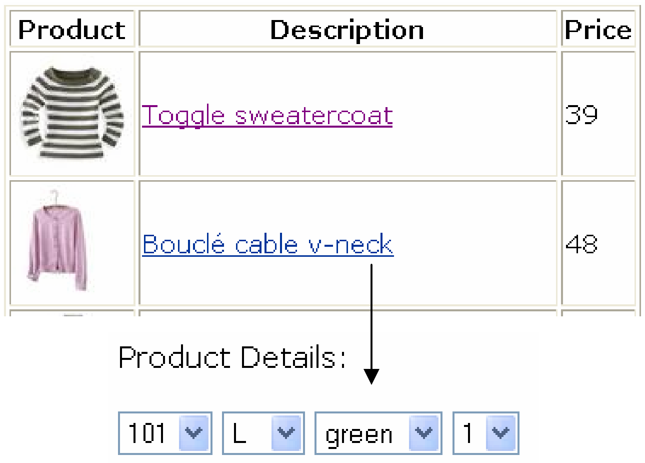
What: Shows data associated with a master-detail (one-to-many) relationship. In the multiple-surface display, this pattern can use a tabular display for both master and detail records.

When: Viewing/inserting/updating master and detail records which display both master and detail records in tabular patterns; displaying master-detail data on Web pages.

How: Display master records using a list, option buttons, or a tabular display, then display related detail records using a list, or a tabular display.

Limitations: Applications require error trapping to prevent users from deleting master items and generating referential integrity violations; user may lose the sense of the master-detail relationship because it is on multiple screens.

Example: Figure 4 shows a multiple-surface master-detail pattern. The user clicks a hyperlink on the master item display, and the related detail items appear in a list display.



Product	Description	Price
	Toggle sweatercoat	39
	Bouclé cable v-neck	48

Product Details:

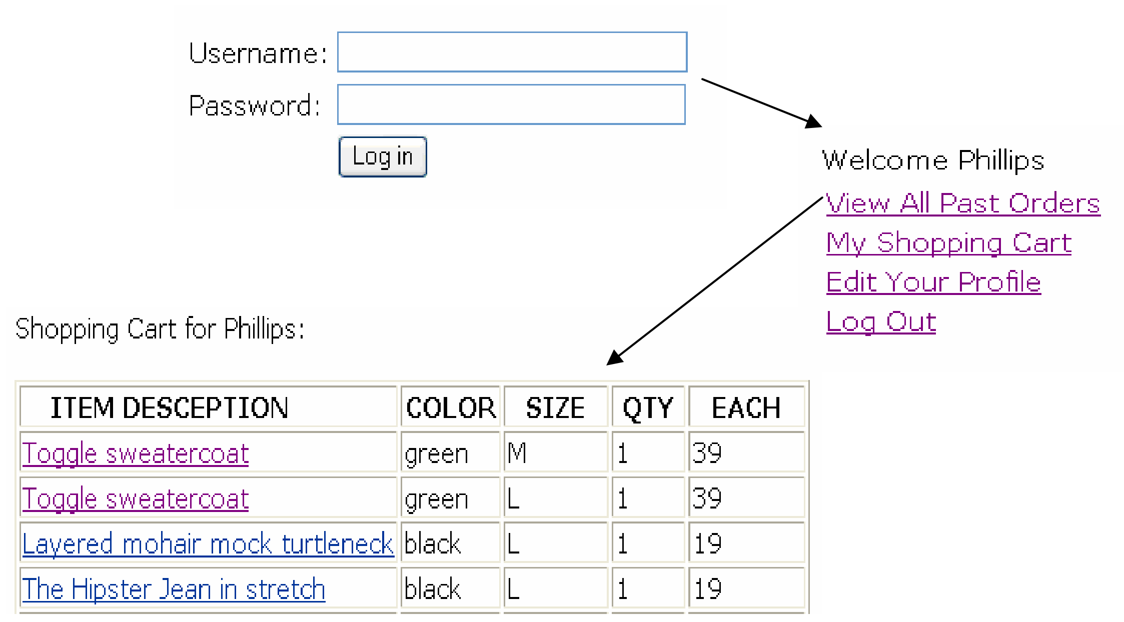
101 L green 1

Figure 4: Multiple-surface master-detail pattern.

3.4 Application Patterns

An *application* pattern specifies an appropriate multiple-surface design pattern for a common database application. Table 1 suggests common application patterns (Login form, Shopping cart, Reservation management form, Appointment management form).

Figure 5 illustrates a shopping cart application design. This is application pattern enables the user to login, order items using a single-record pattern, and then view order detail information. The order detail interface allows the user to select from a list of ordered items that appear in a tabular pattern.



Shopping Cart for Phillips:

ITEM DESCEPTION	COLOR	SIZE	QTY	EACH
Toggle sweatercoat	green	M	1	39
Toggle sweatercoat	green	L	1	39
Layered mohair mock turtleneck	black	L	1	19
The Hipster Jean in stretch	black	L	1	19

Figure 5: Sequence multiple-surface pattern.

4. Common Issues in Database Application Development

A relational database was built before the implementation of the website. When a user makes a request, the system would then insert, modify or delete from the database accordingly.

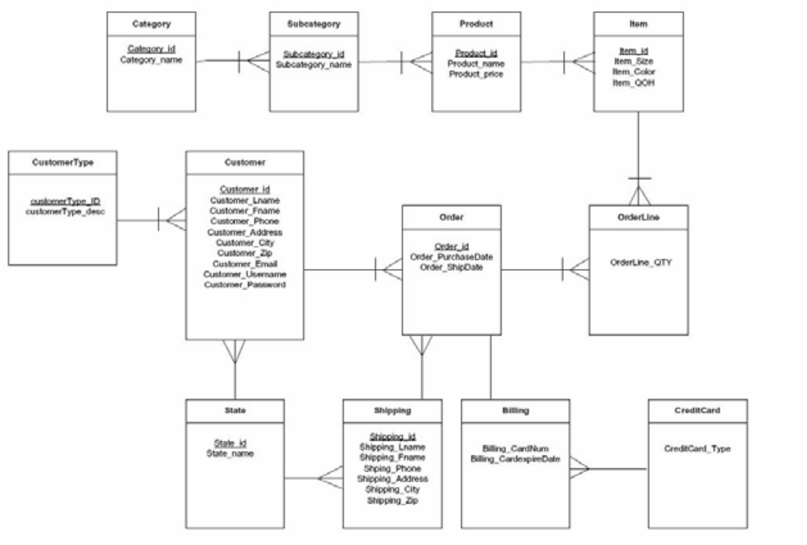
Figure 6 is a screenshot of the webpage where students can view the initial data model diagrams and download the SQLs used in the database.

There is a total of 12 tables.

Data Model Diagram 1

Data Model Diagram 2

Sample Tables



SQL Downloads (Download these files

create_table.sql

create_instances.sql

drop_tables.sql

sample_queries.sql

```
create table orderLine
  (item_id          smallint    ,
   order_id         smallint    ,
   orderLine_orderQTY     int          ,
   CONSTRAINT orderLine_item_id_fk FOREIGN KEY (item_id) REFERENCES item(item_id),
   CONSTRAINT orderLine_order_id_fk FOREIGN KEY (order_id) REFERENCES orders(order_id),
   primary key (item_id, order_id)
);

create table cart
  (cart_id                  smallint,
   cart_itemid      smallint,
   cart_itemQOH         int ,
   cart_date             Date,
   CONSTRAINT cart_itemid_fk FOREIGN KEY (cart_itemid) REFERENCES item(item_id)
  );

create SEQUENCE cart_sequence INCREMENT by 1 START with 2;
```

Figure 6: Tables and Queries

To give an overall picture of how all the web pages fit together, we have presented a tree diagram of the JSP file structure, including where each JSP site is and its related database code. Figure 7 is the overall design of the shopping site.

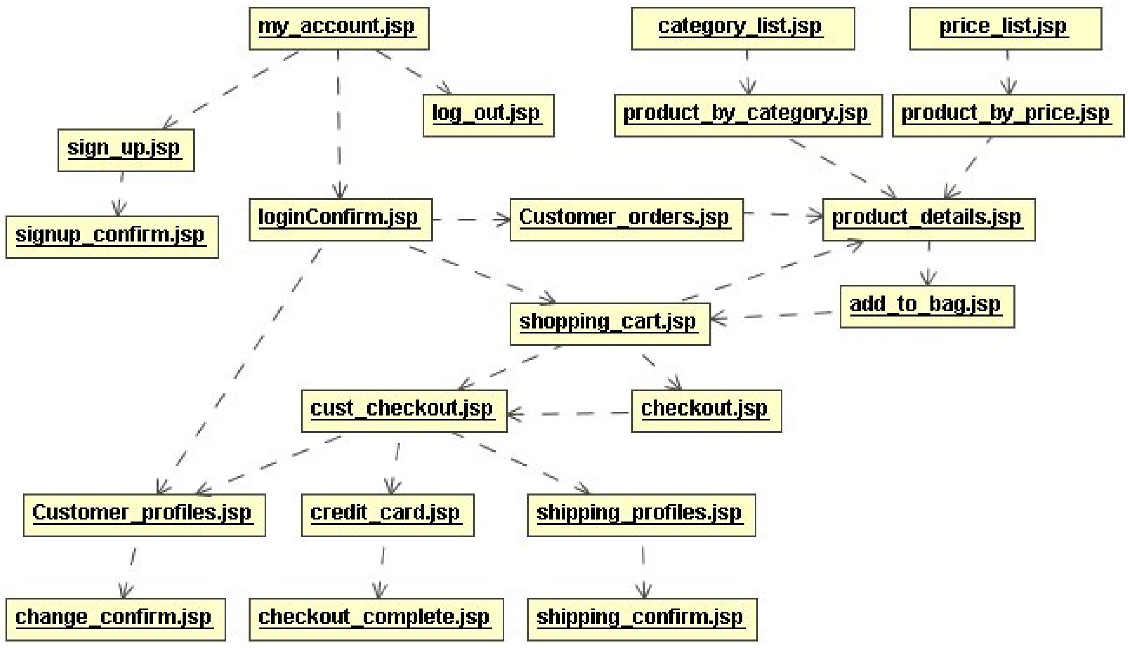
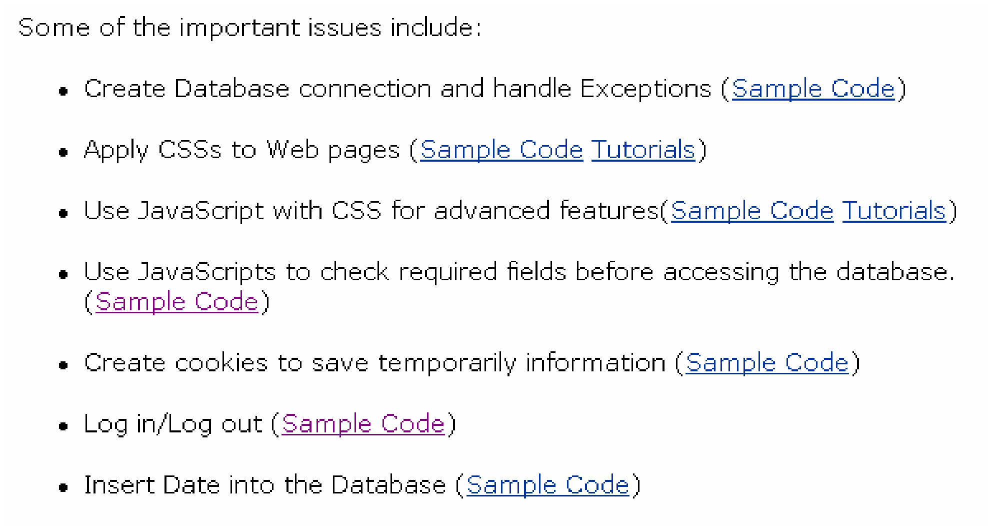


Figure 7: Shopping Site Design

To make the site more helpful to students who are learning how to develop Web/database applications, we have also presented sample code for some critical issues we have encountered during the development of this site. These sample code illustrates topics such as creating cookies, applying CSSs to webpages, checking required fields in JavaScript, and so forth. Figure 8 is a screen shot of the page where we present sample code.


Some of the important issues include:

- Create Database connection and handle Exceptions (Sample Code)
- Apply CSSs to Web pages (Sample Code Tutorials)
- Use JavaScript with CSS for advanced features(Sample Code Tutorials)
- Use JavaScripts to check required fields before accessing the database. (Sample Code)
- Create cookies to save temporarily information (Sample Code)
- Log in/Log out (Sample Code)
- Insert Date into the Database (Sample Code)

Figure 8: JSP Samples

5. System Evaluation

This website is currently being used to aid a database class in developing a similar Web-based application, and we are collecting feedback to help us modify the Web site's utility as a teaching aid. Students are required to use the Web site as part of the class project, and then complete a survey evaluating its usefulness and usability. (The survey is attached to this paper as an appendix.) The results will be used to modify and improve the Web site's features.

6. Conclusion

Design patterns provide a reusable, domain-independent approach for describing common database application elements and element pattern combinations. This paper identifies atomic elements for representing individual data items, and application-independent single- and multiple-surface patterns that combine atomic elements. The guidelines for when and when not to use each pattern will help students and novice designers to make sound design decisions. These patterns provide a logical and structured way to teach database application design.

The patterns have limitations. They are not comprehensive, because each application will require custom interface elements to meet their user requirements. Also, every pattern is not amenable to every development environment.

The website implements some typical patterns to help students visualize design pattern concepts in database applications. The database diagrams and queries, sample code, and the tree structure of the web pages can be used as templates to facilitate teaching database application development: students can modify the templates to meet the needs of their specific database applications.

7. Acknowledgements

Thanks to Mike Morrison for his help in developing the design pattern concepts and associated application examples.

References

[1] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Pattern,* Addison-Wesley, Reading, MA, 1995.

[2] Tidwell, J. *Interaction design patterns. In Patterns Languages of Programs (PLoP '98),* (Allerton Conference Center, Urbana Champaign, IL.)

[3] Tidwell, J. *UI Patterns and Techniques,* http://time-tripper.com/uipatterns.

Appendix - System Evaluation Survey

1. How many times have you visited this site?
   - ○ 1-5 times
   - ○ 6-10 times
   - [✓ 10-15 times]
   - ○ More than 15 times

   On a scale of 1-5:

2. How helpful is this site for doing assignments? (5 = most helpful)

   ○ 5 ○ 4 ○ 3 ○ 2 ○ 1
3. Are the code examples clearly presented? (5 = most clear)

   ○ 5 ○ 4 ○ 3 ○ 2 ○ 1
4. Are the code examples relevant to the assignment? (5 = most relevant)

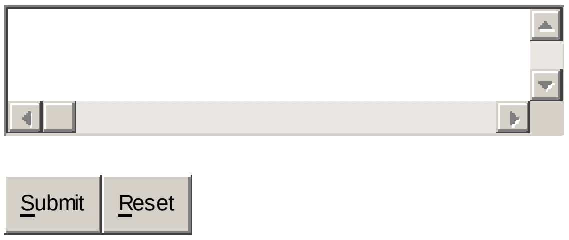
   ○ 5 ○ 4 ○ 3 ○ 2 ○ 1
5. Are the other materials (e.g., database examples, sample tables, queries) clearly presented? (5 = most clear)

   ○ 5 ○ 4 ○ 3 ○ 2 ○ 1
6. Are the other materials relevant to the assignment? (5 = most relevant)

   ○ 5 ○ 4 ○ 3 ○ 2 ○ 1
7. Please list issues you find most helpful about this site. (e.g., text formatting, cookies)

8. Please list issues you would like to include in this site to make it more useful.

9. Any other comments/Suggestions?


Submit
Reset