

Name: Sharon Meraz
Class: CS 327E, Elements of Databases
Professor: Ajay Bhargava
Date: May 6, 2005

Project Title: Books Warehouse

Contents

Project Description.....	2
PowerPoint Presentation	10
Screen Shots	12
ER Diagram	17
Table Design DDL and Stored Procedures	19
Source Code	34

Purpose of the Project

My project, titled “Books Warehouse” was conceived of as a database to house resources pertinent to dissertation studies. Though the title suggests that the primary goal of the database was to house books, the scope of the project included the storing of journal articles as well as URLs. The importance of flexibility in design was pertinent to the success of this project as the database would need to be expandable in the future to include storing conference papers and edited volumes of work. For the initial phase of the project, though, the warehouse was limited in scope to include books, articles, and URLs of resources.

The main purpose of this project was to provide a searchable database as well as a Web application for entering in books related to my dissertation studies. PhD students often need a resource to store their materials. Materials can also be segmented into categories per a PhD student’s interests and in alignment with the student’s program of work and identified areas of study. This database would also be designed to permit segmenting books into user-defined categories. Finally, having a system to track author publications was also essential. This database was designed to permit easy author searches as well as associate authors with multiple publications as publications with multiple authors.

Because there would be only one user of the system, it was important to develop security; yet, the security side of the application was not as essential for the launch of the project. However, the database design would need to be flexible enough to incorporate security and authentication into the system. As a living project that would be continuously developed in the future, it was important that the database be designed with future requirements in mind.

Team Members and Their Roles

Because of the highly specific and defined nature of this project, it was decided that this group would be comprised of only one member—the person who was the main stakeholder of the project. Though the database would be extensible such that others

could use the resource, the highly customized and personalized nature of the project needs necessitated that this project be designed and developed by the person who would eventually use the database. The database's main purpose would be to serve one person's path towards completing course work, preparing for oral/comprehensive exams, writing a dissertation, and future work as a professor. As such, it was an excellent opportunity for a database design tailored to meet the specific needs of one person with a focused goal and clear database requirements.

The sole member of this group played three main roles:

1. Front end-developer, with the main responsibilities to include programming language selection, work flow implementation of UI screens and look and feel of the application from the end user perspective
2. Business Logic Developer with main responsibilities to include providing stored procedures headers, coming up with the body of the stored procedures and creating requirements for database schema.
3. Database Schema Developer with main responsibilities to include converting the logical ERWin diagram into a physical database, coming up with conventions for table/view and column names, referential integrity specifications, the DDL associated with CREATE and INSERT statements, and specification of data types, lengths, and defaults for various columns

With the sole member of the group herself a developer by profession, this project was easily facilitated by the usage of code libraries developed by the group member from previous projects within the workplace and within project team environments. Using reusable objects and namespaces from previous projects developed in group environments enabled the sole project member to rapidly develop code without developing everything from scratch. This enabled the sole project member to easily wear different hats by reusing modular code from previous projects.

Assumptions/Constraints/Requirements

This project has various assumptions, constraints, and requirements. Detailing these requirements and boundaries made it possible to develop phased developmental time periods, as well as ensure that the database would be able to support future development. Some of these assumptions/constraints/requirements are listed below

Materials

1. Materials can consist of books, articles, and URLs. A material can only fit into one of these categories
2. Each of the material types would contain information specific to that type. This type-specific information would be stored at the type level, leaving general information about the type to be stored at the material level. For example, books will be tied to a publisher and a city. Journal articles will need to be tied to a Journal.
3. Referential integrity constraints should be such that if a material is removed, it should automatically cascade deletions/updates to the specific type table that is associated with that material.

Categories

1. It must be easy to add/update/delete categories
2. Categories should have a description field to ensure that the category is well defined and to make categories mutually exclusive. Since categories can be separated by fine lines of distinction, detailing the description of these categories can aid in recall of distinctions
3. A material must be able to be added into several categories at a time. There must be no limit on the amount of categories a specific material can be inserted into.
4. Associations between categories and materials exist independent of the type of material.

5. If categories are deleted/updated, referential integrity rules should be such that the associations between materials and categories are automatically removed from the database without manual deletions/updates.
6. Categories can be added independent of being tied to a publication.

Authors

1. A material can be authored by several authors. It must be possible to enable multiple authors to be associated with one material
2. Associations between authors and materials exist independent of the type of material.
3. An author can also author several materials. The database must find a way to translate this many-to-many relationship.
4. An author must be given the optional ability to add a biography. Biographies help in identifying the affiliation of the author, status, professional affiliations of the author.
5. If an author is deleted from the database, referential integrity rules must be such that all references to that author's connection to materials must be removed. This removal must not disturb or affect the other authors connected to a said material.
6. Authors could be added independent of being tied to a material.

Publishers

1. Since publishers tend to be repeated across different books, it was important to have an easy way to update publishers without touching multiple records.
2. Publishers could be added independent of being tied to a book.
3. Publishers only need to be associated with books.

Cities

1. Since cities tend to be repeated across different books, it was important to have an easy way to update cities without having to update multiple records.
2. Cities could be added independent of being tied to a book.
3. Cities only need to be associated with books.

Journals

1. Since journals tend to be repeated across different journal articles, it was important to have an easy way to update journals without having to update multiple records.
2. Journals could be added independent of being tied to an article.
3. Journals are only associated with articles.

Search

1. It was important for the search to be as open and flexible. In terms of designing the search interface, parameters should be defined as “or” instead of “and” to allow the retrieval of as much materials as fit the description.
2. In keeping with the design of an open, flexible, and comprehensive search, search should span across material titles, authors, and categories. The extent of searching would be dependent on the user entering in criteria on the search page.
3. Results returned should tell user how many match the search criteria. Users should be given a way to access full details of the results items (though hyperlinks).

Admin Interface

1. It was important to have a Web interface to maintain the system. This Web interface would enable Web updating of the database without directly entering records into the database.
2. The idea of security would be important when more than one user is allowed entrance into the system. In other words, the application would need a flexible database design that supported the addition of security in the future.
3. The interface should have global navigation such that it was easy to locate things from any page in the system. Excessive hitting of the back button was something to be avoided because of the dynamic nature of the application.

Phases of Database Design

The database design was validated at several different points in the application development, resulting in a reworking of the database to suit the demands of the requirements. Validation also involved the process of normalization, which created a greater need to rework the tables to allow for third normal form normalization.

The initial design was developed based on gathering all of the surface requirements of the system. Based on the detailed assumptions/requirements/constraints sections, the initial design produced the following tables:

1. Material
2. Book
3. URL
4. Article
5. Category
6. MaterialCategory
7. Author
8. AuthorMaterial

On further examination of the table, it was realized that many articles come from repeat journals. If it were left up to the user to key in the journal title for each publication, it would lead to excessive user error. So, on further analysis, it was decided to add another table

9. Journal

Examining the books table more closely, it was realized that many books have repeat publishers as well as repeat cities of publication. Neither of these attributes, cities or publishers, are determined by books. So, to further normalize the table, both the publisher and the city information was pulled out of the database and stored in its own table. Both of these would now be referenced from the books table through referential integrity and foreign key relationships

10. Publisher

11. City

The idea of iteration was extremely important for designing the database. The rules of normalization and validation by referencing constraints and requirements made it possible to revise the database design to ensure it met the needs of the system.

Hardware and Software

In order to facilitate rapid development, a toolset very familiar to the developer was used:

1. C# .NET
2. MS SQL
3. XHTML and CSS

C# .NET was the language used to extract data from the MS SQL database, as was well as to facilitate an N-tiered design approach. C# .NET as an object oriented programming language made possible, for most of the pages, a strict separation of the data, business and presentation layers. In the end, the choice of C# .NET over PHP/MySQL was due to the object oriented nature of the former, which did allow for a multi-tiered development architecture for the project development.

The developer was already in possession of a host that supporter .NET hosting (discountasp.net). As such, using a familiar server space, server connection, and server language also enabled the sole project member and developer to remain developing in an environment that permitted rapid development as all of the hurdles (how to connect, how to pull data from a MSSQL database, etc.) was already worked out.

Books Warehouse in ASP.NET

By Sharon Meraz

Books Warehouse—Team

- Sharon Meraz (PhD student)
 - School of Journalism
 - PhD Language Requirement
 - Facilitate building tools to conduct research
 - Resource to share with other researchers

Books Warehouse—Logical Model

- [ER Model](#)
- [UI Screen Shots](#)

Books Warehouse—Technology

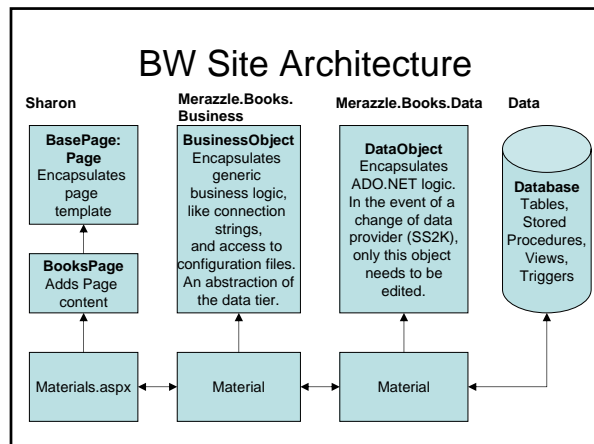
- Why Use ASP.NET
- Building on existing Web site
- Object-Oriented Web Architecture
- Built-in Class Libraries
- Efficient access to SQL Server 2000
- “Code Behind” classes for n-tier design

Books Warehouse—Technology

- ASP and ADO in .NET
- Both are part of .NET’s Framework Class Library
- ADO.NET: a set of classes to access data
- ASP.NET: a web technology that relies on the FCL and XML to produce pages
- Each page is an instantiation of an object that inherits from the Page class in ASP.NET

Books Warehouse—Technology

- A Page in .NET
- System.Web.UI namespace (~ a Java package)
- Has Events (Init, Load, UnLoad, UnInit)
- HTML and Web controls in .aspx file
- Logic that binds data to the Web controls in a separate C# file
- Code Behind technique separates designer from programmer functions







- ### Difficulties
- Played multiple roles of
 - Interface designer (presentation layer)
 - Database designer
 - Business logic designer (stored procedures)

- ### Affordances
- Project I had conceived for some time now. Requirements analysis clear
 - Able to program it rapidly using familiar toolset, i.e., .NET and MS SQL

- ### Constraints
- Solo team member
 - Made it more difficult to bounce off ideas
 - Made the programming more challenging because I developed all tiers
 - Satisfying because I was a stakeholder

- ### Recommendations
- Spend lots of time in planning. A well-planned project provides a strong foundation
 - Develop the project independent of the toolset. Don't allow the toolset to constrain the design
 - Worry about the writing the SQL code after. Develop good database design first

Books Warehouse










MENU OPTIONS
Books Database

Add
Articles
Authors
Books
Categories
Cities
Journals
Materials
Publishers
SEARCH
Urls

Keyword in Material	<input type="text"/>	(title, description)
Author Name	<input type="text"/>	(first, last, bio)
Category	All Categories	
<input type="button" value="Search"/>		

Books Warehouse

MENU OPTIONS
Books Database

Add
Articles
Authors
Books
Categories
Cities
Journals
Materials
Publishers
SEARCH
Urls

Keyword in Material	<input type="text"/>	(title, description)
Author Name	<input type="text"/>	(first, last, bio)
Category	All Categories	
<input type="button" value="Search"/>		

SELECT Distinct m.MatID, m.MatTitle FROM BK_Material m, BK_MaterialCategory mc, BK_Category c, BK_AuthorMaterial a WHERE(m.MatID = mc.MatID AND mc.CatID = c.CatID) OR (am.MatID = m.MatID AND a.AuthID = am.AuthID)

7 records found.

1. [ASP.NET Website Programming: Problem - Design - Solution](#)
2. [The political blogosphere and the 2004 U.S. Election: Divided they blog](#)
3. [Friends and neighbors on the Web](#)
4. [Power-law distribution of the world wide web](#)
5. [Linked: The new science of networks](#)
6. [Republic.com](#)
7. [Emergence of scaling in random networks](#)

Books Warehouse



MENU OPTIONS

[Add](#)

[Articles](#)

[Authors](#)

[Books](#)

[Categories](#)

[Cities](#)

[Journals](#)

[Materials](#)

[Publishers](#)

[SEARCH](#)

[Urls](#)

Details Page

Friends and neighbors on the Web, 1/1/2003 12:00:00 AM
Everybody likes good neighbors.

[Edit](#)

<http://www.cs.man.ac.uk/~rizos/web10.pdf>

[Edit](#)

Authors

L.A. Adamic

E. Adar

[Edit Authors](#)

Categories

☐ Agenda Setting

☐ Agenda Setting and New Media

☐ Blogs and Ethics

☒ Blogs and Social Networks

☐ Blogs and Traditional Journalism

☐ Blogs and Web Credibility

☐ Citizen Journalism

☐ Conferences

☐ Deliberative Democracy

☐ Digital Democracy

☐ Gender and Technology

☐ New Media Technologies

☐ Political Communication

☐ Political Communication and Media

☐ Tagging

☐ Web Development

[Update Categories](#)

Books Warehouse



MENU OPTIONS

Journals

[Add](#)

[Articles](#)

[Authors](#)

[Books](#)

[Categories](#)

[Cities](#)

[Journals](#)

[Materials](#)

[Publishers](#)

[SEARCH](#)

[Urls](#)

Enter Journal:

3

[Edit](#)

Newsweek
4/30/2005 5:47:41 PM

4

[Edit](#)

Communications of the ACM
5/5/2005 7:05:53 AM

5

[Edit](#)

Science
5/5/2005 7:30:43 AM

Books Warehouse



MENU OPTIONS

Publishers

[Add](#)

[Articles](#)

[Authors](#)

[Books](#)

[Categories](#)

[Cities](#)

[Journals](#)

[Materials](#)

[Publishers](#)

[SEARCH](#)

[Urls](#)

Enter Pub:

3 Edit	Sams Publishing 4/30/2005 4:01:09 PM
6 Edit	Joe's Publishing 5/1/2005 9:39:45 AM
7 Edit	Wrox Press 5/4/2005 7:04:46 AM
8 Edit	Perseus 5/5/2005 7:32:54 AM
9 Edit	Princeton University Press 5/5/2005 7:42:45 AM

Books Warehouse



MENU OPTIONS

[Add](#)

[Articles](#)

[Authors](#)

[Books](#)

[Categories](#)

[Cities](#)

[Journals](#)

[Materials](#)

[Publishers](#)

[SEARCH](#)

[Urls](#)

Cities

Enter City:

19	Los Angeles 4/30/2005 5:45:01 PM
Edit	
20	Indianapolis 4/30/2005 4:00:34 PM
Edit	
21	New York 4/30/2005 5:36:25 PM
Edit	
23	Birmingham, UK 5/4/2005 7:04:46 AM
Edit	
24	Cambridge, MA 5/5/2005 7:32:53 AM
Edit	
25	New Jersey 5/5/2005 7:42:45 AM
Edit	

Books Warehouse



MENU OPTIONS

Add

Articles

Authors

Books

Categories

Cities

Journals

Materials

Publishers

SEARCH

Urbs

Authors

First Name

Last Name

Bio

Add Author

13 : Georgie Porgy | [Edit](#) | [Select](#)

17 : Marco Bellinaso | [Edit](#) | [Select](#)

18 : Kevin Hoffman | [Edit](#) | [Select](#)

19 : L.A. Adamic | [Edit](#) | [Select](#)

20 : N. Glance | [Edit](#) | [Select](#)

21 : E. Adar | [Edit](#) | [Select](#)

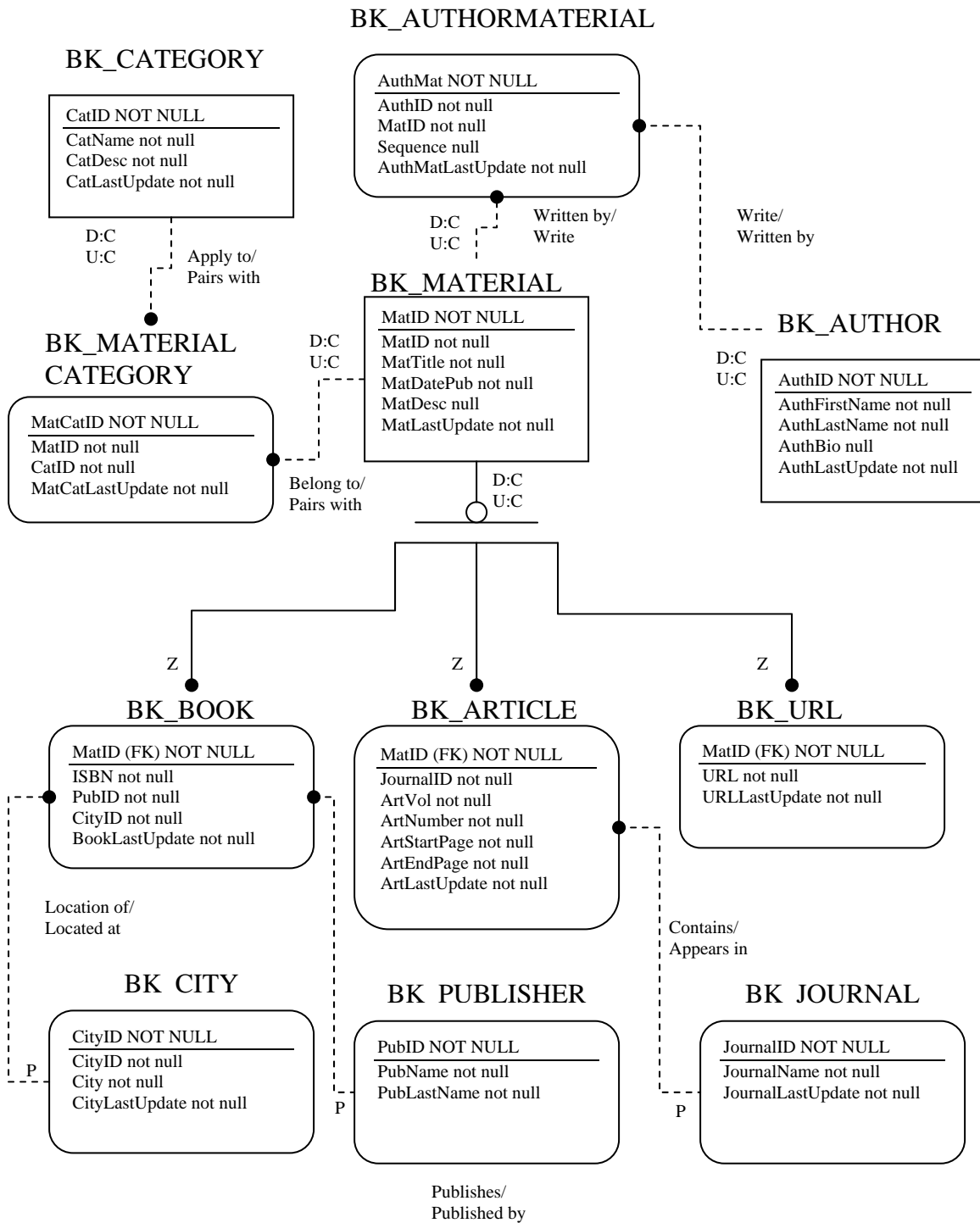
23 : B.A. Huberman | [Edit](#) | [Select](#)

24 : A.L. Barabasi | [Edit](#) | [Select](#)

25 : C. Sustain | [Edit](#) | [Select](#)

26 : R. Albert | [Edit](#) | [Select](#)

Books Database Design




```
CREATE TABLE BK_Material
(
MatID int not null identity (1,1),
MatTitle varchar (250) not null,
MatDatePub datetime null,
MatDesc text null,
MatLastUpdate datetime not null,
CONSTRAINT MatIDPK Primary Key (MatID)
);
```

```
CREATE TABLE BK_Category
(
CatID int not null identity (1,1),
CatName varchar (75) not null,
CatDesc varchar (265) not null,
CatLastUpdate datetime not null,
CONSTRAINT CatIDPK Primary Key (CatID)
);
```

```
CREATE TABLE BK_Author
(
AuthID int not null identity (1,1),
AuthFirstName varchar (25) not null,
AuthLastName varchar (25) not null,
AuthBio varchar (250) null,
AuthLastUpdate datetime not null,
CONSTRAINT AuthIDPK Primary Key (AuthID)
);
```

```
CREATE TABLE BK_Publisher
(
PubID int not null identity (1,1),
PubName varchar (150) not null,
PubLastUpdate datetime not null,
CONSTRAINT PubIDPK Primary Key (PubID)
);
```

```
CREATE TABLE BK_City
(
CityID int not null identity (1,1),
City varchar (50) not null,
CityLastUpdate datetime not null,
CONSTRAINT PubCityIDPK Primary Key (CityID)
);
```

```
CREATE TABLE BK_Book
(
MatID int not null,
ISBN varchar (20) not null,
PubID int not null,
CityID int not null,
BookLastUpdate datetime not null,
CONSTRAINT BookPK Primary Key (MatID),
CONSTRAINT BookMaterialMatID FOREIGN KEY (MatID)
references BK_Material (MatID)
on Update Cascade
on Delete Cascade,
CONSTRAINT BookPubID FOREIGN KEY (PubID)
references BK_Publisher (PubID)
on Update Cascade
on Delete Cascade,
CONSTRAINT BookCity FOREIGN KEY (CityID)
references BK_City (CityID)
on Update Cascade
on Delete Cascade
```

```
);

CREATE TABLE BK_Journal
(
JournalID int not null identity (1,1),
JournalName varchar (100) not null,
JournalLastUpdate datetime not null,
CONSTRAINT JournalIDPK Primary Key (JournalID)
)

CREATE TABLE BK_Article
(
MatID int not null,
JournalID int not null,
ArtVol int not null,
ArtNumber int null,
ArtStartPage int null,
ArtEndPage int null,
ArtLastUpdate datetime not null,
CONSTRAINT ArtMatIDPK Primary Key (MatID),
CONSTRAINT ArticleMaterialMatID FOREIGN KEY (MatID)
references BK_Material (MatID)
on Update Cascade
on Delete Cascade,
CONSTRAINT ArticleJournalJournalID FOREIGN KEY (JournalID)
references BK_Journal (JournalID)
on Update Cascade
on Delete Cascade
)

CREATE TABLE BK_URL
(
MatID int not null,
URL varchar (255) not null,
URLLastUpdate datetime not null,
CONSTRAINT URLMatIDPK Primary Key (MatID),
CONSTRAINT URLMatIDFK Foreign Key (MatID)
references BK_Material (MatID)
on Update Cascade
on Delete Cascade
);

CREATE TABLE BK_BundledMaterial
(
BundMat int not null identity (1,1),
ParentMatID int not null,
ChildMatID int not null,
BundMatLastUpdate datetime not null,
CONSTRAINT BundledMaterialBundMatPK Primary Key (BundMat),
CONSTRAINT BundledMaterialParentMatIDFK Foreign Key (ParentMatID)
references BK_Material(MatID),
CONSTRAINT BundledMaterialChildMatIDFK Foreign Key (ChildMatID)
references BK_Material(MatID)
);

CREATE TABLE BK_MaterialCategory
(
MatCatID int not null,
MatID int not null,
CatID int not null,
MatCatLastUpdate datetime not null,
CONSTRAINT MatCatIDPK Primary Key (MatCatID),
CONSTRAINT MaterialCategoryMatIDFK Foreign Key (MatID)
references BK_Material (MatID),
```



```
CONSTRAINT MaterialCategoryCatIDFK Foreign Key (CatID)
references BK_Category (CatID)
);

CREATE TABLE BK_AuthorMaterial
(
AuthMat int not null,
AuthID int not null,
MatID int not null,
Sequence int not null,
AuthMatLastUpdate datetime not null,
CONSTRAINT AuthorMaterialAuthMatPK Primary Key (AuthMat),
CONSTRAINT AuthorMaterialAuthIDFK Foreign Key (AuthID)
references BK_Author(AuthID),
CONSTRAINT AuthorMaterialMatIDFK Foreign Key (MatID)
references BK_Material (MatID)
);
```

```
CREATE PROC BK_GetAuthorsPerMatID
@AuthorMatID int
AS
SELECT am.Sequence, a.AuthID, a.AuthFirstName + ' ' + a.AuthLastName AS FullName
FROM BK_Author a
JOIN BK_AuthorMaterial am
    ON a.AuthID = am.AuthID
WHERE am.MatID = @AuthorMatID
ORDER BY am.Sequence Desc, a.AuthLastName ASC
GO

CREATE PROC BK_GetMaterialsPerAuthID
@AuthID int
AS
SELECT m.MatID, m.MatTitle
FROM BK_Material m
JOIN BK_AuthorMaterial am
    ON m.MatID = am.MatID
WHERE am.AuthID = @AuthID
GO

CREATE PROC BK_GetAuthorMaterials
AS
SELECT am.MatID, m.MatTitle, am.Sequence, am.AuthID, a.AuthFirstName, a.AuthLastName
FROM BK_AuthorMaterial am
JOIN BK_Material m
    ON m.MatID = am.MatID
JOIN BK_Author a
    ON a.AuthID = am.AuthID
GROUP BY am.MatID, m.MatTitle, am.Sequence, am.AuthID, a.AuthFirstName, a.AuthLastName
ORDER BY am.MatID, am.Sequence Desc
GO

CREATE PROC BK_InsertAuthorMaterial
@AuthMat int OUTPUT,
@AuthID int,
@MatID int,
@Sequence int
AS
DECLARE @CurrID int

-- see if the record already exists
SELECT @CurrID = AuthMat
FROM BK_AuthorMaterial
WHERE AuthID = @AuthID AND
    MatID = @MatID

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_AuthorMaterial(
        AuthID,
        MatID,
        Sequence,
        AuthMatLastUpdate
    )
    VALUES(
        @AuthID,
        @MatID,
        @Sequence,
        getdate()
    )
    SET @AuthMat = @@IDENTITY
    IF @@ERROR > 0
    BEGIN
        RAISERROR ('Insert of Bundle failed', 16, 1)
        RETURN 99
    END
END
```

```
END
ELSE -- indicate record already exists
BEGIN
    SET @AuthMat = -1
END
GO

CREATE PROC BK_UpdateAuthorMaterial
@AuthMat int,
@AuthID int,
@MatID int,
@Sequence int
AS
UPDATE BK_AuthorMaterial
SET AuthID = @AuthID,
    MatID = @MatID,
    Sequence = @Sequence,
    AuthMatLastUpdate = getdate()
WHERE AuthMat = @AuthMat
GO

CREATE PROCEDURE BK_DeleteAuthorMaterial
@AuthMat int
AS
DELETE FROM BK_AuthorMaterial
WHERE AuthMat = @AuthMat
GO

CREATE PROC BK_GetCategoriesPerMatID
@CategoryMatID int
AS
SELECT c.CatID, c.CatName, ISNULL(mc.MatID, 0) Selected
FROM BK_Category c
LEFT JOIN (
    SELECT * FROM BK_MaterialCategory
    WHERE MatID = @CategoryMatID) mc
ON mc.CatID = c.CatID
ORDER BY c.CatName
GO

CREATE PROC BK_GetMaterialsPerCatID
@CatID int
AS
SELECT mc.MatID, m.MatTitle
FROM BK_Material m
JOIN BK_MaterialCategory mc
ON m.MatID = mc.MatID
WHERE mc.CatID = @CatID
ORDER BY m.MatTitle
GO

CREATE PROC BK_GetMaterialCategories
AS
SELECT mc.MatID, c.CatID, c.CatName
FROM BK_Category c
JOIN BK_MaterialCategory mc
ON c.CatID = mc.CatID
GROUP BY mc.MatID, c.CatID, c.CatName
ORDER BY mc.MatID
GO

CREATE PROC BK_InsertMaterialCategory
@MatCatID int OUTPUT,
@MatID int,
@CatID int
AS
DECLARE @CurrID int
```

```
-- see if the record already exists
SELECT @CurrID = MatCatID
FROM BK_MaterialCategory
WHERE MatID = @MatID AND
      CatID = @CatID

-- if not, add it
IF @CurrID IS NULL
    BEGIN
        INSERT INTO BK_MaterialCategory(
            MatID,
            CatID,
            MatCatLastUpdate
        )
        VALUES(
            @MatID,
            @CatID,
            getdate()
        )
        SET @MatCatID = @@IDENTITY
        IF @@ERROR > 0
            BEGIN
                RAISERROR ('Insert of Bundle failed', 16, 1)
                RETURN 99
            END
        END
    ELSE -- indicate record already exists
        BEGIN
            SET @MatCatID = -1
        END
GO

CREATE PROC BK_UpdateMaterialCategory
@MatCatID int,
@MatID int,
@CatID int
AS
UPDATE BK_MaterialCategory
SET MatID = @MatID,
    CatID = @CatID,
    MatCatLastUpdate = getdate()
WHERE MatCatID = @MatCatID
GO

CREATE PROCEDURE BK_DeleteMaterialCategory
@MatCatID int
AS
DELETE FROM BK_MaterialCategory
WHERE MatCatID = @MatCatID
GO

CREATE PROC BK_GetBundles
@ParentMatID int
AS
SELECT * FROM BK_BundledMaterial
WHERE ParentMatID = @ParentMatID
GO

CREATE PROC BK_InsertBundle
@BundMat int OUTPUT,
@ParentMatID int,
@ChildMatID int,
@Sequence int
AS
DECLARE @CurrID int

-- see if the record already exists
```

```

SELECT @CurrID = BundMat
FROM BK_BundledMaterial
WHERE ParentMatID = @ParentMatID AND
      ChildMatID = @ChildMatID

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_BundledMaterial(
        ParentMatID,
        ChildMatID,
        Sequence,
        BundMatLastUpdate
    )
    VALUES(
        @ParentMatID,
        @ChildMatID,
        @Sequence,
        getdate()
    )
    SET @BundMat = @@IDENTITY
    IF @@ERROR > 0
    BEGIN
        RAISERROR ('Insert of Bundle failed', 16, 1)
        RETURN 99
    END
END
ELSE -- indicate record already exists
BEGIN
    SET @BundMat = -1
END
GO

CREATE PROC BK_UpdateBundle
@BundMat int,
@ParentMatID int,
@ChildMatID int,
@Sequence int
AS
UPDATE BK_BundledMaterial
SET ParentMatID = @ParentMatID,
    ChildMatID = @ChildMatID,
    Sequence = @Sequence,
    BundMatLastUpdate = getdate()
WHERE BundMat = @BundMat
GO

CREATE PROCEDURE BK_DeleteBundle
@BundMat int
AS
DELETE FROM BK_BundledMaterial
WHERE BundMat = @BundMat
GO

CREATE PROC BK_GetUrls
AS
SELECT m.MatID, m.MatTitle, m.MatDatePub
FROM BK_Material m
JOIN BK_Url u
    ON m.MatID = u.MatID
GO

CREATE PROC BK_InsertUrl
@MatID int,
@Url varchar(255)
AS
DECLARE @CurrID int

```

```
-- see if the record already exists
SELECT @CurrID = MatID
FROM BK_Url
WHERE Url = @Url

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_Url(
        MatID,
        Url,
        UrlLastUpdate
    )
    VALUES(
        @MatID,
        @Url,
        getdate()
    )
    IF @@ERROR > 0
    BEGIN
        RAISERROR ('Insert of Url failed', 16, 1)
        RETURN 99
    END
    RETURN 1
END
ELSE -- indicate record already exists
BEGIN
    RETURN 0
END
GO

CREATE PROC BK_UpdateUrl
@MatID int,
@Url varchar(255)
AS
UPDATE BK_Url
SET Url = @Url,
    UrlLastUpdate = getdate()
WHERE MatID = @MatID
GO

CREATE PROCEDURE BK_DeleteUrl
@MatID int
AS
DELETE FROM BK_Url
WHERE MatID = @MatID
GO

CREATE PROC BK_GetBooks
AS
SELECT b.MatID, m.MatTitle, m.MatDatePub
FROM BK_Book b
JOIN BK_Material m
ON m.MatID = b.MatID
GO

CREATE PROC BK_InsertBook
@MatID int,
@ISBN varchar(20),
@PubID int,
@CityID int
AS
DECLARE @CurrID int

-- see if the record already exists
SELECT @CurrID = MatID
```

```

FROM      BK_Book
WHERE     ISBN = @ISBN

-- if not, add it
IF @CurrID IS NULL
    BEGIN
        INSERT INTO BK_Book(
            MatID,
            ISBN,
            PubID,
            CityID,
            BookLastUpdate
        )
        VALUES(
            @MatID,
            @ISBN,
            @PubID,
            @CityID,
            getdate()
        )
        IF @@ERROR > 0
            BEGIN
                RAISERROR ('Insert of Book failed', 16, 1)
                RETURN 99
            END
        RETURN 1
    END
ELSE -- indicate record already exists
    BEGIN
        RETURN 0
    END
END
GO

CREATE PROC BK_UpdateBook
@MatID int,
@ISBN varchar(20),
@PubID int,
@CityID int
AS
UPDATE BK_Book
SET ISBN = @ISBN,
    PubID = @PubID,
    CityID = @CityID,
    BookLastUpdate = getdate()
WHERE MatID = @MatID
GO

CREATE PROCEDURE BK_DeleteBook
@MatID int
AS
DELETE FROM BK_Book
WHERE MatID = @MatID
GO

CREATE PROC BK_GetArticles
AS
SELECT m.MatID, m.MatTitle, m.MatDatePub
FROM BK_Material m
JOIN BK_Article a
    ON m.MatID = a.MatID
GO

CREATE PROC BK_InsertArticle
@MatID int,
@JournalID int,
@ArtVol int,
@ArtNumber int,
@ArtStartPage int,

```

```
@ArtEndPage int
AS
DECLARE @CurrID int

-- see if the record already exists
SELECT @CurrID = MatID
FROM BK_Article
WHERE JournalID = @JournalID AND
      ArtVol = @ArtVol AND
      ArtNumber = @ArtNumber AND
      ArtStartPage = @ArtStartPage

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_Article(
        MatID,
        JournalID,
        ArtVol,
        ArtNumber,
        ArtStartPage,
        ArtEndPage,
        ArtLastUpdate
    )
    VALUES(
        @MatID,
        @JournalID,
        @ArtVol,
        @ArtNumber,
        @ArtStartPage,
        @ArtEndPage,
        getdate()
    )
    IF @@ERROR > 0
    BEGIN
        RAISERROR ('Insert of Article failed', 16, 1)
        RETURN 99
    END
    RETURN 1
END
ELSE -- indicate record already exists
BEGIN
    RETURN 0
END
END

GO

CREATE PROC BK_UpdateArticle
@MatID int,
@JournalID int,
@ArtVol int,
@ArtNumber int,
@ArtStartPage int,
@ArtEndPage int
AS
UPDATE BK_Article
SET JournalID = @JournalID,
    ArtVol = @ArtVol,
    ArtNumber = @ArtNumber,
    ArtStartPage = @ArtStartPage,
    ArtEndPage = @ArtEndPage,
    ArtLastUpdate = getdate()
WHERE MatID = @MatID
GO

CREATE PROCEDURE BK_DeleteArticle
@MatID int
AS
DELETE FROM BK_Article
```



```

WHERE MatID = @MatID
GO

CREATE PROC BK_GetMaterials
AS
SELECT * FROM BK_Material
GO

CREATE PROC BK_GetMaterialByID
@MatID int
AS
SELECT m.MatID, m.MatTitle, m.MatDesc, m.MatDatePub,
       a.JournalID, j.JournalName, a.ArtVol, a.ArtNumber, a.ArtStartPage, a.ArtEndPage,
       b.ISBN, b.PubID, p.PubName, b.CityID, c.City,
       u.Url
FROM BK_Material m
LEFT JOIN BK_Article a
    ON a.MatID = m.MatID
LEFT JOIN BK_Journal j
    ON j.JournalID = a.JournalID
LEFT JOIN BK_Book b
    ON m.MatID = b.MatID
LEFT JOIN BK_Publisher p
    ON b.PubID = p.PubID
LEFT JOIN BK_City c
    ON c.CityID = b.CityID
LEFT JOIN BK_Url u
    ON m.MatID = u.MatID
WHERE m.MatID = @MatID
GO

CREATE PROC BK_GetMaterialTitle
@MatID int,
@MatTitle varchar(250) OUTPUT
AS
SELECT @MatTitle = MatTitle FROM BK_Material
WHERE MatID = @MatID
GO

CREATE PROC BK_InsertMaterial
@MatID int OUTPUT,
@MatTitle varchar(250),
@MatDatePub datetime,
@MatDesc text
AS
DECLARE @CurrID int

-- see if the record already exists
SELECT @CurrID = MatID
FROM BK_Material
WHERE MatTitle = @MatTitle AND
       MatDatePub = @MatDatePub

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_Material(
        MatTitle,
        MatDatePub,
        MatDesc,
        MatLastUpdate
    )
    VALUES(@MatTitle,
            @MatDatePub,
            @MatDesc,
            getdate())
    SET @MatID = @@IDENTITY

```

```
        IF @@ERROR > 0
        BEGIN
            RAISERROR ('Insert of Material failed', 16, 1)
            RETURN 99
        END
    END
ELSE -- indicate record already exists
    BEGIN
        SET @MatID = -1
    END
GO

CREATE PROC BK_UpdateMaterial
@MatID int OUTPUT,
@MatTitle varchar(250),
@MatDatePub datetime,
@MatDesc text
AS
UPDATE BK_Material
SET MatTitle = @MatTitle,
    MatDatePub = @MatDatePub,
    MatDesc = @MatDesc,
    MatLastUpdate = getdate()
WHERE MatID = @MatID
GO

CREATE PROCEDURE BK_DeleteMaterial
@MatID int
AS
DELETE FROM BK_Material
WHERE MatID = @MatID
GO

CREATE PROC BK_GetJournals
AS
SELECT * FROM BK_Journal
GO

CREATE PROC BK_InsertJournal
@JournalID int OUTPUT,
@JournalName varchar(100)
AS
DECLARE @CurrID int

-- see if the journal already exists
SELECT @CurrID = JournalID
FROM BK_Journal
WHERE JournalName = @JournalName

-- if not, add it
IF @CurrID IS NULL
    BEGIN
        INSERT INTO BK_Journal(
            JournalName,
            JournalLastUpdate
        )
        VALUES(@JournalName,
            getdate())
        SET @JournalID = @@IDENTITY

        IF @@ERROR > 0
        BEGIN
            RAISERROR ('Insert of Journal failed', 16, 1)
            RETURN 99
        END
    END
END
```

```
ELSE -- indicate journal already exists
BEGIN
    SET @JournalID = -1
END
GO

CREATE PROC BK_UpdateJournal
@JournalID int,
@JournalName varchar(100)
AS
UPDATE BK_Journal
SET JournalName = @JournalName,
    JournalLastUpdate = getdate()
WHERE JournalID = @JournalID
GO

CREATE PROCEDURE BK_DeleteJournal
@JournalID int
AS
DELETE FROM BK_Journal
WHERE JournalID = @JournalID
GO

CREATE PROC BK_GetAuthors
AS
SELECT * FROM BK_Author
ORDER BY AuthLastName
GO

CREATE PROCEDURE BK_GetAuthorNames AS
SELECT AuthID,
    AuthLastName + ', ' + AuthFirstName AS InvertedName,
    AuthFirstName + ' ' + AuthLastName AS FullName
FROM BK_Author
GO

CREATE PROC BK_InsertAuthor
@AuthID int OUTPUT,
@AuthFirstName varchar(25),
@AuthLastName varchar(25),
@AuthBio varchar(250)
AS
DECLARE @CurrID int

-- see if the author already exists
SELECT @CurrID = AuthID
FROM BK_Author
WHERE AuthFirstName = @AuthFirstName AND
    AuthLastName = @AuthLastName

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_Author(
        AuthFirstName,
        AuthLastName,
        AuthBio,
        AuthLastUpdate)
    VALUES(
        @AuthFirstName,
        @AuthLastName,
        @AuthBio,
        getdate())
    SET @AuthID = @@IDENTITY
```

```
        IF @@ERROR > 0
        BEGIN
            RAISERROR ('Insert of Author failed', 16, 1)
            RETURN 99
        END
    END
ELSE -- indicate author already exists
    BEGIN
        SET @AuthID = -1
    END
GO

CREATE PROC BK_UpdateAuthor
@AuthID int,
@AuthFirstName varchar(25),
@AuthLastName varchar(25),
@AuthBio varchar(250)
AS
UPDATE BK_Author
SET AuthFirstName = @AuthFirstName,
    AuthLastName = @AuthLastName,
    AuthBio = @AuthBio,
    AuthLastUpdate = getdate()
WHERE AuthID = @AuthID
GO

CREATE PROCEDURE BK_DeleteAuthor
@AuthID int
AS
DELETE FROM BK_Author
WHERE AuthID = @AuthID
GO

CREATE PROC BK_GetCategories
AS
SELECT * FROM BK_Category
GO

CREATE PROC BK_GetCategory
@CatID int,
@CatName varchar(75) OUTPUT
AS
SELECT @CatName = CatName
FROM BK_Category
WHERE CatID = @CatID
GO

CREATE PROC BK_InsertCategory
@CatID int OUTPUT,
@CatName varchar(75),
@CatDesc varchar(265)
AS
DECLARE @CurrID int

-- see if the city already exists
SELECT @CurrID = CatID
FROM BK_Category
WHERE CatName = @CatName

-- if not, add it
IF @CurrID IS NULL
    BEGIN
        INSERT INTO BK_Category(CatName, CatDesc, CatLastUpdate)
        VALUES(@CatName, @CatDesc, getdate())
        SET @CatID = @@IDENTITY
    END
```

```
        IF @@ERROR > 0
        BEGIN
            RAISERROR ('Insert of Category failed', 16, 1)
            RETURN 99
        END
    END
ELSE -- indicate city already exists
    BEGIN
        SET @CatID = -1
    END
GO

CREATE PROC BK_UpdateCategory
@CatID int,
@CatName varchar(75),
@CatDesc varchar(265)
AS
UPDATE BK_Category
SET CatName = @CatName,
    CatDesc = @CatDesc,
    CatLastUpdate = getdate()
WHERE CatID = @CatID
GO

CREATE PROCEDURE BK_DeleteCategory
@CatID int
AS
DELETE FROM BK_Category
WHERE CatID = @CatID
GO

CREATE PROC BK_GetPublishers
AS
SELECT * FROM BK_Publisher
GO

CREATE PROC BK_InsertPublisher
@PubID int OUTPUT,
@PubName varchar(150)
AS
DECLARE @CurrID int

-- see if the Publisher already exists
SELECT @CurrID = PubID
FROM BK_Publisher
WHERE PubName = @PubName

-- if not, add it
IF @CurrID IS NULL
    BEGIN
        INSERT INTO BK_Publisher(PubName, PubLastUpdate)
        VALUES(@PubName, getdate())
        SET @PubID = @@IDENTITY

        IF @@ERROR > 0
        BEGIN
            RAISERROR ('Insert of Publisher failed', 16, 1)
            RETURN 99
        END
    END
ELSE -- indicate Publisher already exists
    BEGIN
        SET @PubID = -1
    END
GO
```

```
CREATE PROC BK_UpdatePublisher
@PubID int,
@PubName varchar(150)
AS
UPDATE BK_Publisher
SET PubName = @PubName,
    PubLastUpdate = getdate()
WHERE PubID = @PubID
GO

CREATE PROCEDURE BK_DeletePublisher
@PubID int
AS
DELETE FROM BK_Publisher
WHERE PubID = @PubID
GO

CREATE PROC BK_GetCities
AS
SELECT * FROM BK_City
GO

CREATE PROC BK_InsertCity
@CityID int OUTPUT,
@City varchar(50)
AS
DECLARE @CurrID int

-- see if the city already exists
SELECT @CurrID = CityID
FROM BK_City
WHERE City = @City

-- if not, add it
IF @CurrID IS NULL
BEGIN
    INSERT INTO BK_City(City, CityLastUpdate)
    VALUES(@City, getdate())
    SET @CityID = @@IDENTITY

    IF @@ERROR > 0
    BEGIN
        RAISERROR ('Insert of City failed', 16, 1)
        RETURN 99
    END
END
ELSE -- indicate city already exists
BEGIN
    SET @CityID = -1
END
GO

CREATE PROC BK_UpdateCity
@CityID int,
@City varchar(50)
AS
UPDATE BK_City
SET City = @City,
    CityLastUpdate = getdate()
WHERE CityID = @CityID
GO

CREATE PROCEDURE BK_DeleteCity
@CityID int
AS
DELETE FROM BK_City
WHERE CityID = @CityID
```

```

<%@ Page language="c#" Codebehind="Add.aspx.cs" AutoEventWireup="false" validateRequest=
false Inherits="Sharon.Books.Add" %>
    <asp:panel id="MatPanel" runat="server">
        <P>Add a new Material</P>
        <P>
            <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="100%" border
="1">
                <TR>
                    <TD>Title</TD>
                    <TD>
                        <asp:TextBox id="txtTitle" runat="server" Width="100%"></
asp:TextBox></TD>
                </TR>
                <TR>
                    <TD>Date of Publication</TD>
                    <TD>
                        <asp:DropDownList id="ddlMonths" runat="server"></asp:
DropDownList>
                        <asp:DropDownList id="ddlDays" runat="server"></asp:
DropDownList>
                        <asp:TextBox id="txtYear" runat="server"></asp:TextBox></
TD>
                    </TR>
                <TR>
                    <TD>Type</TD>
                    <TD>
                        <asp:DropDownList id="ddlType" runat="server"></asp:
DropDownList></TD>
                </TR>
                <TR>
                    <TD>Description</TD>
                    <TD>
                        <asp:TextBox id="txtDesc" runat="server" Width="100%"
TextMode="MultiLine" Rows="8"></asp:TextBox></TD>
                </TR>
                <TR>
                    <TD>Categories</TD>
                    <TD>
                        <asp:CheckBoxList id="cbCategories" CssClass="checkBoxList"
" runat="server" RepeatColumns="4" CellPadding="10" CellSpacing="0"></asp:
CheckBoxList></TD>
                </TR>
                <TR>
                    <TD></TD>
                    <TD>
                        <asp:Button id="btnMatNext" runat="server" Text="Next"></
asp:Button></TD>
                </TR>
            </TABLE>
        </P>
    </asp:panel><asp:panel id="ArtPanel" runat="server" Visible="False">
        <P>Article Details</P>
        <P>
            <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="100%" border
="1">
                <TR>
                    <TD>Journal</TD>
                    <TD>
                        <asp:DropDownList id="ddlJournal" runat="server"></asp:
DropDownList>&nbsp;or
                        enter
                        <asp:TextBox id="txtJournal" runat="server"></asp:TextBox>
                    </TD>
                </TR>
                <TR>
                    <TD>Volume</TD>
                    <TD>

```

```

        <asp:TextBox id="txtVol" runat="server"></asp:TextBox></
TD>
        </TR>
        <TR>
            <TD>Number</TD>
            <TD>
                <asp:TextBox id="txtNum" runat="server"></asp:TextBox></
TD>
            </TR>
            <TR>
                <TD>Pages</TD>
                <TD>
                    <asp:TextBox id="txtStart" runat="server"></asp:TextBox>&
nbsp;to
                    <asp:TextBox id="txtEnd" runat="server"></asp:TextBox></
TD>
                </TR>
                <TR>
                    <TD></TD>
                    <TD>
                        <asp:Button id="btnArtNext" runat="server" Text="Next"></
asp:Button></TD>
                    </TR>
                </TABLE>
            </P>
            </asp:panel><asp:panel id="BookPanel" runat="server" Visible="False">
                <P>Book Details</P>
                <TABLE id="Table3" cellSpacing="1" cellPadding="1" width="100%" border="1">
                    <TR>
                        <TD>ISBN</TD>
                        <TD>
                            <asp:TextBox id="txtISBN" runat="server"></asp:TextBox></TD>
                        </TR>
                        <TR>
                            <TD style="HEIGHT: 18px">Publisher</TD>
                            <TD style="HEIGHT: 18px">
                                <asp:DropDownList id="ddlPub" runat="server"></asp:
DropDownList>&nbsp;or enter
                                <asp:TextBox id="txtPub" runat="server"></asp:TextBox></TD>
                            </TR>
                            <TR>
                                <TD style="HEIGHT: 18px">City</TD>
                                <TD style="HEIGHT: 18px">
                                    <asp:DropDownList id="ddlCity" runat="server"></asp:
DropDownList>&nbsp;or enter
                                    <asp:TextBox id="txtCity" runat="server"></asp:TextBox></TD>
                                </TR>
                                <TR>
                                    <TD></TD>
                                    <TD>
                                        <asp:Button id="btnBookNext" runat="server" Text="Next"></asp:
Button></TD>
                                    </TR>
                                </TABLE>
                            </asp:panel><asp:panel id="UrlPanel" runat="server" Visible="False">
                                <P>Url Details</P>
                                <P>
                                    <TABLE id="Table4" cellSpacing="1" cellPadding="1" width="100%" border
="1">
                                        <TR>
                                            <TD>Url</TD>
                                            <TD>
                                                <asp:TextBox id="txtAddress" runat="server"></asp:TextBox>
                                            </TD>
                                        </TR>
                                        <TR>

```



```

        <TD></TD>
        <TD>
            <asp:Button id="btnUrlNext" runat="server" Text="Next"></
asp:Button></TD>
        </TR>
    </TABLE>
</P>
</asp:panel><asp:panel id="AuthorPanel" runat="server" Visible="False">
    <P>Author Details</P>
    <P>
        <TABLE id="Table5" cellSpacing="1" cellPadding="1" width="100%" border
="1">
            <TR>
                <TD colspan="2">Add a new Author Here</TD>
                <TD>or Select an Author<BR>
                    from the ListBox Here</TD>
                <TD>&nbsp;<P>Assigned Authors</P>
                </TD>
            </TR>
            <TR>
                <TD>First Name</TD>
                <TD>
                    <asp:textbox id="txtFirstName" runat="server"></asp:
textbox></TD>
                <TD rowspan="3">
                    <P>
                        <asp:ListBox id="lbAuthors" runat="server"
SelectionMode="Multiple"></asp:ListBox></P>
                    </TD>
                <TD rowspan="3">
                    <P>
                        <asp:ListBox id="lbAuthorSelected" runat="server"
SelectionMode="Multiple"></asp:ListBox></P>
                    </TD>
            </TR>
            <TR>
                <TD>Last Name</TD>
                <TD>
                    <asp:textbox id="txtLastName" runat="server"></asp:
textbox></TD>
            </TR>
            <TR>
                <TD>Bio</TD>
                <TD>
                    <asp:textbox id="txtBio" runat="server" TextMode="
MultiLine"></asp:textbox></TD>
            </TR>
            <TR>
                <TD></TD>
                <TD>
                    <asp:button id="btnAuthorAdd" runat="server" Text="Add
Author"></asp:button></TD>
                <TD>
                    <asp:Button id="btnAuthSelect" runat="server" Text="Select
Author"></asp:Button></TD>
                <TD>
                    <asp:Button id="btnAuthRemove" runat="server" Text="Remove
"></asp:Button>
                    <asp:Button id="btnAuthorNext" runat="server" Text="Next">
</asp:Button></TD>
            </TR>
        </TABLE>
    </P>
</asp:panel><asp:panel id="SeqPanel" runat="server" Visible="False">
    <P>Sequence Details</P>
    <asp:DataGrid id="dgSeq" runat="server" AutoGenerateColumns="False">

```

```
        <Columns>
            <asp:BoundColumn DataField="MatID" HeaderText="Material ID"
ReadOnly="True"></asp:BoundColumn>
            <asp:BoundColumn DataField="AuthID" HeaderText="Author ID"
ReadOnly="True"></asp:BoundColumn>
            <asp:BoundColumn DataField="FullName" HeaderText="Author" ReadOnly
="True"></asp:BoundColumn>
            <asp:BoundColumn DataField="AuthMat" HeaderText="Author Material
ID" ReadOnly="True"></asp:BoundColumn>
            <asp:BoundColumn DataField="Sequence"></asp:BoundColumn>
            <asp:EditCommandColumn ButtonType="LinkButton" CancelText="Cancel"
EditText="Edit" UpdateText="Update"></asp:EditCommandColumn>
        </Columns>
    </asp:DataGrid>
    <P><a href="Add.aspx">Add Another!</a></P>
</asp:panel>
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Add.
    /// </summary>
    public class Add : Sharon.BooksPage
    {
        protected CheckBoxList cbCategories;
        protected TextBox txtDesc;
        protected DropDownList ddlType;
        protected TextBox txtYear;
        protected DropDownList ddlDays;
        protected DropDownList ddlMonths;
        protected TextBox txtTitle;

        private string[] months = {
            "Jan", "Feb", "Mar", "Apr",
            "May", "Jun", "Jul", "Aug",
            "Sep", "Oct", "Nov", "Dec"
        };

        protected Panel MatPanel;
        protected Button btnMatNext;
        protected Panel ArtPanel;
        protected Button btnArtNext;
        protected TextBox txtEnd;
        protected TextBox txtStart;
        protected TextBox txtNum;
        protected TextBox txtVol;
        protected TextBox txtJournal;
        protected DropDownList ddlJournal;
        protected Panel BookPanel;
        protected Button btnBookNext;
        protected TextBox txtCity;
        protected DropDownList ddlCity;
        protected TextBox txtPub;
        protected DropDownList ddlPub;
        protected TextBox txtISBN;
        protected Panel UrlPanel;
        protected Button btnUrlNext;
        protected TextBox txtAddress;
        protected Panel AuthorPanel;
        protected TextBox txtBio;
        protected TextBox txtLastName;
        protected TextBox txtFirstName;
        protected ListBox lbAuthors;
        protected Button btnAuthorAdd;
        protected Button btnAuthSelect;
        protected Button btnAuthorNext;
        protected Panel SeqPanel;
        protected ListBox lbAuthorSelected;
        protected System.Web.UI.WebControls.DataGrid dgSeq;
        protected System.Web.UI.WebControls.Button btnAuthRemove;
        private int[] days;

        private void Page_Load(object sender, System.EventArgs e)
```

```

    {
        if(!IsPostBack)
        {
            Title = "Add a Material";
            LoadMatPanel();
        }
    }
    private void LoadMatPanel()
    {
        MatPanel.Visible = true;
        days = new int[31];
        for(int day = 0; day < 31;)
            days[day] = ++day;

        ddlDays.DataSource = days;
        ddlMonths.DataSource = months;
        string[] types = {"Article", "Book", "Url"};
        ddlType.DataSource = types;
        ddlType.DataBind();
        ddlDays.DataBind();
        ddlMonths.DataBind();
        Category cat = new Category();
        DataSet dsCat = cat.GetCategories();
        cbCategories.DataSource =
            dsCat.Tables[0].DefaultView;
        cbCategories.DataTextField = "CatName";
        cbCategories.DataValueField = "CatID";
        cbCategories.DataBind();
    } //end LoadCategories

```

Web Form Designer generated code

```

private void btnMatNext_Click(object sender, System.EventArgs e)
{
    DateTime date = new DateTime(
        Int32.Parse(txtYear.Text),
        ddlMonths.SelectedIndex + 1,
        ddlDays.SelectedIndex + 1);
    Material mat = new Material(
        txtTitle.Text.Trim(),
        txtDesc.Text.Trim(),
        date);

    ArrayList categories = new ArrayList();
    IEnumerator en =
        cbCategories.Items.GetEnumerator();
    while(en.MoveNext())
    {
        ListItem item = (ListItem)en.Current;
        if(item.Selected)
            categories.Add(item.Value);
    }
    Session["Categories"] = categories;
    Session["Material"] = mat;
    MatPanel.Visible = false;
    ViewState["Type"] = ddlType.SelectedItem.Text;
    switch (ViewState["Type"].ToString())
    {
        case "Article":
            LoadArtPanel();
            break;
        case "Book":
            LoadBookPanel();
            break;
        case "Url" :
            LoadUrlPanel();

```

```

        break;
    } //end switch
} //end btnMatNext_Click

private void LoadArtPanel()
{
    ArtPanel.Visible = true;
    Journal j = new Journal();
    ddlJournal.DataSource = j.GetJournals();
    ddlJournal.DataTextField = "JournalName";
    ddlJournal.DataValueField = "JournalID";
    ddlJournal.DataBind();
}

private void LoadBookPanel()
{
    BookPanel.Visible = true;
    City cities = new City();
    ddlCity.DataSource =
        cities.GetCities();
    Tables[0].DefaultView;
    ddlCity.DataTextField = "City";
    ddlCity.DataValueField = "CityID";

    Publisher pubs = new Publisher();
    ddlPub.DataSource =
        pubs.GetPublishers();
    Tables[0].DefaultView;
    ddlPub.DataTextField = "PubName";
    ddlPub.DataValueField = "PubID";
    DataBind();
}

private void LoadUrlPanel()
{
    UrlPanel.Visible = true;
}

private void LoadAuthorPanel()
{
    AuthorPanel.Visible = true;
    Author a = new Author();
    lbAuthors.DataSource =
        a.GetAuthorNames().Tables[0].DefaultView;
    lbAuthors.DataTextField =
        "InvertedName";
    lbAuthors.DataValueField =
        "AuthID";
    lbAuthors.DataBind();
}

private void LoadSeqPanel()
{
    SeqPanel.Visible = true;
    MaterialAuthor ma =
        new MaterialAuthor();
    DataSet maAuthors =
        ma.GetAuthorsPerMatID((int)Session["MatID"]);
    dgSeq.DataSource =
        maAuthors.Tables[0].DefaultView;
    dgSeq.DataBind();
}

private void btnArtNext_Click(object sender, System.EventArgs e)
{
    int journalID, vol, num, start, end;
    journalID = -1;
    if(txtJournal.Text.Length > 0)

```

```

    {
        //user typed value
        Journal j = new Journal();
        journalID =
            j.Add(txtJournal.Text.Trim());
    }
    else if(ddlJournal.SelectedIndex >= 0)
    {
        journalID =
            Int32.Parse(
                ddlJournal.SelectedItem.Value);
    }
    vol = Int32.Parse(txtVol.Text.Trim());
    num = Int32.Parse(txtNum.Text.Trim());
    start = Int32.Parse(txtStart.Text.Trim());
    end = Int32.Parse(txtEnd.Text.Trim());
    Article art =
        new Article(journalID, vol, num, start, end);
    Session["Article"] = art;
    ArtPanel.Visible = false;
    LoadAuthorPanel();
}

private void btnBookNext_Click(object sender, System.EventArgs e)
{
    int cityID = 0, pubID = 0;
    string isbn = txtISBN.Text.Trim();
    string city = txtCity.Text.Trim();
    string pub = txtPub.Text.Trim();
    if(city.Length > 0)
    {
        //user typed in City
        City c = new City();
        cityID = c.Add(city);
    }
    else if(ddlCity.SelectedIndex >= 0)
    {
        cityID = Int32.Parse(
            ddlCity.SelectedItem.Value);
    }
    if(pub.Length > 0)
    {
        //user typed in Publisher
        Publisher p = new Publisher();
        pubID = p.Add(pub);
    }
    else if(ddlPub.SelectedIndex >= 0)
    {
        pubID = Int32.Parse(
            ddlPub.SelectedItem.Value);
    }
    Book book = new Book(pubID, cityID, isbn);
    Session["Book"] = book;
    BookPanel.Visible = false;
    LoadAuthorPanel();
}

private void btnUrlNext_Click(object sender, System.EventArgs e)
{
    Url url = new Url(txtAddress.Text.Trim());
    Session["Url"] = url;
    UrlPanel.Visible = false;
    LoadAuthorPanel();
}

private void btnAuthorAdd_Click(object sender, System.EventArgs e)
{

```

```

        string bio, first, last;
        if(txtBio.Text.Trim().Length == 0)
            bio = "No data available.";
        else
            bio = txtBio.Text.Trim();
        first = txtFirstName.Text.Trim();
        last = txtLastName.Text.Trim();
        Author a = new Author();
        int authID = a.Add(first, last, bio);
        if( authID > 0)
        {
            //add successful
            //reload lbAuthors
            LoadAuthorPanel();
        }
    }

    private void btnAuthSelect_Click(object sender, System.EventArgs e)
    {
        if(lbAuthors.SelectedItem != null)
        {
            lbAuthorSelected.Items.Add
                (lbAuthors.SelectedItem);
        }
    }
    //end btnAuthSelect_Click

    private void btnAuthRemove_Click(object sender, System.EventArgs e)
    {
        lbAuthorSelected.Items.Remove
            (lbAuthorSelected.SelectedItem);
        lbAuthorSelected.ClearSelection();
    }
    //end btnAuthRemove_Click

    private void btnAuthorNext_Click(object sender, System.EventArgs e)
    {
        Material mat = (Material)Session["Material"];
        //Add Material to Database
        int matID = mat.Add();
        if(matID > 0)
        {
            Session["MatID"] = matID;
            switch (ViewState["Type"].ToString())
            {
                //Add type details
                case "Article":
                    Article art =
                        (Article)Session["Article"];
                    art.MatID = matID;
                    art.Add();
                    break;
                case "Book":
                    Book book = (Book)Session["Book"];
                    book.MatID = matID;
                    book.Add();
                    break;
                case "Url" :
                    Url url = (Url)Session["Url"];
                    url.MatID = matID;
                    url.Add();
                    break;
            }
            //end switch

            //associate Categories to Material
            ArrayList cats =
                (ArrayList)Session["Categories"];
            MaterialCategory mc =
                new MaterialCategory();

```

```

        IEnumerator iter = cats.GetEnumerator();
        while(iter.MoveNext())
        {
            int catID = Int32.Parse(iter.Current.ToString());
            mc.Add(matID, catID);
        }

        //associate Authors to Material
        iter = lbAuthorSelected.Items.
            GetEnumerator();
        MaterialAuthor ma =
            new MaterialAuthor();
        while(iter.MoveNext())
        {
            int authID = Int32.Parse(
                ((ListItem)iter.Current).Value);
            ma.Add(matID, authID, 0);
        } //end foreach
    } //end if
    AuthorPanel.Visible = false;
    LoadSeqPanel();
} //end btnAuthorNext_Click

private void dgSeq_CancelCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
{
    dgSeq.EditItemIndex = -1;
    LoadSeqPanel();
}

private void dgSeq_EditCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
{
    dgSeq.EditItemIndex =
        e.Item.ItemIndex;
    LoadSeqPanel();
}

private void dgSeq_UpdateCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
{
    int matID, authMat, authID, sequence;
    TextBox txtSeq;
    matID = Int32.Parse(e.Item.Cells[0].Text);
    authID = Int32.Parse(e.Item.Cells[1].Text);
    authMat = Int32.Parse(e.Item.Cells[3].Text);
    txtSeq = (TextBox)e.Item.Cells[4].Controls[0];
    sequence = Int32.Parse(txtSeq.Text);

    MaterialAuthor ma = new MaterialAuthor();
    ma.Update(authMat, matID, authID, sequence);
    dgSeq.EditItemIndex = -1;
    LoadSeqPanel();
}
} //end class
} //end namespace

```



```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Articles.
    /// </summary>
    public class Articles : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.Repeater rp;
        protected Label lblMessage;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                Article art = new Article();
                DataSet ds = art.GetArticles();
                rp.DataSource =
                    ds.Tables[0].DefaultView;
                rp.DataBind();
                Title = "Articles";

                int count = ds.Tables[0].Rows.Count;
                if(count ==0)
                    lblMessage.Text = "No records found.";
                else if (count == 1)
                    lblMessage.Text = "One record found.";
                else
                    lblMessage.Text = count + " records found.";
            }
        }

        Web Form Designer generated code
    }
}
```

```

<%@ Page language="c#" Codebehind="Articles.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Articles" %>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <p></p>
    <asp:Repeater id="rp" runat="server">
    <HeaderTemplate>
        <table>
            <tr>
                <th colspan="2">
                    All Articles</th></tr>
        </HeaderTemplate>
    <ItemTemplate>
        <tr>
            <td valign="top">
                <%# DataBinder.Eval(Container.DataItem, "MatID") %>
            </td>
            <td>
                <a href='<%# "Details.aspx?MatID=" + DataBinder.Eval(Container.
DataItem, "MatID") %>'><%# DataBinder.Eval(Container.DataItem, "MatTitle") %></a>
                <br />
                <%# DataBinder.Eval(Container.DataItem, "MatDatePub") %>
            </td>
        </tr>
    </ItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>

```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Authors.
    /// </summary>
    public class Authors : Sharon.BooksPage
    {
        protected TextBox txtFirstName;
        protected TextBox txtLastName;
        protected TextBox txtBio;
        protected Button btnAdd;
        protected DataList dlAuth;
        protected Label lblMessage;
        protected System.Web.UI.WebControls.ListBox lbSelected;
        protected System.Web.UI.WebControls.Button btnRemove;
        private int matID;
        protected System.Web.UI.WebControls.Button btnSequence;
        private string matTitle;
        protected Panel panSelected;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(Request.Params.Get("MatID") != null)
            {
                matID = Int32.Parse(
                    Request.Params.Get("MatID")
                );
                Material mat = new Material();
                matTitle = mat.GetTitle(matID);
            }
            else
            {
                panSelected.Visible = false;
            }
            if(!IsPostBack)
            {
                this.Title = "Authors";
                dlAuthLoad();
                lbSelectedLoad();
            }
        }
        //end page_load

        private void dlAuthLoad()
        {
            Author authors = new Author();
            DataSet AuthData = authors.GetAuthors();
            DataView AuthView = AuthData.Tables[0].DefaultView;
            dlAuth.DataSource = AuthView;
            dlAuth.DataBind();
        }
        private void lbSelectedLoad()
        {
            MaterialAuthor ma =
```

```

        new MaterialAuthor();
        DataSet maAuthors =
            ma.GetAuthorsPerMatID(matID);
        lbSelected.DataSource =
            maAuthors.Tables[0].DefaultView;
        lbSelected.DataTextField = "FullName";
        lbSelected.DataValueField = "AuthMat";
        lbSelected.DataBind();
    }

```

Web Form Designer generated code

```

private void btnAdd_Click(object sender, System.EventArgs e)
{
    string msg;
    Author author = new Author();
    int authID = author.Add(txtFirstName.Text, txtLastName.Text, txtBio.Text);
    if(authID > 0)
    {
        if(Request.Params.Get("MatID") != null)
        {
            MaterialAuthor ma = new MaterialAuthor();
            ma.Add( matID, authID, 0);
        }
        //reload the authors
        dlAuthLoad();
        lbSelectedLoad();
        msg = txtFirstName.Text + " " +
            txtLastName.Text +
            " added.";
    }
    else
    {
        msg = "Could not add " +
            txtFirstName.Text + " " +
            txtLastName.Text + ".";
    }
    lblMessage.Text = msg;
}

protected void dlAuth_CancelCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    dlAuth.EditItemIndex = -1;
    dlAuth.SelectedIndex = -1;
    dlAuthLoad();
}

protected void dlAuth_DeleteCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblEditID") ).Text;
    string firstName = ( (TextBox) e.Item.FindControl("txtEditFirst") ).Text;
    string lastName = ( (TextBox) e.Item.FindControl("txtEditLast") ).Text;
    string msg;
    Author author = new Author();
    dlAuth.EditItemIndex = -1;
    if( author.Delete( Int32.Parse(strID.Trim()) ) )
    {
        dlAuthLoad();
        lbSelectedLoad();
        msg = firstName + " " +
            lastName + " deleted.";
    }
    else
    {
        dlAuthLoad();
    }
}

```

```

        msg = "Could not delete " +
            firstName + " " +
            lastName + ".";
    }
    lblMessage.Text = msg;
}

protected void dlAuth_EditCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    dlAuth.EditItemIndex = (int)e.Item.ItemIndex;
    dlAuthLoad();
}

protected void dlAuth_UpdateCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblEditID") ).Text;
    string firstName = ( (TextBox) e.Item.FindControl("txtEditFirst") ).Text;
    string lastName = ( (TextBox) e.Item.FindControl("txtEditLast") ).Text;
    string bio = ( (TextBox) e.Item.FindControl("txtEditBio") ).Text;
    string msg;
    Author author = new Author();
    dlAuth.EditItemIndex = -1;
    if( author.Update( Int32.Parse(strID), firstName, lastName, bio) )
    {
        dlAuthLoad();
        lbSelectedLoad();
        msg = firstName + " " +
            lastName + " updated.";
    }
    else
    {
        dlAuthLoad();
        msg = "Could not update " +
            firstName + " " +
            lastName + ".";
    }
    lblMessage.Text = msg;
}

private void btnRemove_Click(object sender, System.EventArgs e)
{
    MaterialAuthor ma = new MaterialAuthor();
    if(ma.Delete(Int32.Parse(lbSelected.SelectedItem.Value)))
    {
        lblMessage.Text = lbSelected.SelectedItem.Text + " removed from " +
matTitle + ".";
        lbSelected.ClearSelection();
        lbSelectedLoad();
    }
}

private void dlAuth_ItemCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    if(e.CommandName == "select")
    {
        if(Request.Params.Get("MatID") != null)
        {
            string strID = ((Label)(e.Item.FindControl("lblAuthID"))).Text;
            string name = ((Label)(e.Item.FindControl("lblFirstName"))).Text + " "
+
            ((Label)(e.Item.FindControl("lblLastName"))).Text;
            int authID = Int32.Parse(strID);
            //add Author to AuthorMaterials table

```

```
        MaterialAuthor ma = new MaterialAuthor();
        ma.Add( matID, authID, 0);

        lblMessage.Text =
            "Assigned " + name + " as Author to " + matTitle + ".";
        lbSelectedLoad();
    }
}

private void btnSequence_Click(object sender, System.EventArgs e)
{
    if(lbSelected.Rows > 0)
    {
        //there are authors
        Response.Redirect("Sequence.aspx?MatID=" + matID, true);
    }
    else
    {
        //cannot set sequence if there
        //are no authors
        lblMessage.Text="To proceed, you must assign authors to this material.";
    }
}
} //end class
} //end namespace
```

```

<%@ Page language="c#" Codebehind="Authors.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Authors" %>
    <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="100%" border="0"
    >
        <TR>
            <TD>First Name</TD>
            <TD><asp:textbox id="txtFirstName" runat="server"></asp:textbox></
TD>
        </TR>
        <TR>
            <TD>Last Name</TD>
            <TD><asp:textbox id="txtLastName" runat="server"></asp:textbox></
TD>
        </TR>
        <TR>
            <TD>Bio</TD>
            <TD><asp:textbox id="txtBio" runat="server" TextMode="MultiLine"><
/asp:textbox></TD>
        </TR>
        <TR>
            <TD></TD>
            <TD><asp:button id="btnAdd" runat="server" Text="Add Author"></asp:
:button></TD>
        </TR>
    </TABLE>
    <P><asp:label id="lblMessage" runat="server"></asp:label></P>
</div>
<TABLE id="Table3" cellSpacing="1" cellPadding="1" width="100%" border="0">
    <TR>
        <TD><asp:datalist id="dlAuth" runat="server">
            <SelectedItemTemplate>
                <asp:Label id=lblSelAuthID runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthID") %>'>
                </asp:Label>&nbsp;:
                <asp:Label id=lblSelFirstName runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthFirstName") %>'>
                </asp:Label>&nbsp;:
                <asp:Label id=lblSelLastName runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthLastName") %>'>
                </asp:Label><BR>
                <asp:Label id=lblSelBio runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthBio") %>'>
                </asp:Label><BR>
                <asp:Label id=lblSelDate runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthLastUpdate") %>'>
                </asp:Label>&nbsp;:
                <asp:LinkButton id="lbnSelEdit" runat="server" CommandName=
="edit">Edit</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnSelCancel" runat="server"
CommandName="cancel">Cancel</asp:LinkButton>
            </SelectedItemTemplate>
            <ItemTemplate>
                <asp:Label id=lblAuthID runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthID") %>'>
                </asp:Label>&nbsp;:
                <asp:Label id=lblFirstName runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthFirstName") %>'>
                </asp:Label>&nbsp;:
                <asp:Label id=lblLastName runat="server" text='<%#
DataBinder.Eval(Container.DataItem, "AuthLastName") %>'>
                </asp:Label>&nbsp;|
                <asp:LinkButton id="lbnEdit" runat="server" CommandName="
edit">Edit</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnSelect" runat="server" CommandName=
"select">Select</asp:LinkButton>
            </ItemTemplate>
            <SeparatorTemplate>

```

```

        &nbsp;
    </SeparatorTemplate>
    <EditItemTemplate>
        <asp:Label id=lblEditID runat="server" text="<%#
DataBinder.Eval(Container.DataItem,"AuthID") %>'>
        </asp:Label><BR>
        <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="
300" border="1">

            <TR>
                <TD>First Name</TD>
                <TD>
                    <asp:TextBox id=txtEditFirst runat="server"
text="<%# DataBinder.Eval(Container.DataItem,"AuthFirstName") %>'>
                    </asp:TextBox></TD>
                </TR>
                <TR>
                <TD>Last Name</TD>
                <TD>
                    <asp:TextBox id=txtEditLast runat="server"
text="<%# DataBinder.Eval(Container.DataItem,"AuthLastName") %>'>
                    </asp:TextBox></TD>
                </TR>
                <TR>
                <TD>Bio</TD>
                <TD>
                    <asp:TextBox id=txtEditBio runat="server"
TextMode="MultiLine" text="<%# DataBinder.Eval(Container.DataItem,"AuthBio") %>'>
                    </asp:TextBox></TD>
                </TR>
            </TABLE>
            <P align="center">
                <asp:LinkButton id="lbnUpdate" runat="server"
CommandName="update">Update</asp:LinkButton>&nbsp;|&nbsp;&nbsp;
                <asp:LinkButton id="lbnDelete" runat="server"
CommandName="delete">Delete</asp:LinkButton>&nbsp;|&nbsp;&nbsp;
                <asp:LinkButton id="lbnCancel" runat="server"
CommandName="cancel">Cancel</asp:LinkButton></P>
            </EditItemTemplate>
        </asp:datalist></TD>
        <TD width="30"></TD>
        <TD valign="top">
            <asp:Panel id="panSelected" runat="server">
                <P><asp:listbox id="lbSelected" runat="server"></asp:listbox></P>
                <P><asp:button id="btnRemove" runat="server" Text="Remove"></asp:
button></P>
                <P><asp:button id="btnSequence" runat="server" Text="Set Sequence">
></asp:button></P>
            </asp:Panel>
        </TD>
    </TR>
</TABLE>

```



```

using System;
using System.Collections;
using System.Configuration;
using System.Web.UI;
using System.Diagnostics;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Drawing;
using Merazzle.WebModules.Core;
using Sharon.Controls.User;

namespace Sharon
{
    /// <summary>
    /// Summary description for BasePage.
    /// </summary>
    public abstract class BasePage: System.Web.UI.Page
    {
        private HtmlGenericControl title, link, script, body;
        private string returnUrl, javascriptSrc, header, side, footer, root, cssHref;
        protected string Header
        {
            get{ return Root + header;}
            set{ header = value;}
        }
        protected string Root
        {
            get{ return root; }
        }
        protected string Side
        {
            get{ return Root + side;}
            set{ side = value;}
        }
        protected string Footer
        {
            get{ return Root + footer;}
            set{ footer = value;}
        }
        protected string PageTitle
        {
            get { return title.InnerText; }
            set { title.InnerText = value; }
        }
        protected string ReturnUrl
        {
            get
            {
                if (returnUrl == null )
                    returnUrl = Root + @"/Accounts/Login.aspx?ReturnUrl=" + this.Context.
Request.RawUrl;
                return returnUrl;
            }
            set
            {
                returnUrl = Root + @"/Accounts/Login.aspx?ReturnUrl=" + value;
            }
        }
        protected string CssHref
        {
            get{ return Root + cssHref; }
            set
            {
                cssHref = value;
                link.Attributes["href"] = Root + cssHref;
            }
        }
    }
}

```

```

protected string JavaScriptSrc
{
    get{ return javaScriptSrc; }
    set
    {
        javaScriptSrc = Root + value;
        script.Attributes["src"] = javaScriptSrc;
    }
}
protected string BodyBackgroundImage
{
    set
    {
        string background = value;
        body.Style.Add("background-image", background);
    }
}
protected void SetBodyStyle(string style, string styleValue)
{
    IEnumerator keys = body.Style.Keys.GetEnumerator();
    while (keys.MoveNext())
    {
        String key = (String)keys.Current;
        if( style.Equals(key) )
        {
            body.Style.Remove(key);
            break;
        }
    }
    body.Style.Add(style, styleValue);
}
protected void SetBodyAttribute(string attr, string attrValue)
{
    IEnumerator keys = body.Attributes.Keys.GetEnumerator();
    while (keys.MoveNext())
    {
        String key = (String)keys.Current;
        if( attr.Equals(key) )
        {
            body.Attributes.Remove(key);
            break;
        }
    }
    body.Attributes.Add(attr, attrValue);
}
protected virtual void BuildPage( HtmlGenericControl body )
{
    // Build the page and include the generated form
    this.Controls.AddAt( 0, new LiteralControl( @"
<html><head>" ) );

    this.Controls.Add(title);

    link.ID = "CssLink";
    link.Attributes["type"] = "text/css";
    link.Attributes["rel"] = "stylesheet";
    link.Attributes["href"] = CssHref;

    script.Attributes["type"] = "text/javascript";
    script.Attributes["src"] = JavaScriptSrc;

    this.Controls.Add(link);
    this.Controls.Add(script);

    this.Controls.Add( new LiteralControl( @"</head>" ) );
}

```

```

        this.Controls.Add( body );
        this.Controls.Add( new LiteralControl( @"</html>" ) );
    }
    protected HtmlGenericControl BuildBody( HtmlForm form)
    {
        body.ID = "BaseBody";
        body.Controls.Add(form);
        return body;
    }
    protected abstract HtmlForm BuildForm();
    protected void AddControlsFromDerivedPage(Control control)
    {
        Panel panel = new Panel();
        panel.ID = "ContentPanel";
        control.Controls.Add( panel );
        int count = this.Controls.Count;
        for( int i = 0; i<count; ++i )
        {
            System.Web.UI.Control ctrl = this.Controls[i];
            panel.Controls.Add( ctrl );
            this.Controls.Remove( ctrl );
        }
    }

    public BasePage():base()
    {
        title = new HtmlGenericControl("title");
        script = new HtmlGenericControl("script");
        link = new HtmlGenericControl("link");
        body = new HtmlGenericControl("body");
        root = ConfigurationSettings.AppSettings["Root"].ToString();
    }

    protected override void OnInit(EventArgs e)
    {
        BuildPage( BuildBody( BuildForm() ) );
        base.OnInit(e);
        this.Load += new System.EventHandler(this.BasePage_Load);
    }

    private void BasePage_Load(object sender, System.EventArgs e)
    {
        // TODO: Place any code that will take place BEFORE the Page_Load event
        // in the regular page, e.g. cache management, authentication verification,
etc.
    }
}

```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Books.
    /// </summary>
    public class Books : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.Label lblMessage;
        protected System.Web.UI.WebControls.Repeater rp;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                Book book = new Book();
                DataSet ds = book.GetBooks();
                rp.DataSource =
                    ds.Tables[0].DefaultView;
                rp.DataBind();
                Title = "Books";

                int count = ds.Tables[0].Rows.Count;
                if(count ==0)
                    lblMessage.Text = "No records found.";
                else if (count == 1)
                    lblMessage.Text = "One record found.";
                else
                    lblMessage.Text = count + " records found.";
            }
        }

        Web Form Designer generated code
    }
}
```

```

<%@ Page language="c#" Codebehind="Books.aspx.cs" AutoEventWireup="false" Inherits="Sharon
.Books.Books" %>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <p></p>
    <asp:Repeater id="rp" runat="server">
        <HeaderTemplate>
            <table>
                <tr>
                    <th colspan="2">
                        All Books</th></tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr>
                    <td valign="top">
                        <%# DataBinder.Eval(Container.DataItem, "MatID") %>
                    </td>
                    <td>
                        <a href='<%# "Details.aspx?MatID=" + DataBinder.Eval(
Container.DataItem, "MatID") %>'>
                            <%# DataBinder.Eval(Container.DataItem, "MatTitle") %>
                        </a>
                        <br />
                        <%# DataBinder.Eval(Container.DataItem, "MatDatePub") %>
                    </td>
                </tr>
            </ItemTemplate>
            <FooterTemplate>
            </table>
            </FooterTemplate>
        </asp:Repeater>

```

```
namespace Sharon.Books
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    ///     Summary description for BooksNav.
    /// </summary>
    public class BooksNav : System.Web.UI.UserControl
    {
        private void Page_Load(object sender, System.EventArgs e)
        {
            //
            //      Xml xml = new Xml();
            //      xml.TransformSource = "Books.xslt";
            //      xml.DocumentSource = "Books.xml";
            //      NavPanel.Controls.Add(xml);
            //
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        ///     Required method for Designer support - do not modify
        ///     the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.Load += new System.EventHandler(this.Page_Load);
        }
    }
}

#endregion
```

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="BooksNav.ascx.cs" Inherits="
Sharon.Books.BooksNav" TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<!-- BooksNav User Control-->
<div id="blue">
  <table cellpadding="5" cellspacing="0" width="100%">
    <TR>
      <TD height="20"><a href="/Sharon/Books/Add.aspx">Add</a></TD>
    </TR>
    <tr>
      <td height="20"><a href="/Sharon/Books/Articles.aspx">Articles</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Authors.aspx">Authors</a></td>
    </tr>
    <TR>
      <TD height="20"><a href="/Sharon/Books/Books.aspx">Books</a></TD>
    </TR>
    <tr>
      <td height="20"><a href="/Sharon/Books/Categories.aspx">Categories</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Cities.aspx">Cities</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Journals.aspx">Journals</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Materials.aspx">Materials</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Publishers.aspx">Publishers</a></td>
    </tr>
    <tr>
      <td height="20"><a href="/Sharon/Books/Default.aspx">SEARCH</a></td>
    </tr>
    <TR>
      <TD height="20"><a href="/Sharon/Books/Urls.aspx">Urls</a></TD>
    </TR>
  </table>
</div>
<!-- BooksNav User Control End -->
```

```

using System;
using System.Collections;
using System.Configuration;
using System.Web.UI;
using System.Diagnostics;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Drawing;
using Merazzle.WebModules.Core;
using Sharon.Controls.User;
using Sharon.Books;
namespace Sharon
{
    /// <summary>
    /// Summary description for BooksPage.
    /// </summary>
    public class BooksPage:Sharon.BasePage
    {
        private Sharon.Controls.User.Head head;
        private HtmlForm form;
        protected void SetFormAttribute(string attr, string attrValue)
        {
            IEnumerator keys = form.Attributes.Keys.GetEnumerator();
            while (keys.MoveNext())
            {
                String key = (String)keys.Current;
                if( attr.Equals(key) )
                {
                    form.Attributes.Remove(key);
                    break;
                }
            }
            form.Attributes.Add(attr, attrValue);
        }
        public BooksPage():base()
        {
            Header = ConfigurationSettings.AppSettings["MainHeader"];
            Side = ConfigurationSettings.AppSettings["BooksSide"].ToString();
            Footer = ConfigurationSettings.AppSettings["MainFooter"].ToString();
            CssHref = ConfigurationSettings.AppSettings["DefaultStyleSheet"].ToString();
            //this.SetBodyAttribute("onload","initMenus()");
            //this.JavaScriptSrc = ConfigurationSettings.AppSettings["JavaScript"].
            ToString();
            SetBodyAttribute("topmargin","0px");
            SetBodyAttribute("marginheight","0px");
        }

        public string Title
        {
            get{ return PageTitle; }
            set{
                head.Heading = value;
                PageTitle = value;
            }
        }

        protected override HtmlForm BuildForm()
        {
            form = new HtmlForm();
            form.Name = "f";
            head = (Head)LoadControl(Header);
            form.Controls.Add(head);

            Table table = new Table();
            table.ID = "MainTable";
            table.GridLines = GridLines.Both;
            table.BorderWidth = Unit.Pixel(0);

```



```
        table.CellSpacing = 0;
        table.CellPadding = 1;
        table.Height = Unit.Pixel(500);
        table.Width = Unit.Percentage(95.0);

        TableRow row = new TableRow();
        TableCell cell = new TableCell();
        cell.VerticalAlign = VerticalAlign.Top;
        cell.Width = Unit.Percentage(20.0);
        cell.ID = "LeftColumn";
        cell.CssClass = "firstblue";
        cell.Controls.Add(LoadControl(Side));
        row.Cells.Add(cell);

        cell = new TableCell();
        cell.CssClass = "line";
        cell.Width = Unit.Percentage(5.0);
        cell.Text = "&nbsp;";
        row.Cells.Add(cell);

        cell = new TableCell();
        cell.VerticalAlign = VerticalAlign.Top;
        cell.Width = Unit.Percentage(75.0);
        AddControlsFromDerivedPage(cell);
        cell.ID = "RightColumn";
        row.Cells.Add(cell);
        table.Rows.Add(row);
        form.Controls.Add(table);
        form.Controls.Add(LoadControl(Footer));
        return form;
    }
}
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Categories.
    /// </summary>
    public class Categories : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.TextBox txtAddCat;
        protected System.Web.UI.WebControls.TextBox txtAddDesc;
        protected System.Web.UI.WebControls.DataList dlCat;
        protected System.Web.UI.WebControls.Label lblMessage;
        protected System.Web.UI.WebControls.Button btnAddCat;
        protected Panel panMaster, panSlave;

        protected Repeater reMaterials;

        private DataView CatView;
        private DataSet CatData;
        private int catID;
        private string catTitle;

        private void Page_Load(object sender, System.EventArgs e)
        {
            string strID = Request.QueryString.Get("CatID");
            if(!IsPostBack)
            {
                this.Title = "Categories";
                if (strID == null)
                {
                    panMaster.Visible = true;
                    panSlave.Visible = false;
                    LoadCategories();
                }
                else
                {
                    panMaster.Visible = false;
                    panSlave.Visible = true;
                    catID = Int32.Parse(strID.Trim());
                    Category cat = new Category();
                    catTitle = cat.GetCategory(catID);
                    MaterialCategory matCat =
                        new MaterialCategory();
                    reMaterials.DataSource =
                        matCat.GetMaterialsPerCatID(catID);
                    reMaterials.DataBind();
                    this.Title += ": " + catTitle;
                }
            }
        }

        private void LoadCategories()
        {
            Category category = new Category();
            CatData = category.GetCategories();
            CatView = CatData.Tables["Categories"].DefaultView;
        }
    }
}
```

```

        dlCat.DataSource = CatView;
        dlCat.DataBind();
    }

```

Web Form Designer generated code

```

protected void btnAddCat_Click(object sender, System.EventArgs e)
{
    string msg;
    Category category = new Category();
    if(category.Add(txtAddCat.Text, txtAddDesc.Text))
    {
        //reload the categories
        LoadCategories();
        msg = txtAddCat.Text + " added.";
    }
    else
    {
        msg = "Could not add " + txtAddCat.Text + ".";
    }
    lblMessage.Text = msg;
}

protected void dlCat_CancelCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    dlCat.EditItemIndex = -1;
    LoadCategories();
}

protected void dlCat_DeleteCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblEditCatID") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtEditName") ).Text;
    string msg;
    Category category = new Category();
    dlCat.EditItemIndex = -1;
    if( category.Delete( Int32.Parse(strID.Trim()) ) )
        msg = name + " deleted.";
    else
        msg = "Could not delete " + name + ".";

    LoadCategories();
    lblMessage.Text = msg;
}

protected void dlCat_EditCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    dlCat.EditItemIndex = (int)e.Item.ItemIndex;
    LoadCategories();
}

protected void dlCat_UpdateCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblEditCatID") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtEditName") ).Text;
    string description = ( (TextBox) e.Item.FindControl("txtEditDesc") ).Text;
    string msg;
    Category category = new Category();
    dlCat.EditItemIndex = -1;
    if( category.Update( Int32.Parse(strID.Trim()), name, description ) )
        msg = name + " updated.";

    else

```

```
        msg = "Could not update " + name + ".";

        LoadCategories();
        lblMessage.Text = msg;
    }
}
```

```

<%@ Page language="c#" Codebehind="Categories.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Categories" %>

<asp:Panel id="panMaster" runat="server">
    <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300" border="1">
        <TR>
            <TD>Category:
            </TD>
            <TD>
                <asp:TextBox id="txtAddCat" runat="server"></asp:TextBox></TD>
        </TR>
        <TR>
            <TD valign="top">Description:
            </TD>
            <TD>
                <asp:TextBox id="txtAddDesc" runat="server" TextMode="
MultiLine"></asp:TextBox></TD>
        </TR>
        <TR>
            <TD></TD>
            <TD>
                <asp:Button id="btnAddCat" runat="server" Text="Add Category">
</asp:Button></TD>
        </TR>
    </TABLE>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
        <asp:DataList id="dlCat" runat="server" OnCancelCommand="
dlCat_CancelCommand" OnDeleteCommand="dlCat_DeleteCommand"
OnEditCommand="dlCat_EditCommand" OnUpdateCommand="dlCat_UpdateCommand
">
            <ItemTemplate>
                <asp:Label id="lblName" runat="server" Text='<%# DataBinder.Eval(
Container.DataItem, "CatName" ) %>'>
                </asp:Label>
                <BR>
                <asp:Label id="lblDesc" runat="server" Text='<%# DataBinder.Eval(
Container.DataItem, "CatDesc" ) %>'>
                </asp:Label><BR>
                <asp:Label id="lblDate" runat="server" Text='<%# DataBinder.Eval(
Container.DataItem, "CatLastUpdate" ) %>'>
                </asp:Label>&nbsp;<asp:LinkButton id="lbnEdit" runat="server" CommandName="Edit">
Edit</asp:LinkButton>
                | <a href='<%# "Categories.aspx?CatID=" + DataBinder.Eval(
Container.DataItem, "CatID" ) %>'>Select</a>
            </ItemTemplate>
            <SeparatorTemplate>
                &nbsp;</SeparatorTemplate>
            <EditItemTemplate>
                <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="300"
border="1">
                    <TR>
                        <TD>
                            <asp:Label id="lblEditCatID" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "CatID" ) %>'>
                            </asp:Label></TD>
                        <TD></TD>
                    </TR>
                    <TR>
                        <TD>Category:</TD>
                        <TD>
                            <asp:TextBox id="txtEditName" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "CatName" ) %>'>
                            </asp:TextBox></TD>
                    </TR>
                </TABLE>
            </EditItemTemplate>
        </DataList>
    </P>
</asp:Panel>

```

```

        <TR>
            <TD>Description:</TD>
            <TD>
                <asp:TextBox id="txtEditDesc" runat="server" TextMode=
"MultiLine" Text='<%# DataBinder.Eval(Container.DataItem,"CatDesc") %>'>
                </asp:TextBox></TD>
        </TR>
    </TABLE>
    <P align="center">
        <asp:LinkButton id="lbnUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|&nbsp;
        <asp:LinkButton id="lbnDelete" runat="server" CommandName="
delete">Delete</asp:LinkButton>&nbsp;|
        <asp:LinkButton id="lbnCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton>&nbsp;</P>
    </EditItemTemplate>
</asp:DataList>
</asp:Panel>
<asp:Panel id="panSlave" runat="server" visible="false">

<asp:Repeater id="reMaterials" runat="server">
<HeaderTemplate>
<a href="Categories.aspx">Back to Categories</a>
</HeaderTemplate>
<ItemTemplate>
<p><a href='<%# "Materials.aspx?MatID=" + DataBinder.Eval(Container.DataItem,"MatID") %>'>
    <%# DataBinder.Eval(Container.DataItem,"MatTitle") %></a></p>
</ItemTemplate>
</asp:Repeater>
</asp:Panel>

```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Cities.
    /// </summary>
    public class Cities : Sharon.BooksPage
    {
        protected Label lblMessage;
        protected DataList dlCities;
        protected TextBox txtAddCity;
        protected Button btnAdd;

        private DataView CityView;
        private DataSet CityData;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                LoadCities();
                this.Title = "Cities";
            }
        }

        Web Form Designer generated code

        protected void btnAdd_Click(object sender, System.EventArgs e)
        {
            string msg;
            City city = new City();
            if(city.Add(txtAddCity.Text)> 0)
            {
                //reload the cities
                LoadCities();
                msg = txtAddCity.Text + " added.";
            }
            else
            {
                msg = "Could not add " + txtAddCity.Text + ".";
            }
            lblMessage.Text = msg;
        }

        private void LoadCities()
        {
            City city = new City();
            CityData = city.GetCities();
            BindList();
        }

        private void BindList()
        {
            CityView = CityData.Tables["Cities"].DefaultView;
            dlCities.DataSource = CityView;
            dlCities.DataBind();
        }
    }
}
```

```
}

protected void dlCities_CancelCommand(object source, DataListCommandEventArgs e)
{
    dlCities.EditItemIndex = -1;
    BindList();
}

protected void dlCities_DeleteCommand(object source, DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblCityIDEdit") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtCityEdit") ).Text;
    string msg;
    City city = new City();
    dlCities.EditItemIndex = -1;
    if( city.Delete( Int32.Parse(strID.Trim()) ) )
    {
        LoadCities();
        msg = name + " deleted.";
    }
    else
    {
        BindList();
        msg = "Could not delete " + name + ".";
    }
    lblMessage.Text = msg;
}

protected void dlCities_EditCommand(object source, DataListCommandEventArgs e)
{
    dlCities.EditItemIndex = (int)e.Item.ItemIndex;
    LoadCities();
}

protected void dlCities_UpdateCommand(object source, DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblCityIDEdit") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtCityEdit") ).Text;
    string msg;
    City city = new City();
    dlCities.EditItemIndex = -1;
    if( city.Update( Int32.Parse(strID), name ) )
    {
        //reload the cities
        LoadCities();
        msg = name + " updated.";
    }
    else
    {
        msg = "Could not update " + name + ".";
    }
    lblMessage.Text = msg;
}
}
```



```

<%@ Page language="c#" Codebehind="Cities.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Cities" %>
<P>Enter City:
    <asp:textbox id="txtAddCity" runat="server"></asp:textbox><asp:button id="
btnAdd" runat="server" Text="Add City"></asp:button>&nbsp;</P>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
        <asp:DataList id="dlCities" runat="server" OnCancelCommand="
dlCities_CancelCommand" OnDeleteCommand="dlCities_DeleteCommand"
OnEditCommand="dlCities_EditCommand" OnUpdateCommand="
dlCities_UpdateCommand">
            <ItemTemplate>
                <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300"
border="1">
                    <TR>
                        <TD vAlign="top">
                            <P>
                                <asp:Label id=lblCityIDItem runat="server" Text='<
%# DataBinder.Eval(Container.DataItem,"CityID") %>'>
                                </asp:Label><BR>
                                <asp:LinkButton id="lbnEdit" runat="server"
CommandName="edit">Edit</asp:LinkButton></P>
                            </TD>
                            <TD>
                                <asp:Label id=lblCityItem runat="server" Text='<
DataBinder.Eval(Container.DataItem,"City") %>'>
                                </asp:Label><BR>
                                <asp:Label id=lblCityDateItem runat="server" Text='<
DataBinder.Eval(Container.DataItem,"CityLastUpdate") %>'>
                                </asp:Label></TD>
                            </TR>
                        </TABLE>
                    </ItemTemplate>
                    <EditItemTemplate>
                        <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="300"
border="1">
                            <TR>
                                <TD vAlign="top">
                                    <asp:Label id="lblCityIDEdit" runat="server" Text='<
DataBinder.Eval(Container.DataItem,"CityID") %>'>
                                    </asp:Label></TD>
                                <TD>
                                    <asp:TextBox id="txtCityEdit" runat="server" Text='<
DataBinder.Eval(Container.DataItem,"City") %>'>
                                    </asp:TextBox></TD>
                                </TR>
                            </TABLE>
                            <P align="center">
                                <asp:LinkButton id="lbnUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;<
                                <asp:LinkButton id="lbnDelete" runat="server" CommandName="
delete">Delete</asp:LinkButton>&nbsp;<
                                <asp:LinkButton id="lbnCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></P>
                            </EditItemTemplate>
                        </asp:DataList>
                    </div>

```

```

using System;
using System.Collections;
using System.Configuration;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Text;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for _Default.
    /// </summary>
    public class _Default : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.TextBox txtMatKey;
        protected System.Web.UI.WebControls.TextBox txtAuthKey;
        protected System.Web.UI.WebControls.Button btnSearch;
        protected System.Web.UI.WebControls.Label lblMessage;
        protected System.Web.UI.WebControls.Repeater rpResults;
        protected System.Web.UI.WebControls.Label lblQuery;
        protected System.Web.UI.WebControls.DropDownList ddlCatKey;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                // load Categories
                Category cat = new Category();
                ddlCatKey.DataSource =
                    cat.GetCategories().
                        Tables[0].DefaultView;
                ddlCatKey.DataTextField = "CatName";
                ddlCatKey.DataValueField = "CatID";
                ddlCatKey.DataBind();
                ListItem item = new ListItem("All Categories", "0");
                ddlCatKey.Items.Insert(0,item);
            }
        }

        Web Form Designer generated code

        private void btnSearch_Click(object sender, System.EventArgs e)
        {
            string matKey = txtMatKey.Text.Trim();
            string authKey = txtAuthKey.Text.Trim();
            int catID =
                Int32.Parse(
                    ddlCatKey.SelectedItem.Value);

            //Search all Materials
            //Search all Materials with
            //title or desc like keyword

            StringBuilder sb = new StringBuilder();
            sb.Append("SELECT Distinct m.MatID, m.MatTitle ");
            sb.Append("FROM BK_Material m, ");
            sb.Append("BK_MaterialCategory mc, ");
            sb.Append("BK_Category c, ");

```

```

sb.Append("BK_AuthorMaterial am, ");
sb.Append("BK_Author a ");
sb.Append("WHERE");
sb.Append("( m.MatID = mc.MatID ");
sb.Append("AND mc.CatID = c.CatID) ");
sb.Append("OR (am.MatID = m.MatID ");
sb.Append("AND a.AuthID = am.AuthID) ");

if(matKey.Length > 0)
{
    sb.Append("AND (m.MatTitle LIKE '%");
    sb.Append(matKey);
    sb.Append("%' OR m.MatDesc LIKE '%");
    sb.Append(matKey);
    sb.Append("%' ) ");
}
if(authKey.Length > 0)
{
    sb.Append("AND (a.AuthFirstName LIKE '%");
    sb.Append(authKey);
    sb.Append("%' OR a.AuthLastName LIKE '%");
    sb.Append(authKey);
    sb.Append("%' OR a.AuthBio LIKE '%");
    sb.Append(authKey);
    sb.Append("%' ) ");
}
if(catID > 0)
{
    sb.Append("AND c.CatID=" + catID);
}

string query = sb.ToString();
string strConn =
    ConfigurationSettings.
    AppSettings["ConnString"].
    ToString();

SqlConnection conn =
    new SqlConnection(strConn);
SqlCommand comm =
    new SqlCommand(query, conn);

SqlDataAdapter da =
    new SqlDataAdapter(comm);
DataSet ds = new DataSet();
da.Fill(ds);
rpResults.DataSource =
    ds.Tables[0].DefaultView;
rpResults.DataBind();

int count = ds.Tables[0].Rows.Count;
if(count == 0)
    lblMessage.Text = "No records found.";
else if (count == 1)
    lblMessage.Text = "One record found.";
else
    lblMessage.Text = count + " records found.";

lblQuery.Text = query;
} //end btnSearch_Click
} //end class
} //end namespace

```

```

<%@ Page language="c#" Codebehind="Default.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books._Default" %>
    <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="100%" border="1">
        <TR>
            <TD>
                <P>Keyword in Material
                </P>
            </TD>
            <TD>
                <asp:TextBox id="txtMatKey" runat="server"></asp:TextBox></TD>
            <TD>(title, description)</TD>
        </TR>
        <TR>
            <TD>Author Name</TD>
            <TD>
                <asp:TextBox id="txtAuthKey" runat="server"></asp:TextBox></TD>
            <TD>(first, last, bio)</TD>
        </TR>
        <TR>
            <TD>Category</TD>
            <TD>
                <asp:DropDownList id="ddlCatKey" runat="server"></asp:
DropDownList></TD>
            <TD></TD>
        </TR>
        <TR>
            <TD></TD>
            <TD>
                <asp:Button id="btnSearch" runat="server" Text="Search"></asp:
Button></TD>
            <TD></TD>
        </TR>
    </TABLE>
    <P>
        <asp:Label id="lblQuery" runat="server"></asp:Label></P>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <asp:Repeater id="rpResults" runat="server">
        <ItemTemplate>
            <li>
                <a href='<%# "Details.aspx?MatID=" + DataBinder.Eval(Container.
DataItem,"MatID") %>'>
                    <%# DataBinder.Eval(Container.DataItem,"MatTitle") %>
                </a>
            </li>
        </ItemTemplate>
        <HeaderTemplate>
            <ol>
        </HeaderTemplate>
        <FooterTemplate>
            </ol>
        </FooterTemplate>
    </asp:Repeater>

```

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Details.
    /// </summary>
    public class Details : Sharon.BooksPage
    {
        protected DataList dlMat;
        protected DataList dlArt;
        protected DataList dlBook;
        protected DataList dlUrl;
        protected Label lblAuthors;
        protected HyperLink linkAuthors;
        protected LinkButton lbnCatUpdate;
        protected CheckBoxList cbCats;
        protected Panel panArt, panBook, panUrl;
        private int matID;

        private void Page_Load(object sender, System.EventArgs e)
        {
            string strMatID = (Request.QueryString.Get("MatID"));

            if( (strMatID != null) )
            {
                matID = Int32.Parse(strMatID);
                //load details
                if(!IsPostBack)
                {
                    LoadDetails();
                    Title = "Details Page";
                }
            }
            else
            {
                //cannot load details, so redirect
                Server.Transfer("Default.aspx",true);
            }
        } //end page_load

        private void LoadDetails()
        {
            panArt.Visible = false;
            panBook.Visible = false;
            panUrl.Visible = false;
            dlArt.Visible = false;
            dlBook.Visible = false;
            dlUrl.Visible = false;
            Material mat = new Material();
            DataSet ds = mat.GetMaterialByID(matID);
            if(ds.Tables.Contains("Material"))
            {
                DataRow row =
                    ds.Tables["Material"].Rows[0];
                DataView dv =
                    ds.Tables["Material"].DefaultView;
            }
        }
    }
}

```

```

        dlMat.DataSource = dv;
        dlMat.DataBind();

        if(row["JournalID"].ToString() != "")
        {
            //Material is an Article
            panArt.Visible = true;
            dlArt.Visible = true;
            dlArt.DataSource = dv;
            dlArt.DataBind();
        }
        if(row["ISBN"].ToString() != "")
        {
            //Material is a Book
            panBook.Visible = true;
            dlBook.Visible = true;
            dlBook.DataSource = dv;
            dlBook.DataBind();
        }
        if(row["Url"].ToString() != "")
        {
            //Material is a Url
            panUrl.Visible = true;
            dlUrl.Visible = true;
            dlUrl.DataSource = dv;
            dlUrl.DataBind();
        }
    }
    if(ds.Tables.Contains("Authors"))
    {
        IEnumerator en =
            ds.Tables["Authors"].
            Rows.GetEnumerator();
        lblAuthors.Text = "";
        while(en.MoveNext())
        {
            DataRow row = (DataRow)en.Current;
            lblAuthors.Text +=
                row["FullName"].ToString()
                + "<br />";
        }
    }
    if(ds.Tables.Contains("Categories"))
    {
        cbCats.Items.Clear();
        DataRowCollection rows =
            ds.Tables["Categories"].Rows;
        ListItem item;
        foreach(DataRow row in rows)
        {
            item = new ListItem();
            item.Text =
                row["CatName"].ToString();
            item.Value =
                row["CatID"].ToString();
            item.Selected =
                ((int)row["Selected"]) > 0;
            cbCats.Items.Add(item);
        }
    }
    linkAuthors.NavigateUrl =
        "Authors.aspx?MatID=" + matID;
} //end loadDetails

```

Web Form Designer generated code

```
private void dlMat_EditCommand(object source, DataListCommandEventArgs e)
{
    dlMat.EditItemIndex =
        (int)e.Item.ItemIndex;
    LoadDetails();
}

private void dlMat_CancelCommand(object source, DataListCommandEventArgs e)
{
    dlMat.EditItemIndex = -1;
    LoadDetails();
}

private void dlMat_DeleteCommand(object source, DataListCommandEventArgs e)
{
    Material mat = new Material();
    if(mat.Delete(matID))
        Response.Redirect("Default.aspx", true);
}

private void dlMat_UpdateCommand(object source, DataListCommandEventArgs e)
{
    string title, description, strDate;
    DateTime date;
    title = ((TextBox)e.Item.FindControl("txtTitle")).Text.Trim();
    strDate = ((TextBox)e.Item.FindControl("txtDate")).Text.Trim();
    description = ((TextBox)e.Item.FindControl("txtDesc")).Text.Trim();
    date = Convert.ToDateTime(strDate);
    Material mat = new Material();
    mat.Update(matID, title, description, date);
    dlMat.EditItemIndex = -1;
    LoadDetails();
}

private void dlArt_EditCommand(object source, DataListCommandEventArgs e)
{
    dlArt.EditItemIndex = e.Item.ItemIndex;
    //Find dropdownlist in datalist
    DropDownList ddlJ = (DropDownList)e.Item.FindControl("ddlJournal");

    //bind journal data to ddl
    Journal j = new Journal();
    ddlJ.DataSource =
        j.GetJournals().Tables[0].DefaultView;
    ddlJ.DataTextField = "JournalName";
    ddlJ.DataValueField = "JournalID";
    ddlJ.DataBind();

    LoadDetails();
}

private void dlArt_CancelCommand(object source, DataListCommandEventArgs e)
{
    dlArt.EditItemIndex = -1;
    LoadDetails();
}

private void dlArt_UpdateCommand(object source, DataListCommandEventArgs e)
{
    string journalName = ((TextBox)e.Item.
        FindControl("txtJournal")).Text.Trim();
```

```

        DropDownList ddlJ = ((DropDownList)e.Item.
            FindControl("ddlJournal"));
        int journalID, vol, num, start, end;
        journalID = -1;
        if(journalName.Length > 0)
        {
            //user typed value
            Journal j = new Journal();
            journalID = j.Add(journalName);
        }
        else if(ddlJ.SelectedIndex >= 0)
        {
            journalID = Int32.Parse(
                ddlJ.SelectedItem.Value);
        }
        vol = Int32.Parse(((TextBox)e.Item.
            FindControl("txtVol")).Text.Trim());
        num = Int32.Parse(((TextBox)e.Item.
            FindControl("txtNum")).Text.Trim());
        start = Int32.Parse(((TextBox)e.Item.
            FindControl("txtStart")).Text.Trim());
        end = Int32.Parse(((TextBox)e.Item.
            FindControl("txtEnd")).Text.Trim());
        Article art = new Article();
        art.Update(matID, journalID, vol,
            num, start, end);
        dlArt.EditItemIndex = -1;
        LoadDetails();
    }

    private void dlBook_CancelCommand(object source, DataListCommandEventArgs e)
    {
        dlBook.EditItemIndex = -1;
        LoadDetails();
    }

    private void dlBook_EditCommand(object source, DataListCommandEventArgs e)
    {
        dlBook.EditItemIndex = e.Item.ItemIndex;
        LoadDetails();

        //Find dropdownlist in datalist
        DropDownList ddlP =
            (DropDownList)e.Item.
                FindControl("ddlPub");

        ddlP.Items.Add("item");

        //bind publisher data to ddl
        Publisher p = new Publisher();
        //
        ddlP.DataSource =
        //
            p.GetPublishers().Tables[0].DefaultView;
        //
        ddlP.DataTextField = "PubName";
        //
        ddlP.DataValueField = "PubID";
        //
        ddlP.DataBind();

        //Find dropdownlist in datalist
        DropDownList ddlC = (DropDownList)e.Item.FindControl("ddlCity");

        //bind city data to ddl
        City c = new City();
        //
        ddlC.DataSource =
        //
            c.GetCities().Tables[0].DefaultView;
        //
        ddlC.DataTextField = "City";
        //
        ddlC.DataValueField = "CityID";
        //
        ddlC.DataBind();
    }

```



```
}

private void dlBook_UpdateCommand(object source, DataListCommandEventArgs e)
{
    int pubID = 0, cityID = 0;
    string isbn = ((TextBox)e.Item.
        FindControl("txtISBN")).Text.Trim();

    TextBox tc = (TextBox)e.Item.
        FindControl("txtCity");
    TextBox tp = (TextBox)e.Item.
        FindControl("txtPub");

    DropDownList ddlC = (DropDownList)e.Item.
        FindControl("ddlCity");
    DropDownList ddlP = (DropDownList)e.Item.
        FindControl("ddlPub");

    if(tc.Text.Length > 0)
    {
        //user typed in City
        City city = new City();
        cityID = city.Add(tc.Text.Trim());
    }
    else if(ddlC.SelectedIndex >= -1)
    {
        cityID = Int32.Parse(
            ddlC.SelectedItem.Value);
    }
    if(tp.Text.Length > 0)
    {
        //user typed in Publisher
        Publisher pub = new Publisher();
        pubID = pub.Add(tp.Text.Trim());
    }
    else if(ddlP.SelectedIndex >= -1)
    {
        pubID = Int32.Parse(
            ddlP.SelectedItem.Value);
    }
    Book book = new Book();
    book.Update(matID, pubID, cityID, isbn);

    dlBook.EditItemIndex = -1;
    LoadDetails();
}

private void dlUrl_EditCommand(object source, DataListCommandEventArgs e)
{
    dlUrl.EditItemIndex = e.Item.ItemIndex;
    LoadDetails();
}

private void dlUrl_UpdateCommand(object source, DataListCommandEventArgs e)
{
    string address =
        ((TextBox)e.Item.
            FindControl("txtAddress")).
            Text.Trim();
    Url url = new Url();
    url.Add(matID, address);
    dlUrl.EditItemIndex = -1;
    LoadDetails();
}

private void dlUrl_CancelCommand(object source, DataListCommandEventArgs e)
```

```
{
    dlUrl.EditItemIndex = -1;
    LoadDetails();
}

private void lbnCatUpdate_Click(object sender, EventArgs e)
{
    MaterialCategory mc = new MaterialCategory();
    IEnumerator iter = cbCats.Items.GetEnumerator();
    while(iter.MoveNext())
    {
        ListItem item = (ListItem)iter.Current;
        if(item.Selected)
            mc.Add(matID, Int32.Parse(item.Value));
    }
}
}
```

```

<%@ Page language="c#" Codebehind="Details.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Details" %>
<asp:DataList id="dlMat" runat="server" Width="100%">
    <ItemTemplate>
        <P>
            <asp:Label id=lblTitle runat="server" Text='<%# DataBinder.Eval(
Container.DataItem,"MatTitle") %>'>No Title</asp:Label>,
            <asp:Label id=lblDate runat="server" Text='<%# DataBinder.Eval(
Container.DataItem,"MatDatePub") %>'>No Date</asp:Label></P>
            <asp:Panel id="panDesc" runat="server">
                <%# DataBinder.Eval(Container.DataItem,"MatDesc") %>
            </asp:Panel>
            <p>
                <asp:LinkButton id="btnEdit" runat="server" CommandName="edit">
Edit</asp:LinkButton></P>
        </ItemTemplate>
        <EditItemTemplate>
            <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="100%" border
="0">
                <TR>
                    <TD>Title</TD>
                    <TD>
                        <asp:TextBox id=txtTitle runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"MatTitle") %>'>
                        </asp:TextBox></TD>
                </TR>
                <TR>
                    <TD>Publication Date
                    </TD>
                    <TD>
                        <asp:TextBox id=txtDate runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"MatDatePub") %>'></asp:TextBox></TD>
                </TR>
                <TR>
                    <TD>Description</TD>
                    <TD>
                        <asp:TextBox id=txtDesc runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"MatDesc") %>' TextMode="MultiLine">
                        </asp:TextBox></TD>
                </TR>
            </TABLE>
            <p>
                <asp:LinkButton id="lbnMatUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnMatDelete" runat="server" CommandName="
delete">Delete</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnMatCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></P>
        </EditItemTemplate>
    </asp:DataList>
    <P>
        <HR width="100%" SIZE="1">
    <P></P>
    <asp:Panel id="panArt" runat="server">
        <asp:DataList id="dlArt" runat="server" Width="100%">
            <ItemTemplate>
                <P>
                    <asp:Label id=lblJournal runat="server" Text='<%# DataBinder.
Eval(Container.DataItem,"JournalName") %>'>No Journal</asp:Label>.
                    <asp:Label id=lblVol runat="server" Text='<%# DataBinder.Eval(
Container.DataItem,"ArtVol") %>'>No Volume</asp:Label>,
                    <asp:Label id=lblNum runat="server" Text='<%# DataBinder.Eval(
Container.DataItem,"ArtNumber") %>'>No Number</asp:Label>.
                    pgs
                    <asp:Label id=lblStart runat="server" Text='<%# DataBinder.
Eval(Container.DataItem,"ArtStartPage") %>'>No Start</asp:Label>&nbsp;| -
                    <asp:Label id=lblEnd runat="server" Text='<%# DataBinder.Eval(

```

```

Container.DataItem,"ArtEndPage") %>'>No End</asp:Label></P>
    <p>
        <asp:LinkButton id="lbnArtEdit" runat="server" CommandName="
edit">Edit</asp:LinkButton></P>
    </ItemTemplate>
    <EditItemTemplate>
        <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="100%"
border="0">
            <TR>
                <TD>Journal</TD>
                <TD>
                    <asp:DropDownList id="ddlJournal" runat="server"></asp:
:DropDownList>&nbsp;or
                    enter
                    <asp:TextBox id="txtJournal" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"JournalName") %>'></asp:TextBox></TD>
                </TR>
                <TR>
                <TD>Volume</TD>
                <TD>
                    <asp:TextBox id="txtVol" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"ArtVol") %>'>
                    </asp:TextBox></TD>
                </TR>
                <TR>
                <TD>Number</TD>
                <TD>
                    <asp:TextBox id="txtNum" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"ArtNumber") %>'>
                    </asp:TextBox></TD>
                </TR>
                <TR>
                <TD>Start</TD>
                <TD>
                    <asp:TextBox id="txtStart" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"ArtStartPage") %>'>
                    </asp:TextBox></TD>
                </TR>
                <TR>
                <TD>End</TD>
                <TD>
                    <asp:TextBox id="txtEnd" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"ArtEndPage") %>'>
                    </asp:TextBox></TD>
                </TR>
            </TABLE>
            <p>
                <asp:LinkButton id="lbnArtUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnArtCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></P>
            </EditItemTemplate>
        </asp:DataList>
    </asp:Panel>
    <asp:Panel id="panBook" runat="server" Visible="False">
        <asp:DataList id="dlBook" runat="server">
            <ItemTemplate>
                <P>ISBN:
                <asp:Label id="lblISBN" runat="server" Text='<%# DataBinder.Eval
(Container.DataItem,"ISBN") %>'>No ISBN</asp:Label><BR>
                <asp:Label id="lblPub" runat="server" Text='<%# DataBinder.Eval(
Container.DataItem,"PubName") %>'>No Publisher</asp:Label>:
                <asp:Label id="lblCity" runat="server" Text='<%# DataBinder.Eval
(Container.DataItem,"City") %>'>No City</asp:Label></P>
                <p>
                    <asp:LinkButton id="lbnBookEdit" runat="server" CommandName="
edit">Edit</asp:LinkButton></P>

```

```

        </ItemTemplate>
        <EditItemTemplate>
            <TABLE id="Table3" cellSpacing="1" cellPadding="1" width="100%"
border="0">
                <TR>
                    <TD>ISBN</TD>
                    <TD>
                        <asp:TextBox id="txtISBN" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "ISBN") %>'>
                        </asp:TextBox></TD>
                    </TR>
                <TR>
                    <TD>Publisher</TD>
                    <TD>
                        <asp:DropDownList id="ddlPub" runat="server"></asp:
DropDownList>&nbsp;or enter
                        <asp:TextBox id="txtPub" runat="server"></asp:TextBox>
                    </TD>
                </TR>
                <TR>
                    <TD>City</TD>
                    <TD>
                        <asp:DropDownList id="ddlCity" runat="server"></asp:
DropDownList>&nbsp;or enter
                        <asp:TextBox id="txtCity" runat="server"></asp:
TextBox></TD>
                </TR>
            </TABLE>
            <p>
                <asp:LinkButton id="lbnBookUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|
                <asp:LinkButton id="lbnBookCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></p>
            </EditItemTemplate>
        </asp:DataList>
    </asp:Panel>
    <asp:Panel id="panUrl" runat="server" Visible="False">
        <asp:DataList id="dlUrl" runat="server">
            <ItemTemplate>
                <p>
                    <asp:Label id="lblAddress" runat="server" Text='<%# DataBinder.
Eval(Container.DataItem, "Url") %>'>No Address</asp:Label></p>
                    <p>
                        <asp:LinkButton id="lbnUrlEdit" runat="server" CommandName="
edit">Edit</asp:LinkButton></p>
                    </ItemTemplate>
                <EditItemTemplate>
                    <p>Address:
                        <asp:TextBox id="txtAddress" runat="server" Text='<%# DataBinder
.Eval(Container.DataItem, "Url") %>'>
                        </asp:TextBox></p>
                    <p>
                        <asp:LinkButton id="lbnUrlUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|
                        <asp:LinkButton id="lbnUrlCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></p>
                    </EditItemTemplate>
                </asp:DataList>
            </asp:Panel>
            <p></p>
            <HR width="100%" SIZE="1">
            <p></p>
            <p>Authors</p>
            <p>
                <asp:Label id="lblAuthors" runat="server">No Authors</asp:Label><BR>
                <asp:HyperLink id="linkAuthors" runat="server">Edit Authors</asp:
HyperLink></p>

```

```
<HR width="100%" SIZE="1">

<P>Categories</P>
<asp:CheckBoxList id="cbCats" runat="server" RepeatColumns="4" CssClass="
checkBoxList"></asp:CheckBoxList>
<asp:LinkButton id="lbnCatUpdate" text="Update Categories" runat="server"></asp:
LinkButton>
```

```
namespace Sharon.Controls.User
{
    using System;
    using System.Data;
    using System.Drawing;
    using System.Web;
    using System.Web.UI.WebControls;
    using System.Web.UI.HtmlControls;

    /// <summary>
    ///     Summary description for Head.
    /// </summary>
    public class Head : System.Web.UI.UserControl
    {
        protected System.Web.UI.WebControls.Label PageHeader;

        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
        }

        public string Heading
        {
            get{ return PageHeader.Text; }
            set{ PageHeader.Text = value; }
        }

        Web Form Designer generated code
    }
}
```

```
<%@ Control Language="c#" AutoEventWireup="false" Codebehind="Head.ascx.cs" Inherits="
Sharon.Controls.User.Head" TargetSchema="http://schemas.microsoft.com/intellisense/ie5
"%>
<!-- Header -->
<TABLE id="Table2" height="100" cellSpacing="0" cellPadding="1" width="95%" border="0">
  <TR>
    <TD bgColor="#666666"><IMG alt="" src="/Sharon/Images/banner.png"></TD>
  </TR>
</TABLE>
<TABLE id="Table1" cellSpacing="0" cellPadding="1" width="95%" border="0">
  <TR>
    <TD class="darkblue" width="20%" height="20">&nbsp;MENU OPTIONS</TD>
    <TD class="darkblue" width="5%" height="20">&nbsp;</TD>
    <TD class="darkblue" width="75%" colSpan="3" height="20">
      <asp:Label id="PageHeader" runat="server" CssClass="PageHeader">Books Database
    </asp:Label></TD>
  </TR>
</TABLE> <!-- Header End -->
```



```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Journals.
    /// </summary>
    public class Journals : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.Label lblMessage;
        protected System.Web.UI.WebControls.DataList dlJournals;
        protected System.Web.UI.WebControls.TextBox txtAddJournal;
        protected System.Web.UI.WebControls.Button btnAdd;

        private DataView JView;
        private DataSet JData;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                this.Title = "Journals";
                LoadJournals();
            }
        }

        Web Form Designer generated code

        private void btnAdd_Click(object sender, System.EventArgs e)
        {
            string msg;
            Journal journal = new Journal();
            if(journal.Add(txtAddJournal.Text) > 0)
            {
                //reload the cities
                LoadJournals();
                msg = txtAddJournal.Text + " added.";
            }
            else
            {
                msg = "Could not add " + txtAddJournal.Text + ".";
            }
            lblMessage.Text = msg;
        }

        private void LoadJournals()
        {
            Journal j = new Journal();
            JData = j.GetJournals();
            JView = JData.Tables["Journals"].DefaultView;
            dlJournals.DataSource = JView;
            dlJournals.DataBind();
        }

        protected void dlJournals_CancelCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
        {
            dlJournals.EditItemIndex = -1;
        }
    }
}

```

```

        LoadJournals();
    }

    protected void dlJournals_DeleteCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
    {
        string strID = ( (Label) e.Item.FindControl("lblJournalIDEdit") ).Text;
        string name = ( (TextBox) e.Item.FindControl("txtJournalEdit") ).Text;
        string msg;
        Journal journal = new Journal();
        dlJournals.EditItemIndex = -1;
        if( journal.Delete( Int32.Parse(strID.Trim()) ) )
        {
            msg = name + " deleted.";
        }
        else
        {
            msg = "Could not delete " + name + ".";
        }
        LoadJournals();
        lblMessage.Text = msg;
    }

    protected void dlJournals_EditCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
    {
        dlJournals.EditItemIndex = (int)e.Item.ItemIndex;
        LoadJournals();
    }

    protected void dlJournals_UpdateCommand(object source, System.Web.UI.WebControls.
DataListCommandEventArgs e)
    {
        string strID = ( (Label) e.Item.FindControl("lblJournalIDEdit") ).Text;
        string name = ( (TextBox) e.Item.FindControl("txtJournalEdit") ).Text;
        string msg;
        Journal journal = new Journal();
        dlJournals.EditItemIndex = -1;
        if( journal.Update( Int32.Parse(strID), name ) )
        {
            //reload the datalist
            msg = name + " updated.";
        }
        else
        {
            msg = "Could not update " + name + ".";
        }
        LoadJournals();
        lblMessage.Text = msg;
    }
}
}

```

```

<%@ Page language="c#" Codebehind="Journals.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Journals" %>
    <P>Enter Journal:
        <asp:textbox id="txtAddJournal" runat="server"></asp:textbox>
        <asp:button id="btnAdd" runat="server" Text="Add Journal"></asp:button>&
    nbsp;</P>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
        <asp:DataList id="dlJournals" runat="server" OnUpdateCommand="
dlJournals_UpdateCommand" OnEditCommand="dlJournals_EditCommand"
        OnDeleteCommand="dlJournals_DeleteCommand" OnCancelCommand="
dlJournals_CancelCommand">
            <ItemTemplate>
                <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300"
border="1">
                    <TR>
                        <TD vAlign="top">
                            <P>
                                <asp:Label id="lblJournalIDItem" runat="server" Text=
'<%# DataBinder.Eval(Container.DataItem, "JournalID") %>'>
                                </asp:Label><BR>
                                <asp:LinkButton id="lbnEdit" runat="server"
CommandName="edit">Edit</asp:LinkButton></P>
                            </TD>
                            <TD>
                                <asp:Label id="lblJournalItem" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem, "JournalName") %>'>
                                </asp:Label><BR>
                                <asp:Label id="lblJournalDateItem" runat="server" Text='
<%# DataBinder.Eval(Container.DataItem, "JournalLastUpdate") %>'>
                                </asp:Label></TD>
                            </TR>
                        </TABLE>
                    </ItemTemplate>
                    <EditItemTemplate>
                        <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="300"
border="1">
                            <TR>
                                <TD vAlign="top">
                                    <asp:Label id="lblJournalIDEdit" runat="server" Text='
<%# DataBinder.Eval(Container.DataItem, "JournalID") %>'>
                                    </asp:Label></TD>
                                <TD>
                                    <asp:TextBox id="txtJournalEdit" runat="server" Text='
<%# DataBinder.Eval(Container.DataItem, "JournalName") %>'>
                                    </asp:TextBox></TD>
                                </TR>
                            </TABLE>
                            <P align="center">
                                <asp:LinkButton id="lbnUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;|
                                <asp:LinkButton id="lbnDelete" runat="server" CommandName="
delete">Delete</asp:LinkButton>&nbsp;|&nbsp;
                                <asp:LinkButton id="lbnCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></P>
                        </EditItemTemplate>
                    </asp:DataList>

```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Materials.
    /// </summary>
    public class Materials : Sharon.BooksPage
    {
        protected Label lblMessage;
        protected Repeater rp;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                Title = "Materials";
                Material mat = new Material();
                DataSet ds = mat.GetMaterials();
                rp.DataSource =
                    ds.Tables[0].DefaultView;
                rp.DataBind();

                int count = ds.Tables[0].Rows.Count;
                if(count ==0)
                    lblMessage.Text = "No records found.";
                else if (count == 1)
                    lblMessage.Text = "One record found.";
                else
                    lblMessage.Text = count + " records found.";
            }
        }

        Web Form Designer generated code
    }
}
```

```

<%@ Page language="c#" Codebehind="Materials.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Materials" %>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <p></p>
    <asp:Repeater id="rp" runat="server">
        <HeaderTemplate>
            <table>
                <tr>
                    <th colspan="2">
                        All Materials</th></tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr>
                    <td valign="top">
                        <%# DataBinder.Eval(Container.DataItem, "MatID") %>
                    </td>
                    <td>
                        <a href='<%# "Details.aspx?MatID=" + DataBinder.Eval(Container.
DataItem, "MatID") %>'><%# DataBinder.Eval(Container.DataItem, "MatTitle") %></a>
                        <br />
                        <%# DataBinder.Eval(Container.DataItem, "MatDatePub") %>
                    </td>
                </tr>
            </ItemTemplate>
            <FooterTemplate>
                </table>
            </FooterTemplate>
        </asp:Repeater>

```

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Publishers.
    /// </summary>
    public class Publishers : Sharon.BooksPage
    {
        protected Label lblMessage;
        protected DataList dlPubs;
        protected TextBox txtAddPub;
        protected Button btnAdd;

        private DataView PubView;
        private DataSet PubData;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                LoadPubs();
                this.Title = "Publishers";
            }
        }

        Web Form Designer generated code

        protected void btnAdd_Click(object sender, System.EventArgs e)
        {
            string msg;
            Publisher pub = new Publisher();
            if(pub.Add(txtAddPub.Text)> 0)
            {
                //reload the cities
                LoadPubs();
                msg = txtAddPub.Text + " added.";
            }
            else
            {
                msg = "Could not add " + txtAddPub.Text + ".";
            }
            lblMessage.Text = msg;
        }

        private void LoadPubs()
        {
            Publisher pub = new Publisher();
            PubData = pub.GetPublishers();
            BindList();
        }

        private void BindList()
        {
            PubView = PubData.Tables["Publishers"].DefaultView;
            dlPubs.DataSource = PubView;
            dlPubs.DataBind();
        }
    }
}

```

```
}

protected void dlPubs_CancelCommand(object source, DataListCommandEventArgs e)
{
    dlPubs.EditItemIndex = -1;
    BindList();
}

protected void dlPubs_DeleteCommand(object source, DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblPubIDEdit") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtPubEdit") ).Text;
    string msg;
    Publisher pub = new Publisher();
    dlPubs.EditItemIndex = -1;
    if( pub.Delete( Int32.Parse(strID.Trim()) ) )
    {
        LoadPubs();
        msg = name + " deleted.";
    }
    else
    {
        BindList();
        msg = "Could not delete " + name + ".";
    }
    lblMessage.Text = msg;
}

protected void dlPubs_EditCommand(object source, DataListCommandEventArgs e)
{
    dlPubs.EditItemIndex = (int)e.Item.ItemIndex;
    LoadPubs();
}

protected void dlPubs_UpdateCommand(object source, DataListCommandEventArgs e)
{
    string strID = ( (Label) e.Item.FindControl("lblPubIDEdit") ).Text;
    string name = ( (TextBox) e.Item.FindControl("txtPubEdit") ).Text;
    string msg;
    Publisher pub = new Publisher();
    dlPubs.EditItemIndex = -1;
    if( pub.Update( Int32.Parse(strID), name ) )
    {
        //reload the cities
        LoadPubs();
        msg = name + " updated.";
    }
    else
    {
        msg = "Could not update " + name + ".";
    }
    lblMessage.Text = msg;
}
}
```

```

<%@ Page language="c#" Codebehind="Publishers.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Publishers" %>
    <P>Enter Pub:
    <asp:textbox id="txtAddPub" runat="server"></asp:textbox><asp:button id="
btnAdd" runat="server" Text="Add Pub"></asp:button>&nbsp;</P>
    <P>
    <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <asp:DataList id="dlPubs" runat="server" OnCancelCommand="
dlPubs_CancelCommand" OnDeleteCommand="dlPubs_DeleteCommand"
OnEditCommand="dlPubs_EditCommand" OnUpdateCommand="
dlPubs_UpdateCommand">
    <ItemTemplate>
    <TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300"
border="1">
        <TR>
            <TD vAlign="top">
                <P>
                    <asp:Label id=lblPubIDItem runat="server" Text='<%#
# DataBinder.Eval(Container.DataItem,"PubID") %>'>
                    </asp:Label><BR>
                    <asp:LinkButton id="lbnEdit" runat="server"
CommandName="edit">Edit</asp:LinkButton></P>
                </TD>
                <TD>
                    <asp:Label id=lblPubItem runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"PubName") %>'>
                    </asp:Label><BR>
                    <asp:Label id=lblPubDateItem runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"PubLastUpdate") %>'>
                    </asp:Label></TD>
                </TR>
            </TABLE>
        </ItemTemplate>
        <EditItemTemplate>
            <TABLE id="Table2" cellSpacing="1" cellPadding="1" width="300"
border="1">
                <TR>
                    <TD vAlign="top">
                        <asp:Label id="lblPubIDEdit" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"PubID") %>'>
                        </asp:Label></TD>
                    <TD>
                        <asp:TextBox id="txtPubEdit" runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"PubName") %>'>
                        </asp:TextBox></TD>
                    </TR>
                </TABLE>
                <P align="center">
                    <asp:LinkButton id="lbnUpdate" runat="server" CommandName="
update">Update</asp:LinkButton>&nbsp;<asp:LinkButton id="lbnDelete" runat="server" CommandName="
delete">Delete</asp:LinkButton>&nbsp;<asp:LinkButton id="lbnCancel" runat="server" CommandName="
cancel">Cancel</asp:LinkButton></P>
            </EditItemTemplate>
        </asp:DataList>

```



```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Sequence.
    /// </summary>
    public class Sequence : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.Label lblMessage;
        private int matID;
        protected System.Web.UI.WebControls.DataGrid dgSeq;
        private string matTitle;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(Request.Params.Get("MatID") != null)
            {
                matID = Int32.Parse(
                    Request.Params.Get("MatID")
                );
                Material mat = new Material();
                matTitle = mat.GetTitle(matID);
                if(!IsPostBack)
                {
                    this.Title = "Sequence";
                    lblMessage.Text = "Set Authors in Sequence for " + matTitle + ".";
                    LoadAuthors();
                }
            }
        } //end Page_Load

        private void LoadAuthors()
        {
            MaterialAuthor ma =
                new MaterialAuthor();
            DataSet maAuthors =
                ma.GetAuthorsPerMatID(matID);
            dgSeq.DataSource =
                maAuthors.Tables[0].DefaultView;
            dgSeq.DataBind();
        }

        Web Form Designer generated code

        private void dgSeq_CancelCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
        {
            dgSeq.EditItemIndex = -1;
            LoadAuthors();
        }

        private void dgSeq_EditCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
        {
            dgSeq.EditItemIndex =
                e.Item.ItemIndex;
        }
    }
}

```

```
        LoadAuthors();
    }

    private void dgSeq_UpdateCommand(object source, System.Web.UI.WebControls.
DataGridCommandEventArgs e)
    {
        int authMat, authID, sequence;
        TextBox txtSeq;
        authID = Int32.Parse(e.Item.Cells[0].Text);
        authMat = Int32.Parse(e.Item.Cells[2].Text);
        txtSeq = (TextBox)e.Item.Cells[3].Controls[0];
        sequence = Int32.Parse(txtSeq.Text);

        MaterialAuthor ma = new MaterialAuthor();
        ma.Update(authMat, matID, authID, sequence);
        dgSeq.EditItemIndex = -1;
        LoadAuthors();
    }
}
```

```
<%@ Page language="c#" Codebehind="Sequence.aspx.cs" AutoEventWireup="false" Inherits="
Sharon.Books.Sequence" %>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
        <asp:DataGrid id="dgSeq" runat="server" AutoGenerateColumns="False">
            <Columns>
                <asp:BoundColumn DataField="AuthID" HeaderText="Author ID" ReadOnly="True"
            >
                </asp:BoundColumn>
                <asp:BoundColumn DataField="FullName" HeaderText="Author" ReadOnly="True">
                </asp:BoundColumn>
                <asp:BoundColumn DataField="AuthMat" HeaderText="Author Material ID"
ReadOnly="True">
                </asp:BoundColumn>
                <asp:BoundColumn DataField="Sequence"></asp:BoundColumn>
                <asp:EditCommandColumn ButtonType="LinkButton" CancelText="Cancel"
EditText="Edit" UpdateText="Update"></asp:EditCommandColumn>
            </Columns>
        </asp:DataGrid>
```

```
/* Styles for the Spotlight */
```

```
.line { background-color: #FFFFFF; background-image: url(innerline.jpg); background-  
repeat: repeat-y; background-position: center center; }  
  
body  
{  
    margin-top: 0px;  
    padding-top: 0px;  
    background-color: #ffffff;  
}  
  
#ContentPanel  
{  
    padding-top: 15px;  
}  
  
.checkBoxList td  
{  
    vertical-align = top;  
}  
  
.blue { background-color: #336699; font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-weight: bold; color: #FFFFFF; text-align: left; }  
  
.darkgrey { background-color: #666666; font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-weight: bold; color: #FFFFFF; text-align: left; }  
  
.darkmaroon { background-color: #663366; font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-weight: bold; color: #FFFFFF; text-align: left; }  
  
.blue2 { background-color: #336699; font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-weight: bold; color: #FFFFFF; text-align: right; }  
  
.darkgrey2 { background-color: #666666; font-family: Arial, Helvetica, sans-serif; font-size: 12px; font-weight: bold; color: #FFFFFF; text-align: left; }  
  
.orangetext { color: #cc5500; font-size: 12px; }  
  
.orange { background-color: #cc5500; font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; font-weight: bold; color: #FFFFFF; }  
  
.orangelink { color: #cc5500; font-weight: bold; text-decoration: none; }  
  
td { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12px; }  
  
p { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 12px; line-height: 18; }
```

```
px; padding-top: 5px; padding-bottom: 0px; margin-top: 5px; margin-bottom: 0px}

.darkgrey { font-family: Arial, Helvetica, sans-serif; font-size: 20px; font-weight: bold;
; color: #FFFFFF; background-color: #333333; text-align: center}

.lightgreen { background-color: #CCCC99; font-family: Verdana, Arial, Helvetica, sans-
serif; font-size: 10px; color: #000000; text-align: left;}

.lightgreen2 { background-color: #CCCC99; font-family: Verdana, Arial, Helvetica, sans-
serif; font-size: 10px; color: #000000; text-align: center;}

.yellow { background-color: #FFFFCC; font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 11px; color: #000000}

.lightblue { background-color: #d1ecef; font-family: Verdana, Arial, Helvetica, sans-
serif; font-size: 10px; font-weight: bold; color: #000000}

.contest { background-color: #ccccff; font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 10px; color: #000000}

.lightorange { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color
: #000000; background-color: #fde5c0}

.innerline { background-image: url(images/innerline.jpg); background-repeat: repeat-y;
background-position: left top}

.line { background-color: #FFFFFF; background-image: url(innerline.jpg); background-
repeat: repeat-y; background-position: center center; }

/* style for the navbar */

.navbar { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 11px; color: #
ffffff; font-style: normal; font-variant: normal; }

a.navbar:link { color: #ffffff; font-weight: bold; text-decoration: none; }

a.navbar:visited { color: #ffffff; font-weight: bold; text-decoration: none; }

a.navbar:active { color:#ffffff; font-weight: bold; text-decoration: none; }

a.navbar:hover { color:#ffffff; font-weight: bold; text-decoration: none; }

/* style for the workplace */

.navbar2 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #
ffffff; font-style: normal; font-variant: normal; }
```

```
a.navbar2:link { color: #66144b; font-weight: bold; text-decoration: none; }
a.navbar2:visited { color: #66144b; font-weight: bold; text-decoration: none; }
a.navbar2:active { color:#66144b; font-weight: bold; text-decoration: none; }
a.navbar2:hover { color:#66144b; font-weight: bold; text-decoration: underline; }

/* style for the calendar */
.navbar3 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #
    ffffff; font-style: normal; font-variant: normal; }
a.navbar3:link { color: #669933; font-weight: bold; text-decoration: none; }
a.navbar3:visited { color: #669933; font-weight: bold; text-decoration: none; }
a.navbar3:active { color:#669933; font-weight: bold; text-decoration: none; }
a.navbar3:hover { color:#669933; font-weight: bold; text-decoration: underline; }

/* style for the viewpoint */
.navbar4 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #
    ffffff; font-style: normal; font-variant: normal; }
a.navbar4:link { color: #185171; font-weight: bold; text-decoration: none; }
a.navbar4:visited { color: #185171; font-weight: bold; text-decoration: none; }
a.navbar4:active { color:#185171; font-weight: bold; text-decoration: none; }
a.navbar4:hover { color:#185171; font-weight: bold; text-decoration: underline; }

/* style for the spotlight */
.navbar5 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #
    ffffff; font-style: normal; font-variant: normal; }
a.navbar5:link { color: #7d4412; font-weight: bold; text-decoration: none; }
a.navbar5:visited { color: #7d4412; font-weight: bold; text-decoration: none; }
a.navbar5:active { color:#7d4412; font-weight: bold; text-decoration: none; }
a.navbar5:hover { color:#7d4412; font-weight: bold; text-decoration: underline; }

/* style for the news */
.navbar6 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #
    ffffff; font-style: normal; font-variant: normal; }
a.navbar6:link { color: #336633; font-weight: bold; text-decoration: none; }
a.navbar6:visited { color: #336633; font-weight: bold; text-decoration: none; }
a.navbar6:active { color:#336633; font-weight: bold; text-decoration: none; }
```

```
a.navbar6:hover { color:#336633; font-weight: bold; text-decoration: underline; }

/* style for the announcements */

.announcetext { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #645b0f; }

.navbar7 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 10px; color: #ffffff; font-style: normal; font-variant: normal; }

a.navbar7:link { color: #645b0f; font-weight: bold; text-decoration: none; }
a.navbar7:visited { color: #645b0f; font-weight: bold; text-decoration: none; }
a.navbar7:active { color:#645b0f; font-weight: bold; text-decoration: none; }
a.navbar7:hover { color:#645b0f; font-weight: bold; text-decoration: underline; }

/* styles for the headers */

h1 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 18px; padding-top: 0px; padding-bottom: 0px; margin-top: 10px; margin-bottom: 0px}
h2 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 16px; padding-top: 0px; padding-bottom: 0px; margin-top: 15px; margin-bottom: 0px}
h3 { font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 14px; padding-top: 0px; padding-bottom: 0px; margin-top: 15px; margin-bottom: 0px}

.footer { color: #666666; font-size: 10px; }

.sidebars { color: #000000; font-size: 11px; }

/* For the line repeat */

.line { background-color: #FFFFFF; background-image: url(images/blackinner.jpg); background-repeat: repeat-y; background-position: center center}

/* Styles for the CEE Intranet */

.blackbehind { background-color: #000000; }

/* Style for home page */

.burntorange { background-color: #CC6600; color: #ffffff; font-size: 11px; font-family: Verdana, Arial, Helvetica, sans-serif; font-weight: bold; }
```

```
/* Lighter style for home page */
```

```
.burntlight { background-color: #FFCC9C; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
.burntlight2 { background-color: #FCEBDB; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
/* Divs for the hover on home page */
```

```
div#burnt a {display: block; font: bold 11px sans-serif; text-decoration: none; color: #000066; padding: 5px; }
```

```
div#burnt a:hover {color: #411; background-color: #FCEBDB; padding: 5px; }
```

```
/* Style for logoff */
```

```
.logoff { font-size: 11px; font-weight: bold; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
a.logoff:link { color: #ffffff; font-weight: bold; text-decoration: none; }
```

```
a.logoff:visited { color: #ffffff; font-weight: bold; text-decoration: none; }
```

```
a.logoff:active { color:#ffffff; font-weight: bold; text-decoration: none; }
```

```
a.logoff:hover { color:#ffffff; font-weight: bold; text-decoration: underline; }
```

```
/* Style for news page */
```

```
.darkgreen { background-color: #336633; color: #ffffff; font-weight: bold; font-size: 11px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
/* Lighter style for news page */
```

```
.firstgreen { background-color: #99cc99; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
.secondgreen { background-color: #def8e2; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
.burntlight2 { background-color: #FCEBDB; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```



```
/* Divs for the hover on news page */

div#green a {display: block; font: bold 11px sans-serif;

    text-decoration: none; color: #000066; padding: 5px; }

div#green a:hover {color: #411; background-color: #def8e2; padding: 5px; }


/* Style for spotlight page */

.darkbrown { background-color: #7d4412; color: #ffffff; font-weight: bold; font-size: 11px;
    ; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Lighter style for spotlight page */

.firstbrown { background-color: #cb9362; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

.secondbrown { background-color: #f9dbc0; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

/* Style for directories page */

.darkpurple { background-color: #66144b; color: #ffffff; font-weight: bold; font-size: 11
    px; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Lighter style for directories page */

.firstpurple { background-color: #b47fa1; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

.secondpurple { background-color: #e9c0dc; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

/* Style for workplace page */

.darkcyan { background-color: #11826e; color: #ffffff; font-weight: bold; font-size: 11px;
    font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Lighter style for worlplace page */

.firstcyan { background-color: #c0eee6; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

.secondcyan { background-color: #c0eee6; color: #000000; font-size: 10px; font-family:
    Verdana, Arial, Helvetica, sans-serif; }

/* Style for surveys page */

.darkpoop { background-color: #8d8855; color: #ffffff; font-weight: bold; font-size: 11px;
    font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
/* Lighter style for surveys page */
```

```
.firstpoop { background-color: #BCB788; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
.secondpoop { background-color: #F1ECBF; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
.burntlight2 { background-color: #FCEBDB; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
/* Divs for the hover on spotlight page */
```

```
div#brown a {display: block; font: bold 11px sans-serif;
text-decoration: none; color: #000066; padding: 5px; }
```

```
div#brown a:hover {color: #411; background-color: #f9dbc0; padding: 5px; }
```

```
/* Divs for the hover on directories page */
```

```
div#purple a {display: block; font: bold 11px sans-serif;
text-decoration: none; color: #000066; padding: 5px; }
```

```
div#purple a:hover {color: #411; background-color: #e9c0dc; padding: 5px; }
```

```
/* Divs for the hover on workplace page */
```

```
div#cyan a {display: block; font: bold 11px sans-serif;
text-decoration: none; color: #000066; padding: 5px; }
```

```
div#cyan a:hover {color: #411; background-color: #e2f8f4; padding: 5px; }
```

```
/* Divs for the hover on surveys page */
```

```
div#poop a {display: block; font: bold 11px sans-serif;
text-decoration: none; color: #000066; padding: 5px; }
```

```
div#poop a:hover {color: #411; background-color: #F1ECBF; padding: 5px; }
```

```
/* Style for calendar page */
```

```
.darkapple { background-color: #669933; color: #ffffff; font-weight: bold; font-size: 11px; font-family: Verdana, Arial, Helvetica, sans-serif; }
```

```
/* Lighter style for calendar page */
```

```
.firstapple { background-color: #99cc66; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
.secondapple { background-color: #ccff99; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

.firstyellow2 { background-color: #ccff99; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

.burntlight2 { background-color: #FCEBDB; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Divs for the hover on calendar page */
div#apple a {display: block; font: bold 11px sans-serif;
    text-decoration: none; color: #000066; padding: 5px; }
div#apple a:hover {color: #411; background-color: #ccff99; padding: 5px; }

/* Style for viewpoint page */
.darkblue { background-color: #185171; color: #ffffff; font-weight: bold; font-size: 11px; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Lighter style for viewpoint page */
.firstblue { background-color: #9DC1D5; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }
.secondblue { background-color: #d2e4ee; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

.burntlight2 { background-color: #FCEBDB; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

.burntlight3 { background-color: #d2e4ee; color: #000000; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* Divs for the hover on viewpoint page */
div#blue a {display: block; font: bold 11px sans-serif;
    text-decoration: none; color: #000066; padding: 5px; }
div#blue a:hover {color: #411; background-color: #d2e4ee; padding: 5px; }

/* For smaller sized text */
```

```
.smalltext { font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; }

/* For the orange colored text in the calendar */

.smalltext2 { font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; color: #cc5500; font-weight: bold }.palegreen {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: 10px;
    color: #000000;
    background-color: DEF8E2;
}

/* Colors for the tables in viewpoint and the spotlight */

.viewpointblue { background-color: #E4E7EA; font-size: 12px; font-family: Verdana, Arial, Helvetica, sans-serif; font-weight: bold; color: #000000; }

.spotlightbrown { background-color: #F6EEDD; font-size: 12px; font-family: Verdana, Arial, Helvetica, sans-serif; font-weight: bold; color: #000000; }

.newsidegreen { background-color: #DCE9D9; font-size: 10px; font-family: Verdana, Arial, Helvetica, sans-serif; color: #000000; }
```

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Merazzle.Books.Business;

namespace Sharon.Books
{
    /// <summary>
    /// Summary description for Urls.
    /// </summary>
    public class Urls : Sharon.BooksPage
    {
        protected System.Web.UI.WebControls.Repeater rp;
        protected System.Web.UI.WebControls.Label lblMessage;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if(!IsPostBack)
            {
                Url url = new Url();
                DataSet ds = url.GetUrls();

                rp.DataSource =
                    ds.Tables[0].DefaultView;
                rp.DataBind();
                Title = "Urls";

                int count = ds.Tables[0].Rows.Count;
                if(count ==0)
                    lblMessage.Text = "No records found.";
                else if (count == 1)
                    lblMessage.Text = "One record found.";
                else
                    lblMessage.Text = count + " records found.";
            }
        }

        Web Form Designer generated code
    }
}
```

```

<%@ Page language="c#" Codebehind="Urls.aspx.cs" AutoEventWireup="false" Inherits="Sharon.
Books.Urls" %>
    <P>
        <asp:Label id="lblMessage" runat="server"></asp:Label></P>
    <p></p>
    <asp:Repeater id="rp" runat="server">
        <HeaderTemplate>
            <table>
                <tr>
                    <th colspan="2">
                        All Urls</th></tr>
            </HeaderTemplate>
            <ItemTemplate>
                <tr>
                    <td valign="top">
                        <%# DataBinder.Eval(Container.DataItem, "MatID") %>
                    </td>
                    <td>
                        <a href='<%# "Details.aspx?MatID=" + DataBinder.Eval(
Container.DataItem, "MatID") %>'>
                            <%# DataBinder.Eval(Container.DataItem, "MatTitle") %>
                        </a>
                        <br />
                        <%# DataBinder.Eval(Container.DataItem, "MatDatePub") %>
                    </td>
                </tr>
            </ItemTemplate>
            <FooterTemplate>
            </table>
            </FooterTemplate>
        </asp:Repeater>

```

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ConnString" value="server=***;database=***;uid=***;pwd=****" />
    <add key="Root" value="/Sharon" />
      <add key="MainHeader" value="/Controls/user/Head.ascx" />
      <add key="MainFooter" value="/controls/user/Foot.ascx" />
      <add key="BooksSide" value="/Books/BooksNav.ascx" />
      <add key="DefaultStyleSheet" value="/styles/styles.css" />
      <add key="JavaScript" value="/scripts/scripts.js" />
    </add>
  </appSettings>

  <system.web>

    <compilation
      defaultLanguage="c#"
      debug="true"
    />

    <customErrors
      mode="Off"
    />

    <authentication mode="Windows" />

    <authorization>
      <allow users="*" />
    </authorization>
    <trace
      enabled="false"
      requestLimit="10"
      pageOutput="false"
      traceMode="SortByTime"
      localOnly="true"
    />
    <sessionState
      mode="InProc"
      stateConnectionString="tcpip=127.0.0.1:42424"
      sqlConnectionString="data source=127.0.0.1;Trusted_Connection=yes"
      cookieless="false"
      timeout="20"
    />

  </system.web>
</configuration>
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Article.
    /// </summary>
    public class Article: BusinessObject
    {
        private int _matID, _journalID, _vol, _num, _start, _end;
        public Article(){}
        public Article(int inJournalID, int inVol, int inNum, int inStart, int inEnd)
        {
            _journalID = inJournalID;
            _vol = inVol;
            _num = inNum;
            _start = inStart;
            _end = inEnd;
        }
        public Article(int inMatID, int inJournalID, int inVol, int inNum, int inStart,
int inEnd)
        {
            _matID = inMatID;
            _journalID = inJournalID;
            _vol = inVol;
            _num = inNum;
            _start = inStart;
            _end = inEnd;
        }

        //property
        public int MatID
        {
            get{ return _matID;}
            set{ _matID = value;}
        }

        //methods
        public int Add()
        {
            return Add(_matID, _journalID, _vol, _num, _start, _end);
        }
        public int Add(int matID, int journalID, int vol, int num, int start, int end)
        {
            Data.Article art =
                new Merazzle.Books.Data.Article(ConnString);
            return art.Add(matID, journalID, vol, num, start, end);
        }//end Add

        public bool Update(int matID, int journalID, int vol, int num, int start, int end)
        {
            Data.Article art =
                new Merazzle.Books.Data.Article(ConnString);
            return art.Update(matID, journalID, vol, num, start, end);
        }//end Update

        public bool Delete(int id)
        {
            Data.Article art =
                new Merazzle.Books.Data.Article(ConnString);
            return art.Delete(id);
        }//end Delete

        public DataSet GetArticles()
```



```
{
    Data.Article art =
        new Merazzle.Books.Data.Article(ConnString);
    return art.GetArticles();
} //end GetArticles
} //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Author.
    /// </summary>
    public class Author: BusinessObject
    {
        public Author(){}

        ///methods
        public int Add(string firstName, string lastName, string bio)
        {
            Data.Author author =
                new Merazzle.Books.Data.Author(ConnString);
            return author.Add(firstName,lastName, bio);
        }//end Add

        public bool Update(int id, string firstName, string lastName, string bio)
        {
            Data.Author author =
                new Merazzle.Books.Data.Author(ConnString);
            return author.Update(id, firstName, lastName, bio);
        }//end Update

        public bool Delete(int id)
        {
            Data.Author author =
                new Merazzle.Books.Data.Author(ConnString);
            return author.Delete(id);
        }//end Delete

        public DataSet GetAuthors()
        {
            Data.Author author =
                new Merazzle.Books.Data.Author(ConnString);
            return author.GetAuthors();
        }//end GetAuthors

        public DataSet GetAuthorNames()
        {
            Data.Author author =
                new Merazzle.Books.Data.Author(ConnString);
            return author.GetAuthorNames();
        }//end GetAuthors
    }//end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Book.
    /// </summary>
    public class Book: BusinessObject
    {
        private string _isbn;
        private int _matID, _pubID, _cityID;

        //Constructors
        public Book(){}
        public Book(int inPubID, int inCityID, string inIsbn)
        {
            _isbn = inIsbn;
            _pubID = inPubID;
            _cityID = inCityID;
        }

        //property
        public int MatID
        {
            get{ return _matID;}
            set{ _matID = value;}
        }

        //methods
        public bool Add()
        {
            return Add(_matID, _pubID, _cityID, _isbn);
        }
        public bool Add(int matID, int pubID, int cityID, string isbn)
        {
            Data.Book book =
                new Merazzle.Books.Data.Book(ConnString);
            return book.Add(matID, pubID, cityID, isbn);
        } //end Add

        public bool Update(int matID, int pubID, int cityID, string isbn)
        {
            Data.Book book =
                new Merazzle.Books.Data.Book(ConnString);
            return book.Update(matID, pubID, cityID, isbn);
        } //end Update

        public bool Delete(int matID)
        {
            Data.Book book =
                new Merazzle.Books.Data.Book(ConnString);
            return book.Delete(matID);
        } //end Delete

        public DataSet GetBooks()
        {
            Data.Book book =
                new Merazzle.Books.Data.Book(ConnString);
            return book.GetBooks();
        } //end GetBooks
    } //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Category.
    /// </summary>
    public class Category: BusinessObject
    {
        public Category(){}

        ///methods

        public DataSet GetCategories()
        {
            Data.Category category =
                new Merazzle.Books.Data.Category(ConnString);
            return category.GetCategories();
        }//end GetCategories

        public string GetCategory(int catID)
        {
            Data.Category category =
                new Merazzle.Books.Data.Category(ConnString);
            return category.GetCategory(catID);
        }//end GetCategory

        public bool Add(string name, string description)
        {
            Data.Category category =
                new Merazzle.Books.Data.Category(ConnString);
            return (category.Add(name, description) > 0);
        }//end Add

        public bool Update(int id, string name, string description)
        {
            Data.Category category =
                new Merazzle.Books.Data.Category(ConnString);
            return category.Update(id, name, description);
        }//end Update

        public bool Delete(int id)
        {
            Data.Category category =
                new Merazzle.Books.Data.Category(ConnString);
            return category.Delete(id);
        }//end Delete
    }//end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for City.
    /// </summary>
    public class City: BusinessObject
    {
        public City(){}

        //methods
        public int Add(string name)
        {
            Data.City city =
                new Merazzle.Books.Data.City(ConnString);
            return city.Add(name);
        } //end Add

        public bool Update(int id, string name)
        {
            Data.City city =
                new Merazzle.Books.Data.City(ConnString);
            return city.Update(id,name);
        } //end Update

        public bool Delete(int id)
        {
            Data.City city =
                new Merazzle.Books.Data.City(ConnString);
            return city.Delete(id);
        } //end Delete

        public DataSet GetCities()
        {
            Data.City city =
                new Merazzle.Books.Data.City(ConnString);
            return city.GetCities();
        } //end GetCities
    } //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Journal.
    /// </summary>
    public class Journal: BusinessObject
    {
        public Journal(){}

        //methods
        public int Add(string name)
        {
            Data.Journal journal =
                new Merazzle.Books.Data.Journal(ConnString);
            return journal.Add(name);
        } //end Add

        public bool Update(int id, string name)
        {
            Data.Journal journal =
                new Merazzle.Books.Data.Journal(ConnString);
            return journal.Update(id,name);
        } //end Update

        public bool Delete(int id)
        {
            Data.Journal journal =
                new Merazzle.Books.Data.Journal(ConnString);
            return journal.Delete(id);
        } //end Delete

        public DataSet GetJournals()
        {
            Data.Journal journal =
                new Merazzle.Books.Data.Journal(ConnString);
            return journal.GetJournals();
        } //end GetJournals
    }
}
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business{

    public class Material: BusinessObject{

        private string _title, _description;
        private DateTime _date;

        //Constructors
        public Material(){}
        public Material(string inTitle, string inDesc, DateTime inDate){
            _title = inTitle;
            _description = inDesc;
            _date = inDate;
        }

        //methods
        public DataSet GetMaterials(){
            Data.Material material =
                new Merazzle.Books.Data.Material(ConnString);
            return material.GetMaterials();
        } //end GetMaterials

        public DataSet GetMaterialByID(int matID){
            Data.Material material =
                new Merazzle.Books.Data.Material(ConnString);
            return material.GetMaterialByID(matID);
        }

        public string GetTitle(int matID){
            Data.Material mat =
                new Data.Material(ConnString);
            return mat.GetTitle(matID);
        }
        public int Add(){
            return Add(_title, _description, _date);
        }

        public int Add(string title, string description, DateTime date){
            Data.Material material =
                new Merazzle.Books.Data.Material(ConnString);
            return material.Add(title,description, date);
        } //end Add

        public bool Update(int id, string title, string description, DateTime date){
            Data.Material material =
                new Merazzle.Books.Data.Material(ConnString);
            return material.Update(id, title, description, date);
        } //end Update

        public bool Delete(int id){
            Data.Material material =
                new Merazzle.Books.Data.Material(ConnString);
            return material.Delete(id);
        } //end Delete
    } //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for MaterialAuthor.
    /// </summary>
    public class MaterialAuthor: BusinessObject
    {
        public MaterialAuthor(){}

        ///methods

        public DataSet GetMaterialAuthors()
        {
            Data.MaterialAuthor materialAuthor =
                new Merazzle.Books.Data.MaterialAuthor(ConnString);
            return materialAuthor.GetMaterialAuthors();
        }
        public DataSet GetAuthorsPerMatID(int matID)
        {
            Data.MaterialAuthor materialAuthor =
                new Merazzle.Books.Data.MaterialAuthor(ConnString);
            return materialAuthor.GetAuthorsPerMatID(matID);
        } //end GetAuthorsPerMatID

        public int Add(int matID, int authId, int sequence)
        {
            Data.MaterialAuthor materialAuthor =
                new Merazzle.Books.Data.MaterialAuthor(ConnString);
            return materialAuthor.Add(matID, authId, sequence);
        }
        public bool Update(int authMat, int matID, int authId, int sequence)
        {
            Data.MaterialAuthor materialAuthor =
                new Merazzle.Books.Data.MaterialAuthor(ConnString);
            return materialAuthor.Update(authMat, matID, authId, sequence);
        }
        public bool Delete( int authMat )
        {
            Data.MaterialAuthor materialAuthor =
                new Merazzle.Books.Data.MaterialAuthor(ConnString);
            return materialAuthor.Delete(authMat);
        }
    } //end class
} //end namespace
```



```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for MaterialCategory.
    /// </summary>
    public class MaterialCategory: BusinessObject
    {
        public MaterialCategory(){}

        public DataSet GetMaterialCategories()
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.GetMaterialCategories();
        }//end GetMaterialCategories

        public DataSet GetCategoriesPerMatID(int matID)
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.GetCategoriesPerMatID(matID);
        }//end GetCategoriesPerMatID

        public DataSet GetMaterialsPerCatID(int catID)
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.GetMaterialsPerCatID(catID);
        }//end GetMaterialsPerCatID

        public int Add(int matID, int catID)
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.InsertMaterialCategory(matID, catID);
        }//end Add

        public bool Update(int matCatID, int matID, int catID)
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.UpdateMaterialCategory(matCatID, matID, catID);
        }//end Update

        public bool Delete(int matCatID)
        {
            Data.MaterialCategory mc =
                new Data.MaterialCategory(ConnString);
            return mc.DeleteMaterialCategory(matCatID);
        }//end Delete
    }//end class
} //end namespace
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Publisher.
    /// </summary>
    public class Publisher: BusinessObject
    {
        public Publisher(){}

        ///methods
        public int Add(string name)
        {
            Data.Publisher pub =
                new Merazzle.Books.Data.Publisher(ConnString);
            return pub.Add(name);
        }//end Add

        public bool Update(int id, string name)
        {
            Data.Publisher pub =
                new Merazzle.Books.Data.Publisher(ConnString);
            return pub.Update(id,name);
        }//end Update

        public bool Delete(int id)
        {
            Data.Publisher pub =
                new Merazzle.Books.Data.Publisher(ConnString);
            return pub.Delete(id);
        }//end Delete

        public DataSet GetPublishers()
        {
            Data.Publisher pub =
                new Merazzle.Books.Data.Publisher(ConnString);
            return pub.GetPublishers();
        }//end GetPublishers
    }
}
```

```
using System;
using System.Data;
using System.Configuration;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Business
{
    /// <summary>
    /// Summary description for Url.
    /// </summary>
    public class Url: BusinessObject
    {
        private string _address;
        private int _matID;

        public Url(){}
        public Url(string address)
        {
            _address = address;
        }

        //property
        public int MatID
        {
            get{return _matID;}
            set{_matID = value;}
        }

        //methods
        public bool Add()
        {
            return Add(_matID, _address);
        }
        public bool Add(int id, string address)
        {
            Data.Url url =
                new Merazzle.Books.Data.Url(ConnString);
            return url.Add(id, address);
        }//end Add

        public bool Update(int id, string address)
        {
            Data.Url url =
                new Merazzle.Books.Data.Url(ConnString);
            return url.Update(id,address);
        }//end Update

        public bool Delete(int id)
        {
            Data.Url url =
                new Merazzle.Books.Data.Url(ConnString);
            return url.Delete(id);
        }//end Delete

        public DataSet GetUrls()
        {
            Data.Url url =
                new Merazzle.Books.Data.Url(ConnString);
            return url.GetUrls();
        }//end GetUrls
    }//end class
}//end namespace
```

```
using System;
using System.Configuration;
using System.Web;

namespace Merazzle.WebModules.Core
{
    public class BusinessObject
    {
        private string connString, root;

        public string ConnString
        {
            get{ return connString;}
            set{ connString = value;}
        }
        public string Root
        {
            get{ return root;}
            set{ root = value;}
        }

        public BusinessObject()
        {
            connString = ConfigurationSettings.AppSettings["ConnString"];
            root = ConfigurationSettings.AppSettings["Root"];
        }
    }
}
```

```

using System;
using System.Data;
using System.Data.SqlClient;

namespace Merazzle.WebModules.Core
{
    /// <summary>
    /// DataObject is the class from which all classes in the Data Services
    /// Tier inherit. The core functionality of establishing a connection
    /// with the database and executing simple stored procedures is also
    /// provided by this base class.
    /// </summary>
    public abstract class DataObject
    {
        protected SqlConnection Connection;
        private string connectionString;

        /// <summary>
        /// A parameterized constructor, it allows us to take a connection
        /// string as a constructor argument, automatically instantiating
        /// a new connection.
        /// </summary>
        /// <param name="newConnectionString">Connection String to the associated database
        </param>
        public DataObject( string newConnectionString )
        {
            connectionString = newConnectionString;
            Connection = new SqlConnection( connectionString );
        }

        /// <summary>
        /// Protected property that exposes the connection string
        /// to inheriting classes. Read-Only.
        /// </summary>
        protected string ConnectionString
        {
            get
            {
                return connectionString;
            }
        }

        /// <summary>
        /// Private routine allowed only by this base class, it automates the task
        /// of building a SqlCommand object designed to obtain a return value from
        /// the stored procedure.
        /// </summary>
        /// <param name="storedProcName">Name of the stored procedure in the DB, eg.
        sp_DoTask</param>
        /// <param name="parameters">Array of IDataParameter objects containing parameters
        to the stored proc</param>
        /// <returns>Newly instantiated SqlCommand instance</returns>
        private SqlCommand BuildIntCommand( string storedProcName, IDataParameter[]
parameters)
        {
            SqlCommand command = BuildQueryCommand( storedProcName, parameters );

            command.Parameters.Add( new SqlParameter ( "ReturnValue",
                SqlDbType.Int,
                4, /* Size */
                ParameterDirection.ReturnValue,
                false, /* is nullable */
                0, /* byte precision */
                0, /* byte scale */
                string.Empty,
                DataRowVersion.Default,
            ) );
        }
    }
}

```

```

        null );

    return command;
}

/// <summary>
/// Builds a SqlCommand designed to return a SqlDataReader, and not
/// an actual integer value.
/// </summary>
/// <param name="storedProcName">Name of the stored procedure</param>
/// <param name="parameters">Array of IDataParameter objects</param>
/// <returns></returns>
private SqlCommand BuildQueryCommand(string storedProcName, IDataParameter[]
parameters)
{
    SqlCommand command = new SqlCommand( storedProcName, Connection );
    command.CommandType = CommandType.StoredProcedure;

    foreach (SqlParameter parameter in parameters)
    {
        command.Parameters.Add( parameter );
    }

    return command;
}

/// <summary>
/// Runs a stored procedure, can only be called by those classes deriving
/// from this base. It returns an integer indicating the return value of the
/// stored procedure, and also returns the value of the RowsAffected aspect
/// of the stored procedure that is returned by the ExecuteNonQuery method.
/// </summary>
/// <param name="storedProcName">Name of the stored procedure</param>
/// <param name="parameters">Array of IDataParameter objects</param>
/// <param name="rowsAffected">Number of rows affected by the stored procedure.</
param>
/// <returns>An integer indicating return value of the stored procedure</returns>
protected int RunProcedure(string storedProcName, IDataParameter[] parameters, out
int rowsAffected )
{
    int result;

    Connection.Open();
    SqlCommand command = BuildIntCommand( storedProcName, parameters );
    rowsAffected = command.ExecuteNonQuery();
    result = (int)command.Parameters["ReturnValue"].Value;
    Connection.Close();
    return result;
}

/// <summary>
/// Will run a stored procedure, can only be called by those classes deriving
/// from this base. It returns a SqlDataReader containing the result of the stored
/// procedure.
/// </summary>
/// <param name="storedProcName">Name of the stored procedure</param>
/// <param name="parameters">Array of parameters to be passed to the procedure</
param>
/// <returns>A newly instantiated SqlDataReader object</returns>
protected SqlDataReader RunProcedure(string storedProcName, IDataParameter[]
parameters )
{
    SqlDataReader returnReader;

    Connection.Open();

```

```

        SqlCommand command = BuildQueryCommand( storedProcName, parameters );
        command.CommandType = CommandType.StoredProcedure;

        returnReader = command.ExecuteReader();
        //Connection.Close();
        return returnReader;
    }

    /// <summary>
    /// Creates a DataSet by running the stored procedure and placing the results
    /// of the query/proc into the given tablename.
    /// </summary>
    /// <param name="storedProcName"></param>
    /// <param name="parameters"></param>
    /// <param name="tableName"></param>
    /// <returns></returns>
    protected DataSet RunProcedure( string storedProcName, IDataParameter[] parameters,
    string tableName )
    {
        DataSet dataSet = new DataSet();
        Connection.Open();
        SqlDataAdapter sqlDA = new SqlDataAdapter();
        sqlDA.SelectCommand = BuildQueryCommand( storedProcName, parameters );
        sqlDA.Fill( dataSet, tableName );
        Connection.Close();

        return dataSet;
    }

    /// <summary>
    /// Takes an -existing- dataset and fills the given table name with the results
    /// of the stored procedure.
    /// </summary>
    /// <param name="storedProcName"></param>
    /// <param name="parameters"></param>
    /// <param name="dataSet"></param>
    /// <param name="tableName"></param>
    /// <returns></returns>
    protected void RunProcedure( string storedProcName, IDataParameter[] parameters,
    DataSet dataSet, string tableName )
    {
        Connection.Open();
        SqlDataAdapter sqlDA = new SqlDataAdapter();
        sqlDA.SelectCommand = BuildIntCommand( storedProcName, parameters );
        sqlDA.Fill( dataSet, tableName );
        Connection.Close();
    }
}

```

```

using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class Article: DataObject{
        public Article(string connString):base(connString){}

        public DataSet GetArticles(){
            return RunProcedure("BK_GetArticles",
                new SqlParameter[0], "Articles");
        }//end GetArticles

        public int Add(int matID, int journalID, int vol, int num, int start, int end){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@JournalID", SqlDbType.Int, 4),
new SqlParameter("@ArtVol", SqlDbType.Int, 4),
new SqlParameter("@ArtNumber", SqlDbType.Int, 4),
new SqlParameter("@ArtStartPage", SqlDbType.Int, 4),
new SqlParameter("@ArtEndPage", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            parameters[1].Value = journalID;
            parameters[2].Value = vol;
            parameters[3].Value = num;
            parameters[4].Value = start;
            parameters[5].Value = end;

            return RunProcedure("BK_InsertArticle",
                parameters, out rowsAffected);
        }//end add

        public bool Update(int matID, int journalID, int vol, int num, int start, int end)↵
        {
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@JournalID", SqlDbType.Int, 4),
new SqlParameter("@ArtVol", SqlDbType.Int, 4),
new SqlParameter("@ArtNumber", SqlDbType.Int, 4),
new SqlParameter("@ArtStartPage", SqlDbType.Int, 4),
new SqlParameter("@ArtEndPage", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            parameters[1].Value = journalID;
            parameters[2].Value = vol;
            parameters[3].Value = num;
            parameters[4].Value = start;
            parameters[5].Value = end;
            RunProcedure("BK_UpdateArticle",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int matID){
            int numAffected;
            SqlParameter[] parameters =
{new SqlParameter("@MatID", SqlDbType.Int, 4)};
            parameters[0].Value = matID;

            RunProcedure("BK_DeleteArticle",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }
}

```



```

using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;
namespace Merazzle.Books.Data{
    public class Author: DataObject
    {
        public Author(string connString):base(connString){}

        public DataSet GetAuthors(){
            return RunProcedure("BK_GetAuthors",
                new SqlParameter[0], "Authors");
        }//end GetAuthors

        public DataSet GetAuthorNames(){
            return RunProcedure("BK_GetAuthorNames",
                new SqlParameter[0], "Authors");
        }//end GetAuthorNames

        public int Add(string firstName, string lastName, string bio){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@AuthID", SqlDbType.Int, 4),
new SqlParameter("@AuthFirstName", SqlDbType.VarChar, 25),
new SqlParameter("@AuthLastName", SqlDbType.VarChar, 25),
new SqlParameter("@AuthBio", SqlDbType.VarChar, 265)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = firstName;
            parameters[2].Value = lastName;
            parameters[3].Value = bio;

            RunProcedure("BK_InsertAuthor",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        }//end add

        public bool Update(int authID, string firstName, string lastName, string bio){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@AuthID", SqlDbType.Int, 4),
new SqlParameter("@AuthFirstName", SqlDbType.VarChar, 25),
new SqlParameter("@AuthLastName", SqlDbType.VarChar, 25),
new SqlParameter("@AuthBio", SqlDbType.VarChar, 265)
            };
            parameters[0].Value = authID;
            parameters[1].Value = firstName;
            parameters[2].Value = lastName;
            parameters[3].Value = bio;
            RunProcedure("BK_UpdateAuthor",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int authID){
            int numAffected;
            SqlParameter[] parameters =
            {new SqlParameter("@AuthID", SqlDbType.Int, 4)};
            parameters[0].Value = authID;

            RunProcedure("BK_DeleteAuthor",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }//end class
}//end namespace

```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class Book: DataObject{
        public Book(string connString):base(connString){}

        public DataSet GetBooks(){
            return RunProcedure("BK_GetBooks",
                new SqlParameter[0], "Books");
        }//end getBooks

        public bool Add(int matID, int pubID, int cityID, string isbn){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@ISBN", SqlDbType.VarChar, 20),
new SqlParameter("@PubID", SqlDbType.Int, 4),
new SqlParameter("@CityID", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            parameters[1].Value = isbn;
            parameters[2].Value = pubID;
            parameters[3].Value = cityID;

            RunProcedure("BK_InsertBook",
                parameters, out rowsAffected);
            return (rowsAffected==1);
        }//end add

        public bool Update(int matID, int pubID, int cityID, string isbn){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@ISBN", SqlDbType.VarChar, 20),
new SqlParameter("@PubID", SqlDbType.Int, 4),
new SqlParameter("@CityID", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            parameters[1].Value = isbn;
            parameters[2].Value = pubID;
            parameters[3].Value = cityID;
            RunProcedure("BK_UpdateBook",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int matID){
            int numAffected;
            SqlParameter[] parameters =
            {new SqlParameter("@MatID", SqlDbType.Int, 4)};
            parameters[0].Value = matID;

            RunProcedure("BK_DeleteBook",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }//end class
}//end namespace
```

```

using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class Category: DataObject{
        public Category(string connString):base(connString){}

        public DataSet GetCategories(){
            return RunProcedure("BK_GetCategories",
                new SqlParameter[0], "Categories");
        } //end getcities

        public string GetCategory(int catID){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@CatName", SqlDbType.VarChar, 75),
new SqlParameter("@CatID", SqlDbType.Int, 4)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = catID;
            RunProcedure("BK_GetCategory",
                parameters, out rowsAffected);
            return (parameters[0].Value).ToString();
        }

        public int Add(string name, string description){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@CatID", SqlDbType.Int, 4),
new SqlParameter("@CatName", SqlDbType.VarChar, 75),
new SqlParameter("@CatDesc", SqlDbType.VarChar, 265)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = name;
            parameters[2].Value = description;

            RunProcedure("BK_InsertCategory",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        } //end add

        public bool Update(int catID, string name, string description){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@CatID", SqlDbType.Int, 4),
new SqlParameter("@CatName", SqlDbType.VarChar, 75),
new SqlParameter("@CatDesc", SqlDbType.VarChar, 265)
            };
            parameters[0].Value = catID;
            parameters[1].Value = name;
            parameters[2].Value = description;
            RunProcedure("BK_UpdateCategory",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        } //end Update

        public bool Delete(int catID){
            int numAffected;
            SqlParameter[] parameters =
{new SqlParameter("@CatID", SqlDbType.Int, 4)};
            parameters[0].Value = catID;

            RunProcedure("BK_DeleteCategory",
                parameters, out numAffected);
            return (numAffected == 1);
        }
    }
}

```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class City: DataObject{
        public City(string connString):base(connString){}

        public DataSet GetCities(){
            return RunProcedure("BK_GetCities",
                new SqlParameter[0], "Cities");
        }//end getcities

        public int Add(string name){
            int rowsAffected;
            SqlParameter[] parameters = {
                new SqlParameter("@CityID", SqlDbType.Int, 4),
                new SqlParameter("@City", SqlDbType.VarChar, 50)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = name;

            RunProcedure("BK_InsertCity",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        }//end add

        public bool Update(int cityID, string name){
            int rowsAffected;
            SqlParameter[] parameters ={
                new SqlParameter("@CityID", SqlDbType.Int, 4),
                new SqlParameter("@City", SqlDbType.VarChar, 50)
            };
            parameters[0].Value = cityID;
            parameters[1].Value = name;
            RunProcedure("BK_UpdateCity",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int cityID){
            int numAffected;
            SqlParameter[] parameters =
            {new SqlParameter("@CityID", SqlDbType.Int, 4)};
            parameters[0].Value = cityID;

            RunProcedure("BK_DeleteCity",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }//end class
}//end namespace
```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class Journal: DataObject{
        public Journal(string connString):base(connString){}

        public DataSet GetJournals(){
            return RunProcedure("BK_GetJournals",
                new SqlParameter[0], "Journals");
        }//end GetJournals

        public int Add(string name){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@JournalID", SqlDbType.Int, 4),
new SqlParameter("@JournalName", SqlDbType.VarChar, 100)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = name;

            RunProcedure("BK_InsertJournal",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        }//end add

        public bool Update(int journalID, string name){
            int rowsAffected;
            SqlParameter[] parameters =
            {
new SqlParameter("@JournalID", SqlDbType.Int, 4),
new SqlParameter("@JournalName", SqlDbType.VarChar, 100)
            };
            parameters[0].Value = journalID;
            parameters[1].Value = name;
            RunProcedure("BK_UpdateJournal",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int journalID){
            int numAffected;
            SqlParameter[] parameters =
            {new SqlParameter("@JournalID", SqlDbType.Int, 4)};
            parameters[0].Value = journalID;

            RunProcedure("BK_DeleteJournal",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }
}
```

```

using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data{
    public class Material: DataObject{
        public Material(string connString):base(connString){}

        public string GetTitle(int matID){
            int numAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@MatTitle", SqlDbType.VarChar, 250)
};
            parameters[0].Value = matID;
            parameters[1].Direction = ParameterDirection.Output;

            RunProcedure("BK_GetMaterialTitle",
                parameters, out numAffected);
            return parameters[1].Value.ToString();
        }

        public DataSet GetMaterials(){
            return RunProcedure("BK_GetMaterials",
                new SqlParameter[0], "Materials");
        } //end GetMaterials

        public DataSet GetMaterialByID(int matID){
            SqlParameter[] parameter1 = {
new SqlParameter("@MatID", SqlDbType.Int, 4)};
            SqlParameter[] parameter2 = {
new SqlParameter("@CategoryMatID", SqlDbType.Int, 4)};
            SqlParameter[] parameter3 = {
new SqlParameter("@AuthorMatID", SqlDbType.Int, 4)};
            parameter1[0].Value = matID;
            parameter2[0].Value = matID;
            parameter3[0].Value = matID;
            DataSet ds =
                RunProcedure("BK_GetMaterialByID",
                    parameter1, "Material");
            RunProcedure("BK_GetCategoriesPerMatID",
                parameter2, ds, "Categories");
            RunProcedure("BK_GetAuthorsPerMatID",
                parameter3, ds, "Authors");
            return ds;
        } //end GetMaterials

        public int Add(string title, string description, DateTime date){
            int rowsAffected;
            SqlParameter[] parameters = {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@MatTitle", SqlDbType.VarChar, 250),
new SqlParameter("@MatDatePub", SqlDbType.DateTime),
new SqlParameter("@MatDesc", SqlDbType.Text)
};
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = title;
            parameters[2].Value = date;
            parameters[3].Value = description;

            RunProcedure("BK_InsertMaterial",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        } //end add

        public bool Update(int matID, string title, string description, DateTime date)

```

```
{
    int rowsAffected;
    SqlParameter[] parameters =
    {
new SqlParameter("@MatID", SqlDbType.Int, 4),
new SqlParameter("@MatTitle", SqlDbType.VarChar, 250),
new SqlParameter("@MatDatePub", SqlDbType.DateTime),
new SqlParameter("@MatDesc", SqlDbType.Text)
    };
    parameters[0].Value = matID;
    parameters[1].Value = title;
    parameters[2].Value = date;
    parameters[3].Value = description;
    RunProcedure("BK_UpdateMaterial",
        parameters, out rowsAffected);
    return (rowsAffected == 1);
} //end Update

public bool Delete(int matID)
{
    int numAffected;
    SqlParameter[] parameters =
    {new SqlParameter("@MatID", SqlDbType.Int, 4)};
    parameters[0].Value = matID;

    RunProcedure("BK_DeleteMaterial",
        parameters, out numAffected);
    return (numAffected == 1);
} //end Delete
} //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data
{
    /// <summary>
    /// Summary description for MaterialAuthor.
    /// </summary>
    public class MaterialAuthor: DataObject
    {
        public MaterialAuthor(string connString):base(connString){}

        public DataSet GetAuthorsPerMatID(int matID)
        {
            SqlParameter[] parameters =
            {
                new SqlParameter("@AuthorMatID", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            return RunProcedure("BK_GetAuthorsPerMatID",
                parameters, "MaterialAuthors");
        } //end GetAuthorsPerMatID

        public DataSet GetMaterialAuthors()
        {
            return RunProcedure("BK_GetAuthorMaterials",
                new SqlParameter[0], "MaterialAuthors");
        } //end GetMaterialAuthors

        public int Add(int matID, int authID, int sequence)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@AuthMat", SqlDbType.Int, 4),
                new SqlParameter("@AuthID", SqlDbType.Int, 4),
                new SqlParameter("@MatID", SqlDbType.Int, 4),
                new SqlParameter("@Sequence", SqlDbType.Int, 4)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = authID;
            parameters[2].Value = matID;
            parameters[3].Value = sequence;

            RunProcedure("BK_InsertAuthorMaterial",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        } //end Add

        public bool Update(int authMat, int matID, int authID, int sequence)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@AuthMat", SqlDbType.Int, 4),
                new SqlParameter("@AuthID", SqlDbType.Int, 4),
                new SqlParameter("@MatID", SqlDbType.Int, 4),
                new SqlParameter("@Sequence", SqlDbType.Int, 4)
            };
            parameters[0].Value = authMat;
            parameters[1].Value = authID;
            parameters[2].Value = matID;
            parameters[3].Value = sequence;
            RunProcedure("BK_UpdateAuthorMaterial",
```



```
        parameters, out rowsAffected);
        return (rowsAffected == 1);
    } //end Update

    public bool Delete( int authMat )
    {
        int numAffected;
        SqlParameter[] parameters =
        {new SqlParameter("@AuthMat", SqlDbType.Int, 4)};
        parameters[0].Value = authMat;

        RunProcedure("BK_DeleteAuthorMaterial",
            parameters, out numAffected);
        return (numAffected == 1);
    } //end Delete
} //end class
} //end namespace
```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data
{
    /// <summary>
    /// Summary description for MaterialCategory.
    /// </summary>
    public class MaterialCategory: DataObject
    {
        public MaterialCategory(string connString):base(connString){}
        public DataSet GetMaterialCategories()
        {
            return RunProcedure("BK_GetMaterialCategories",
                new SqlParameter[0], "MaterialCategories");
        }//end GetMaterialCategories

        public DataSet GetCategoriesPerMatID(int matID)
        {
            SqlParameter[] parameters =
            {
                new SqlParameter("@MatID", SqlDbType.Int, 4)
            };
            parameters[0].Value = matID;
            return RunProcedure("BK_GetCategoriesPerMatID",
                parameters, "CategoriesPerMatID");
        }//end GetCategoriesPerMatID

        public DataSet GetMaterialsPerCatID(int catID)
        {
            SqlParameter[] parameters =
            {
                new SqlParameter("@CatID", SqlDbType.Int, 4)
            };
            parameters[0].Value = catID;
            return RunProcedure("BK_GetMaterialsPerCatID",
                parameters, "MaterialsPerCatID");
        }//end GetMaterialsPerCatID

        public int InsertMaterialCategory(int matID, int catID)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@MatCatID", SqlDbType.Int, 4),
                new SqlParameter("@MatID", SqlDbType.Int, 4),
                new SqlParameter("@CatID", SqlDbType.Int, 4)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = matID;
            parameters[2].Value = catID;

            RunProcedure("BK_InsertMaterialCategory",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        }//end InsertMaterialCategory

        public bool UpdateMaterialCategory(int matCatID, int matID, int catID)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@MatCatID", SqlDbType.Int, 4),
                new SqlParameter("@MatID", SqlDbType.Int, 4),
                new SqlParameter("@CatID", SqlDbType.Int, 4)
            }
        }
    }
}
```

```
    };
    parameters[0].Value = matCatID;
    parameters[1].Value = matID;
    parameters[2].Value = catID;

    RunProcedure("BK_UpdateMaterialCategory",
        parameters, out rowsAffected);
    return (rowsAffected == 1);
} //end UpdateMaterialCategory

public bool DeleteMaterialCategory(int matCatID)
{
    int numAffected;
    SqlParameter[] parameters =
    {new SqlParameter("@MatCatID", SqlDbType.Int, 4)};
    parameters[0].Value = matCatID;

    RunProcedure("BK_DeleteMaterialCategory",
        parameters, out numAffected);
    return (numAffected == 1);
} //end DeleteMaterialCategory
} //end class
} //end namespace
```

```

using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data
{
    /// <summary>
    /// Summary description for Publisher.
    /// </summary>
    public class Publisher: DataObject
    {
        public Publisher(string connString):base(connString){}

        public DataSet GetPublishers()
        {
            return RunProcedure("BK_GetPublishers",
                new SqlParameter[0], "Publishers");
        } //end GetPublishers

        public int Add(string name)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@PubID", SqlDbType.Int, 4),
                new SqlParameter("@PubName", SqlDbType.VarChar, 150)
            };
            parameters[0].Direction = ParameterDirection.Output;
            parameters[1].Value = name;

            RunProcedure("BK_InsertPublisher",
                parameters, out rowsAffected);
            return (int)(parameters[0].Value);
        } //end add

        public bool Update(int pubID, string name)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@PubID", SqlDbType.Int, 4),
                new SqlParameter("@PubName", SqlDbType.VarChar, 150)
            };
            parameters[0].Value = pubID;
            parameters[1].Value = name;
            RunProcedure("BK_UpdatePublisher",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        } //end Update

        public bool Delete(int pubID)
        {
            int numAffected;
            SqlParameter[] parameters =
            { new SqlParameter("@PubID", SqlDbType.Int, 4) };
            parameters[0].Value = pubID;

            RunProcedure("BK_DeletePublisher",
                parameters, out numAffected);
            return (numAffected == 1);
        } //end Delete
    }
}

```

```
using System;
using System.Data;
using System.Data.SqlClient;
using Merazzle.WebModules.Core;

namespace Merazzle.Books.Data
{
    /// <summary>
    /// Summary description for Url.
    /// </summary>
    public class Url: DataObject
    {
        public Url(string connString):base(connString){}

        public DataSet GetUrls()
        {
            return RunProcedure("BK_GetUrls",
                new SqlParameter[0], "Urls");
        }//end GetUrls

        public bool Add(int matID, string address)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@MatID", SqlDbType.Int, 4),
                new SqlParameter("@Url", SqlDbType.VarChar, 255)
            };
            parameters[0].Value = matID;
            parameters[1].Value = address;

            RunProcedure("BK_InsertUrl",
                parameters, out rowsAffected);
            return (rowsAffected==1);
        }//end add

        public bool Update(int urlID, string address)
        {
            int rowsAffected;
            SqlParameter[] parameters =
            {
                new SqlParameter("@UrlID", SqlDbType.Int, 4),
                new SqlParameter("@Url", SqlDbType.VarChar, 255)
            };
            parameters[0].Value = urlID;
            parameters[1].Value = address;
            RunProcedure("BK_UpdateUrl",
                parameters, out rowsAffected);
            return (rowsAffected == 1);
        }//end Update

        public bool Delete(int urlID)
        {
            int numAffected;
            SqlParameter[] parameters =
            {new SqlParameter("@UrlID", SqlDbType.Int, 4)};
            parameters[0].Value = urlID;

            RunProcedure("BK_DeleteUrl",
                parameters, out numAffected);
            return (numAffected == 1);
        }//end Delete
    }//end class
}//end namespace
```