Program.cs

using System;

using System.Collections.Generic;

using System.Data.Odbc;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

/\*

\* SAI430 Week 7 Lab

\* Nathan Stawhand

\* Reports Class

\*/

namespace SAI430\_Wk7\_NStrawhand

{

class Program

{

static void Main(string[] args)

{

//Connect to database

//Connection string needed to talk to MySQL on local machine

string conString = "Driver={MySQL ODBC 5.3 ANSI Driver};"

+ "Server=localhost;Port=3306;"

+ "Database=sai430\_db;"

+ "uid=root;pwd=";

OdbcConnection connection = new OdbcConnection(conString);

connection.Open();

//Open input file

//Set where the file comes from

string filepath = @"C:\Users\Nathan\Desktop\Pictures\School CLasses and Things\SAI 430\Labs\Lab Files For DB\";

string filename = @"TestUpdate.xml";

//Open XML reader – name it “theFile”

XmlReader theFile = XmlReader.Create(filepath + filename);

//Loop through file and add to database. Read will return FALSE when there are no more lines to read.

while (theFile.Read())

{

//Create an object to use each time through the loop

Item theItem = new Item();

//Check each node in the XML file to see what it is: ADD, UPDATE, or DELETE

if (theFile.Name.Equals("ADD"))

{

theItem.XMLAdd(theFile, connection);

}

else if (theFile.Name.Equals("UPDATE"))

{

theItem.XMLUpdate(theFile, connection);

}

else if (theFile.Name.Equals("DELETE"))

{

theItem.XMLDelete(theFile, connection);

}

} //end of while loop

Reports.AllInventory(connection, filepath);

Reports.ErrorLog(connection, filepath);

Reports.Reorder(connection, filepath);

connection.Close();

theFile.Close();

Console.WriteLine("\n\n\nPress ENTER to continue");

Console.ReadLine();

}

}

}

Error.cs

using System.Collections.Generic;

using System.Data.Odbc;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*

\* SAI430 Week 7 Lab

\* Nathan Stawhand

\* Reports Class

\*/

namespace SAI430\_Wk7\_NStrawhand

{

public static class Error

{

public static bool AddErrorLog(OdbcConnection db, string ErrorMsg, int itemId)

{

String sql = "INSERT INTO errorLog "

+ "(item\_id, errorTime, errorMsg) "

+ "VALUES( ?, ?, ?)";

OdbcCommand Command = new OdbcCommand(sql, db);

Command.Parameters.Add("@ID", OdbcType.Int).Value = itemId;

Command.Parameters.Add("@ErrorTime", OdbcType.DateTime).Value = DateTime.Now;

Command.Parameters.Add("@ErrorMsg", OdbcType.VarChar).Value = ErrorMsg;

//Returns 1 if successful

int result = Command.ExecuteNonQuery();

if (result > 0)

//Was successful in adding

return true;

else

//failed to add

return false;

} //end of AddRow

}

}

Reports.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Odbc;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*

\* SAI430 Week 7 Lab

\* Nathan Stawhand

\* Reports Class

\*/

namespace SAI430\_Wk7\_NStrawhand

{

public static class Reports

{

public static void AllInventory(OdbcConnection db, string filepath)

{

//Get all data

string theQuery = "SELECT \* "

+ "FROM item, inventory "

+ "WHERE item.invent\_id = inventory.invent\_id";

OdbcDataAdapter DataAdapter = new OdbcDataAdapter(theQuery, db);

DataSet theData = new DataSet();

DataAdapter.Fill(theData);

DataTable theTable = theData.Tables[0];

//Set the report filename

string filename = @"AllItemsTest.html";

//Check to see if directory exists, if not create it.

if (!Directory.Exists(filepath))

Directory.CreateDirectory(filepath);

//Open file for output

TextWriter webPage = new StreamWriter(filepath + filename, false);

//Page header

webPage.WriteLine("<html>");

webPage.WriteLine("<head>");

webPage.WriteLine("<link rel='stylesheet' type='text/css' href='Stylesheet.css'>");

webPage.WriteLine("<title>Inventory Report</title");

webPage.WriteLine("</head>");

//Page body

webPage.WriteLine("<body>");

//Start an output table

webPage.WriteLine("<h1>Inventory Listing</h1>");

webPage.WriteLine("<table border=1>");

//header row of table

webPage.WriteLine("<tr>");

webPage.WriteLine("<th>ITEM ID</th>");

webPage.WriteLine("<th>DESCRIPTION</th>");

webPage.WriteLine("<th>QUANT</th>");

webPage.WriteLine("<th>SIZE</th>");

webPage.WriteLine("<th>COLOR</th>");

webPage.WriteLine("<th>PRICE</th>");

//end header row

webPage.WriteLine("</tr>");

//Loop through all data results

foreach (DataRow dataRow in theTable.Rows)

{

webPage.WriteLine("<tr>");

webPage.WriteLine("<td>{0}</td>", dataRow["item\_ID"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["invent\_desc"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["qoh"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["itemsize"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["color"].ToString());

//Use the "C" specifier to format currency.

webPage.WriteLine("<td>{0}</td>",

((decimal)dataRow["curr\_price"]).ToString("C"));

webPage.WriteLine("</tr>");

}

//end table

webPage.WriteLine("</table>");

webPage.WriteLine("</body>");

webPage.WriteLine("</html>");

//Make sure all characters are in the file.

webPage.Flush();

//Closes the file and officially writes it to disk.

webPage.Close();

}

public static void ErrorLog(OdbcConnection db, string filepath)

{

//Get all data

string theQuery = "SELECT \* "

+ "FROM errorlog ";

OdbcDataAdapter DataAdapter = new OdbcDataAdapter(theQuery, db);

DataSet theData = new DataSet();

DataAdapter.Fill(theData);

DataTable theTable = theData.Tables[0];

//Set the report filename

string filename = @"ErrorLogTest.html";

//Check to see if directory exists, if not create it.

if (!Directory.Exists(filepath))

Directory.CreateDirectory(filepath);

//Open file for output

TextWriter webPage = new StreamWriter(filepath + filename, false);

//Page header

webPage.WriteLine("<html>");

webPage.WriteLine("<head>");

webPage.WriteLine("<link rel='stylesheet' type='text/css' href='Stylesheet.css'>");

webPage.WriteLine("<title>Error Log</title");

webPage.WriteLine("</head>");

//Page body

webPage.WriteLine("<body>");

//Start an output table

webPage.WriteLine("<h1>Error Log</h1>");

webPage.WriteLine("<table border=1>");

//header row of table

webPage.WriteLine("<tr>");

webPage.WriteLine("<th>ERROR ID</th>");

webPage.WriteLine("<th>ITEM ID</th>");

webPage.WriteLine("<th>DATE/TIME</th>");

webPage.WriteLine("<th>DESCRIPTION</th>");

//end header row

webPage.WriteLine("</tr>");

//Loop through all data results

foreach (DataRow dataRow in theTable.Rows)

{

webPage.WriteLine("<tr>");

webPage.WriteLine("<td>{0}</td>", dataRow["error\_id"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["item\_id"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["errorTime"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["errorMsg"].ToString());

webPage.WriteLine("</tr>");

}

//end table

webPage.WriteLine("</table>");

webPage.WriteLine("</body>");

webPage.WriteLine("</html>");

//Make sure all characters are in the file.

webPage.Flush();

//Closes the file and officially writes it to disk.

webPage.Close();

}

public static void Reorder(OdbcConnection db, string filepath)

{

//Get all data

string theQuery = "SELECT \* "

+ "FROM item, inventory "

+ "WHERE item.invent\_id = inventory.invent\_id and qoh <= 100";

OdbcDataAdapter DataAdapter = new OdbcDataAdapter(theQuery, db);

DataSet theData = new DataSet();

DataAdapter.Fill(theData);

DataTable theTable = theData.Tables[0];

//Set the report filename

string filename = @"ReOrderTest.html";

//Check to see if directory exists, if not create it.

if (!Directory.Exists(filepath))

Directory.CreateDirectory(filepath);

//Open file for output

TextWriter webPage = new StreamWriter(filepath + filename, false);

//Page header

webPage.WriteLine("<html>");

webPage.WriteLine("<head>");

webPage.WriteLine("<link rel='stylesheet' type='text/css' href='Stylesheet.css'>");

webPage.WriteLine("<title>Inventory Reorder Report</title");

webPage.WriteLine("</head>");

//Page body

webPage.WriteLine("<body>");

//Start an output table

webPage.WriteLine("<h1>Inventory Reorder Report</h1>");

webPage.WriteLine("<table border=1>");

//header row of table

webPage.WriteLine("<tr>");

webPage.WriteLine("<th>ITEM ID</th>");

webPage.WriteLine("<th>DESCRIPTION</th>");

webPage.WriteLine("<th>QUANT</th>");

webPage.WriteLine("<th>SIZE</th>");

webPage.WriteLine("<th>COLOR</th>");

webPage.WriteLine("<th>PRICE</th>");

//end header row

webPage.WriteLine("</tr>");

//Loop through all data results

foreach (DataRow dataRow in theTable.Rows)

{

webPage.WriteLine("<tr>");

webPage.WriteLine("<td>{0}</td>", dataRow["item\_ID"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["invent\_desc"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["qoh"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["itemsize"].ToString());

webPage.WriteLine("<td>{0}</td>", dataRow["color"].ToString());

//Use the "C" specifier to format price to currency.

webPage.WriteLine("<td>{0}</td>",

((decimal)dataRow["curr\_price"]).ToString("C"));

webPage.WriteLine("</tr>");

}

//end table

webPage.WriteLine("</table>");

webPage.WriteLine("</body>");

webPage.WriteLine("</html>");

//Make sure all characters are in the file.

webPage.Flush();

//Closes the file and officially writes it to disk.

webPage.Close();

}

}

}

Items.cs

using System;

using System.Collections.Generic;

using System.Data.Odbc;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

/\*

\* SAI430 Week 7 Lab

\* Nathan Stawhand

\* Reports Class

\*/

namespace SAI430\_Wk7\_NStrawhand

{

class Item

{

public Item()

{

}

//Set and Get properties

public int Item\_ID { get; set; }

public int Invent\_id { get; set; }

public string Itemsize { get; set; }

public string Color { get; set; }

public decimal Curr\_price { get; set; }

public int Qoh { get; set; }

//Parse CSV line

public bool ParseCSVline(string aLine)

{

try

{

string[] fields = aLine.Split(',');

this.Item\_ID = int.Parse(fields[0]);

this.Invent\_id = int.Parse(fields[1]);

this.Itemsize = fields[2];

this.Color = fields[3];

this.Curr\_price = decimal.Parse(fields[4]);

this.Qoh = int.Parse(fields[5]);

return true; //if everything parsed, return true

}

catch (Exception ex)

{

return false; //if a parse failed, return false

}

} //end of barseCSVline

//See if an item is in the database.

//Pass in the database connection as db.

public bool IsInDatabase(OdbcConnection db)

{

String sql = "SELECT \* FROM item WHERE item\_ID=?";

OdbcCommand Command = new OdbcCommand(sql, db);

Command.Parameters.Add("@ID", OdbcType.Int).Value = this.Item\_ID;

//If query returns a row - there is already an item in the table.

if (Command.ExecuteReader().HasRows)

return true;

else

return false;

} //end of IsInDatabase

//Add a row to the database passed in as db

public bool AddRow(OdbcConnection db)

{

String sql = "INSERT INTO item "

+ "(item\_id, invent\_id, itemsize, color, curr\_price, qoh) "

+ "VALUES( ?, ?, ?, ?, ?, ?)";

OdbcCommand Command = new OdbcCommand(sql, db);

Command.Parameters.Add("@ID", OdbcType.Int).Value = this.Item\_ID;

Command.Parameters.Add("@INVID", OdbcType.Int).Value = this.Invent\_id;

Command.Parameters.Add("@SZ", OdbcType.VarChar).Value = this.Itemsize.Trim();

Command.Parameters.Add("@COL", OdbcType.VarChar).Value = this.Color.Trim();

Command.Parameters.Add("@PR", OdbcType.Double).Value = (double)this.Curr\_price;

Command.Parameters.Add("@QOH", OdbcType.Int).Value = this.Qoh;

int result = Command.ExecuteNonQuery(); //Returns 1 if successful

if (result > 0)

return true; //Was successful in adding

else

return false; //failed to add

} //end of AddRow

//Update a row to the database passed in as db

public bool UpdateRow(OdbcConnection db)

{

String sql = "UPDATE item "

+ "SET itemsize=?, "

+ "color=?, "

+ "curr\_price=?, "

+ "qoh=? "

+ "WHERE item\_id=?";

OdbcCommand Command = new OdbcCommand(sql, db);

Command.Parameters.Add("@SZ", OdbcType.VarChar).Value = this.Itemsize.Trim();

Command.Parameters.Add("@COL", OdbcType.VarChar).Value = this.Color.Trim();

Command.Parameters.Add("@PR", OdbcType.Double).Value = (double)this.Curr\_price;

Command.Parameters.Add("@QOH", OdbcType.Int).Value = this.Qoh;

Command.Parameters.Add("@ID", OdbcType.Int).Value = this.Item\_ID;

int result = Command.ExecuteNonQuery(); //Returns 1 if successful

if (result > 0)

return true; //Was successful in updating

else

return false; //failed to update

}

//Delete a row from the database passed in as db

public bool DeleteRow(OdbcConnection db)

{

String sql = "DELETE FROM item WHERE item\_id=?";

OdbcCommand Command = new OdbcCommand(sql, db);

Command.Parameters.Add("@ID", OdbcType.Int).Value = this.Item\_ID;

int result = Command.ExecuteNonQuery(); //Returns 1 if successful

if (result > 0)

return true; //Was successful in deleting

else

return false; //failed to delete

}

//Used to parse an XML file pointed to by XmlReader

public bool parseXML(XmlReader f)

{

try

{

this.Item\_ID = int.Parse(f.GetAttribute("item\_id"));

this.Invent\_id = int.Parse(f.GetAttribute("invent\_id"));

this.Itemsize = f.GetAttribute("itemsize");

this.Color = f.GetAttribute("color");

this.Curr\_price = decimal.Parse(f.GetAttribute("curr\_price"));

this.Qoh = int.Parse(f.GetAttribute("qoh"));

}

catch (Exception ex)

{

return false;

}

return true;

}

//Used to parse an XML file pointed to by XmlReader

public bool parseXMLForId(XmlReader f)

{

try

{

this.Item\_ID = int.Parse(f.GetAttribute("item\_id"));

}

catch (Exception ex)

{

return false;

}

return true;

}

//Get this item from the XML file and

//add this item to the database passed in as db

public bool XMLAdd(XmlReader f, OdbcConnection db)

{

if (!this.parseXML(f)) //parse the item from "f"

{

Error.AddErrorLog(db, "Could not parse the XML data from the file",this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not ADDED: An input value was an improper datatype", this.Item\_ID));

return false; //Leave if the parse failed

}

if (this.Item\_ID < 1) {

Error.AddErrorLog(db, "Item id was less than 1.",this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Added: Item id was less than 1.", this.Item\_ID));

return false;

}

if (this.Curr\_price < 0)

{

Error.AddErrorLog(db, "Current price was negative.",this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Added: Current price was negative.", this.Item\_ID));

return false;

}

if (this.Qoh < 0)

{

Error.AddErrorLog(db, "Quantity on hand was negative.",this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Added: Quantity on hand was negative.", this.Item\_ID));

return false;

}

//Is it in database? Check that it is NOT.

if (!this.IsInDatabase(db))

{

//if not, add it

if (this.AddRow(db))

return true;

else

{

Error.AddErrorLog(db, "Tried to send an SQL command, but it failed for some reason.",this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not ADDED: It failed for some reason", this.Item\_ID));

return false; //if something went wrong

}

}

else

{

Error.AddErrorLog(db, "Tried to do an ADD, but the item already was in the database.", this.Item\_ID);

Console.WriteLine(String.Format("{0} already in the database, can't insert.", this.Item\_ID));

return false; //already in DB

}

}

//Get this item from the XML file and

//update this item to the database passed in as db

public bool XMLUpdate(XmlReader f, OdbcConnection db)

{

if (!this.parseXML(f)) //parse the item from "f"

{

Error.AddErrorLog(db, "Could not parse the XML data from the file.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not UPDATED: Failed to update item", this.Item\_ID));

return false; //Leave if the parse failed

}

if (this.Item\_ID < 1)

{

Error.AddErrorLog(db, "Item id was less than 1.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Updated: Item id was less than 1.", this.Item\_ID));

return false;

}

if (this.Curr\_price < 0)

{

Error.AddErrorLog(db, "Current price was negative.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Updated: Current price was negative.", this.Item\_ID));

return false;

}

if (this.Qoh < 0)

{

Error.AddErrorLog(db, "Quantity on hand was negative.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Updated: Quantity on hand was negative.", this.Item\_ID));

return false;

}

//Is it in database? Check that it is NOT.

if (this.IsInDatabase(db))

{

//if not, add it

if (this.UpdateRow(db))

return true;

else

{

Error.AddErrorLog(db, "Tried to send an SQL command, but it failed for some reason.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not UPDATED: Failed to update item", this.Item\_ID));

return false; //if something went wrong

}

}

else

{

Error.AddErrorLog(db, "Tried to do an ADD, but the item already was in the database.", this.Item\_ID);

Console.WriteLine(String.Format("{0} already in the database, can't insert.", this.Item\_ID));

return false; //already in DB

}

}

//Get this item from the XML file and

//delete this item from the database passed in as db

public bool XMLDelete(XmlReader f, OdbcConnection db)

{

this.Item\_ID = int.Parse(f.GetAttribute("item\_id"));

if (!this.parseXMLForId(f)) //parse the item from "f"

{

Error.AddErrorLog(db, "Could not parse the XML data from the file.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not deleted: Failed to parse data", this.Item\_ID));

return false; //Leave if the parse failed

}

//Is it in database? Check that it is NOT.

if (this.IsInDatabase(db))

{

//if not, add it

if (this.DeleteRow(db))

return true;

else

{

Error.AddErrorLog(db, "Tried to send an SQL command, but it failed for some reason.", this.Item\_ID);

Console.WriteLine(String.Format("Item {0} not Deleted: Failed to delete item", this.Item\_ID));

return false; //if something went wrong

}

}

else

{

Error.AddErrorLog(db, "Tried to do a DELETE, but the item wasn’t there.", this.Item\_ID);

Console.WriteLine(String.Format("{0} not in database, can't delete.", this.Item\_ID));

return false; //already in DB

}

}

}

}