

Lecture 07

Project

Wednesday, October 11, 2006

Phase 1

- Task 1: Schema design
- Task 2: Import sample data
- Task 3: Modify starter code

Task 1: Schema Design

Official requirement

- Read the project description
- Design a “good” database schema

Task 1: Schema Design

What you should do:

- Read description AND look inside the starter code `App_code/Provided/...`
- Read the classes, determine the fields...

Task 1: Schema Design

- Optional: draw an E/R diagram
- Create a file:

```
CREATE TABLE Customer ( ... )  
CREATE TABLE Invoice ( ... )  
...
```

- Create a second file:

```
DROP TABLE Customer  
DROP TABLE Invoice  
...
```

(why ?)

Task 1: Schema Design

Things to worry about:

- Keys/foreign keys: note table order matters!
- Make sure you represent all the data
- Null-able or not (don't worry too muchh)

Things not to worry about:

- fname or FirstName or PersonFirstName ?
- varchar(20) or char(200) or varchar(120) ?

Task 2: Import Sample Data

- Create a file:

```
INSERT INTO Customer ( ... )  
VALUES ('John', ....)  
INSERT INTO Customer ( ... )  
VALUES ('Sue', ....)  
...
```

- You may need to run this:

```
DROP TABLE Customer  
DROP TABLE Invoice  
...
```

(why ?)

Task 3: Modify Starter Code

The starter code:

- C#
- ASP.NET (you do not need to understand it)

It provides a Website for accessing your online store
BUT it misses the fragments of code that get the
data from the database

See

http://iisqlsrv.cs.washington.edu/CSE444/Phase1_Example/

C# - Crash Course

- Hello World
- Properties (getters/setters)
- Enums
- Partial classes
- Dataset: DataTable, DataRow
- Connecting to a database

<http://www.ecma-international.org/activities/Languages/Introduction%20to%20Csharp.pdf>

C# - Highlights

- C# = C++.Syntax + Java.Semantics
- It is a “safe” language (like Java)
- Can be embedded in Webpages
- Can access a database
 - Complex, but you should see the predecessors !

Hello World

```
using System;  
  
class Hello {  
    static void Main() {  
        Console.WriteLine("Hello world");  
    }  
}
```

Properties: Getters and Setters

```
public class Point {  
    private int x;  
    private string c;  
  
    public int position {  
        get { return x; }  
        set { x = value; c = "red"; }  
    }  
  
    public string color {  
        get { return c; }  
        set { c = value; x++; }  
    }  
}
```

```
Point uvw = new point();  
  
uvw.position = 55;  
uvw.color = "green";  
uvw.position =  
    uvw.position * 2;  
if (uvw.color == "green")  
    ...
```

Indexers

```
public class Stuff {  
    private int x[];  
  
    public int this[int i] {  
        get { x[2*i+1]=0; return x[2*i]; }  
        set { x[2*i] = value; x[2*i+1]=1; }  
    }  
}
```

```
Stuff uvw = new Stuff();
```

```
uvw[12] = 55;
```

```
uvw[99] = uvw[12]*7 + 2;
```

Enum

```
enum Color: byte {  
    Red = 1,  
    Green = 2,  
    Blue = 4,  
    Black = 0,  
    White = Red | Green | Blue,  
}
```

Partial Classes

- Some fields defined in file 1
- Other fields defined in file 2

- Why ?
Brian creates file 1, you create file 2

Dataset

This is an important class that allows you to interact with a database

Dataset = a “mini” database in main memory

- DataTable
- DataRow

DataSet

```
DataSet myLocalDB = new DataSet();  
  
.....  
..... /* create inside a table called “books” */  
..... /* (this is shown on a following slide) */  
  
/* now use “books” */  
DataTable x = myLocalDB.Tables[“books”]  
  
foreach (DataRow y in x.Rows) {  
    if (y[“title”] == “Harry Potter”) y[“price”]++;  
}
```

Connecting to a Database

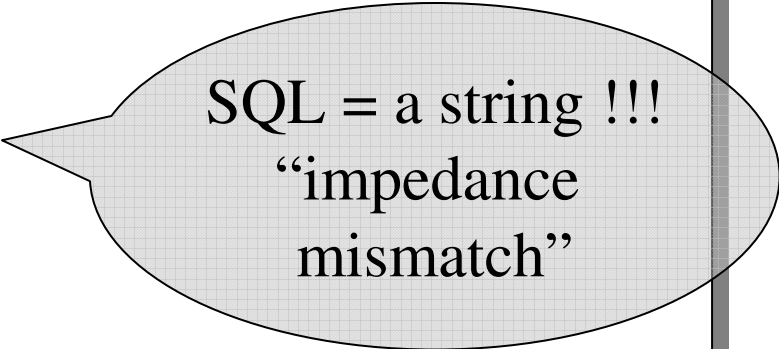
- Create or edit web.config file
 - Specify iisqlsrv, user, password
 - Give a 'name'
- Create a SqlConnection
 - refer to 'name'
- Create a SqlDataAdapter
 - embed SQL query string
- Execute the Fill() method to run query and store answers in a datarow

Connecting to a Database

```
/* create inside a table called "books" */
```

```
SqlConnection c = new SqlConnection( . . . "name" . . . );
```

```
string q = "select title, price year  
from products  
where price < 100";
```



SQL = a string !!!
"impedance
mismatch"

```
SqlDataAdapter a = new SqlDataAdapter(q, c);
```

```
DataSet myLocalDB = new DataSet();
```

```
a.Fill(myLocalDB, "books");
```

Task 3: Modify Starter Code

- What you have to do:
- App_Code/Phase1/Billing and Shipping/...

```
Public partial class Customer {  
    /* add your own fields, like: */  
    private int id,  
  
    Procedure List<invoice> GetInvoices() {  
        /* your GetInvoices code goes here */  
    }  
}
```

Task 3: Modify Starter Code

```
/* your GetInvoices code goes here */
```

```
string s = String.Format(  
    @“SELECT ...  
    FROM ....  
    WHERE x.customerId = {0} ...”, id);
```

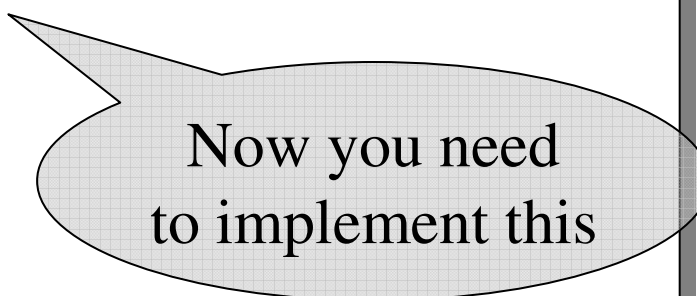
Substitutes
id for {0}

Defined in
Provided

```
StoreContext store = StoreContext.current;  
DataSet ds = new DataSet( );  
DataTable invoices = store.GetDataTable(s, ds, “invoices”);  
/* continued on next slide.... */
```

Task 3: Modify Starter Code

```
/* ... continued from previous slide */  
  
List<Invoice> invoiceList new List<Invoice> ( );  
  
foreach(datarow r in invoices.Rows) {  
    invoiceList.Add(new Invoice( r ));  
}  
  
return invoiceList;  
}
```



Now you need
to implement this

Task 3: Modify Starter Code

```
public partial class Invoice {  
    public Invoice(DataRow invoiceData) {  
        /* here goes your code, something like that: */  
        init(invoiceData); /* may need it in several places */  
    }  
    ....  
    private void init(DataRow invoiceData) {  
        invoiceId = (int) invoiceData["invoiceId"];  
        orderDate = (DateTime) invoiceData["date"];  
        ....  
    }  
}
```

In Provided

In you SQL

Time Estimate

- Task 1: about 9 tables or so, 2 hours or more
- Task 2: try 2 tuples per table, 1 hour
- Task 3: here's when you'll discover that tasks 1 and 2 are bad: DROP TABLEs then go to Task 1 (should be faster 2nd time). Total time here: ???

Brian will be in the lab for questions: WHEN ?