N – File I/O

Code Samples - Documentation

# Examples

File Input/Output (IO) is all about using files for storing and retrieving data. These files exist “outside” of your program, and code is required to transfer the data between the file and the objects/variables in your program.

Data inside of files can be stored in either a binary or a text format. All of these examples will deal with text files. When using text files, the text itself needs to follow some pattern in order to distinguish the individual pieces of data from each other.

An old and traditional way of storing the text is to use a **CSV** (**C**omma-**S**eparated **V**alues) format. In this format, each row of information (also known as a “record” of data) is broken down into individual pieces of data by separating each piece (or “field”) by a comma. Rows are separated by a new line (or “carriage return”) marker. This format has had the advantage of being easy for humans to read and edit directly; it is also fairly flexible and programmers can easily adapt the form to fit the shape of the data they wish to store.

A more modern approach is to store the textual information in **XML** (e**X**tensible **M**arkup **L**anguage) format. While also being human-readable and easy to format, XML has the advantage of making it easier for computer programs to read and save data that is both relational and complex. XML format uses “tags” (or “mark-up”) to wrap data. For example, if information on a person (“Bob Smith”) is stored in an XML file, it might look like this.

<Person>

<FirstName>Bob</FirstName>

<LastName>Smith</LastName>

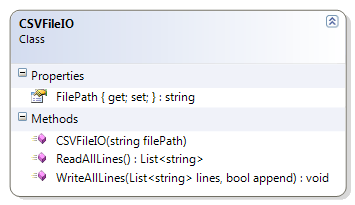
</Person>

File Input/Output is best performed by having one class (or object) deal with formatting the data and a separate class to deal with directly accessing the file. In the following examples, the CSVFileIO and XMLFileIO classes read and write directly to the files, while the other classes (which end with “Adapter”) are concerned with determining the format of the data.

The following examples are used to illustrate this topic.

* **PhoneNumberFileAdapter** – This reads a text file to fill a list of PhoneNumber data.
* **StudentFileAdapter** – This reads a text file to fill a list of Student data.
* **BookFileAdapter** – This reads a text file to fill a list of Book data. Note that in this example, not every field in the record structure is used in creating the required Book objects; unused fields are simply read and “discarded” by the BookFileAdapter. In addition, the Book class and text file used in this example allows for a book to have multiple authors.
* **SafetyDepositBoxFileAdapter** – This reads a text file to fill a list of SafetyDepositBox data. Note that in this particular example, some records may not have an account number for the box number.

# CSVFileIO



/// <summary>

/// The CSVFileStorage class provides simple reading of all the lines of text of a CSV (Comma-Separated-Values) file.

/// </summary>

public class CSVFileIO

{

public string FilePath { get; private set; }

/// <summary>

/// Initializes a new instance of the CSVFileStorage class.

/// </summary>

public CSVFileIO(string filePath)

{

FilePath = filePath;

}

public List<string> ReadAllLines()

{

List<string> lines = new List<string>();

using (TextReader reader = new StreamReader(FilePath))

{

string aSingleLine = reader.ReadLine();

while (aSingleLine != null)

{

lines.Add(aSingleLine);

aSingleLine = reader.ReadLine();

}

}

return lines;

}

public void WriteAllLines(List<string> lines, bool append)

{

using (TextWriter writer = new StreamWriter(FilePath, append))

{

foreach (string line in lines)

{

writer.WriteLine(line);

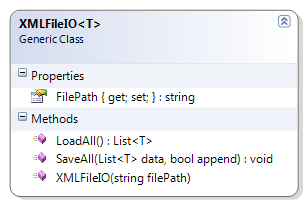
}

}

}

}

# XMLFileIO



/// <summary>

/// The XMLFileIO class manages loading and saving any type to an XML file.

/// </summary>

public class XMLFileIO<T>

{

public string FilePath { get; private set; }

public XMLFileIO(string filePath)

{

FilePath = filePath;

}

public List<T> LoadAll()

{

XmlSerializer deserializer = new XmlSerializer(typeof(List<T>));

List<T> data;

using (TextReader textReader = new StreamReader(FilePath))

{

data = (List<T>)deserializer.Deserialize(textReader);

textReader.Close();

}

return data;

}

public void SaveAll(List<T> data, bool append)

{

if (append && File.Exists(FilePath))

{

List<T> existingData = LoadAll();

existingData.AddRange(data);

data = existingData;

}

using (Stream stream = File.Open(FilePath, FileMode.OpenOrCreate))

{

XmlSerializer serializer = new XmlSerializer(typeof(List<T>));

using (TextWriter writer = new StreamWriter(stream))

{

serializer.Serialize(writer, data);

writer.Close();

}

}

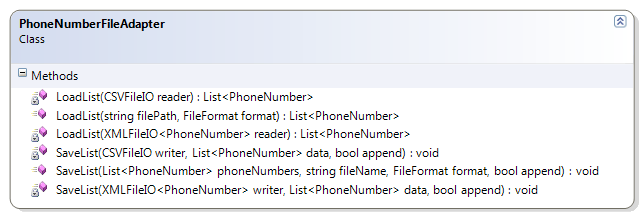
}

}

# PhoneNumberFileAdapter

This reads a text file to fill a list of PhoneNumber data.

* Record Structure:
  1. Telephone Number : String
  2. Last Name : String
  3. First Name : String



## Code Solution

public class PhoneNumberFileAdapter

{

#region Read from a file

public static List<PhoneNumber> LoadList(string filePath, FileFormat format)

{

List<PhoneNumber> data;

if (format == FileFormat.CSV)

data = LoadList(new CSVFileIO(filePath));

else

data = LoadList(new XMLFileIO<PhoneNumber>(filePath));

return data;

}

private static List<PhoneNumber> LoadList(CSVFileIO reader)

{

List<PhoneNumber> data = new List<PhoneNumber>();

List<string> lines = reader.ReadAllLines();

foreach (string individualLine in lines)

{

// code specifics here..

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

string firstName = fields[0], lastName = fields[1], number = fields[2];

data.Add(new PhoneNumber(firstName, lastName, number));

}

return data;

}

private static List<PhoneNumber> LoadList(XMLFileIO<PhoneNumber> reader)

{

return reader.LoadAll();

}

#endregion

#region Write to a file

public static void SaveList(List<PhoneNumber> data, string fileName, FileFormat format, bool append)

{

if (format == FileFormat.CSV)

SaveList(new CSVFileIO(fileName), data, append);

else

SaveList(new XMLFileIO<PhoneNumber>(fileName), data, append);

}

private static void SaveList(CSVFileIO writer, List<PhoneNumber> data, bool append)

{

List<string> lines = new List<string>();

foreach (PhoneNumber item in data)

{

lines.Add(item.FirstName + "," + item.LastName + "," + item.Number);

}

writer.WriteAllLines(lines, append);

}

private static void SaveList(XMLFileIO<PhoneNumber> writer, List<PhoneNumber> data, bool append)

{

writer.SaveAll(data, append);

}

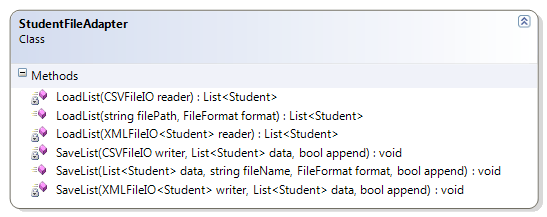
#endregion

}

# StudentFileAdapter

This reads a text file to fill a list of Student data.

* Record Structure:
  1. Student Id : Integer
  2. Name : String
  3. Gender : GenderType



## Code Solution

public class StudentFileAdapter

{

#region Read from a file

public static List<Student> LoadList(string filePath, FileFormat format)

{

List<Student> data;

if (format == FileFormat.CSV)

data = LoadList(new CSVFileIO(filePath));

else

data = LoadList(new XMLFileIO<Student>(filePath));

return data;

}

private static List<Student> LoadList(CSVFileIO reader)

{

List<Student> data = new List<Student>();

List<string> lines = reader.ReadAllLines();

foreach (string individualLine in lines)

{

// code specifics here..

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

int id = System.Convert.ToInt32(fields[0]);

string name = fields[1];

GenderType gender = (GenderType)System.Enum.Parse(typeof(GenderType), fields[2]);

data.Add(new Student(name, gender, id));

}

return data;

}

private static List<Student> LoadList(XMLFileIO<Student> reader)

{

return reader.LoadAll();

}

#endregion

#region Write to a file

public static void SaveList(List<Student> data, string fileName, FileFormat format, bool append)

{

if (format == FileFormat.CSV)

SaveList(new CSVFileIO(fileName), data, append);

else

SaveList(new XMLFileIO<Student>(fileName), data, append);

}

private static void SaveList(CSVFileIO writer, List<Student> data, bool append)

{

List<string> lines = new List<string>();

foreach (Student item in data)

{

lines.Add(item.StudentId.ToString() + "," + item.Name + "," + item.Gender.ToString());

}

writer.WriteAllLines(lines, append);

}

private static void SaveList(XMLFileIO<Student> writer, List<Student> data, bool append)

{

writer.SaveAll(data, append);

}

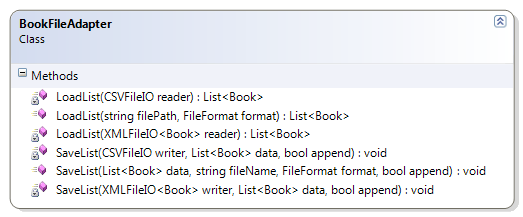
#endregion

}

# BookFileAdapter

This reads a CSV text file to fill a list of Book data. Note that in this example, not every field in the record structure of CSV files is used in creating the required Book objects; unused fields are simply read and “discarded” by the BookFileAdapter. In addition, the Book class and text file used in this example allows for a book to have multiple authors.

* Record Structure in the file:
  1. Title : String
  2. Authors : String – delimited by semi-colons
  3. Publication year : Integer
  4. Publisher : String
  5. Format : String
  6. Dewey Decimal Number : String – could be empty in the file
  7. ISBN : String
  8. Number of Pages : Integer
  9. Cover Price : Money



## Code Solution

/// <summary>

/// BookFileAdapter provides a way to read and write

/// Book information to and from text files.

/// </summary>

/// <remarks>

/// This class requires that the file structure be in the following format:

/// <ol><li>Title : String

/// <li>Authors : String[]

/// <ol><li>The Author information is a semicolon (;) separated list of authors

/// </li></ol>

/// <li>Publication Year : Integer

/// <li>Publisher : String

/// <li>Format : String

/// <li>Dewey Decimal Number : String

/// <li>ISBN : String

/// <li>Number of Pages : Integer

/// <li>Cover Price : Money

/// </ol>

/// <para>

/// This method fills a List of Book data. Each Book only needs

/// the title, the authors (as an array of Strings), and the ISBN.

/// Therefore, the other data in the file will simply be discarded as it

/// is read. In the file, the Author information is a semicolon (;)

/// separated list of authors.

/// </para>

/// </remarks>

public class BookFileAdapter

{

#region Read from a file

public static List<Book> LoadList(string filePath, FileFormat format)

{

List<Book> data;

if (format == FileFormat.CSV)

data = LoadList(new CSVFileIO(filePath));

else

data = LoadList(new XMLFileIO<Book>(filePath));

return data;

}

private static List<Book> LoadList(CSVFileIO reader)

{

List<Book> data = new List<Book>();

List<string> lines = reader.ReadAllLines();

foreach (string individualLine in lines)

{

string title, authorList;

ISBN bookUPC;

// Parse the record

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

title = fields[0];

authorList = fields[1];

// fields[2] Publication Year

// fields[3] Publisher

// fields[4] Format

// fields[5] Dewey Decimal Number

bookUPC = new ISBN(fields[6]);

// fields[7] Number of Pages

// fields[8] Cover Price

// Parse the authorList

string[] authors = null;

authors = authorList.Split(';');

// Create the book and add it to the array

data.Add(new Book(title, authors, bookUPC));

}

return data;

}

private static List<Book> LoadList(XMLFileIO<Book> reader)

{

return reader.LoadAll();

}

#endregion

#region Write to a file

public static void SaveList(List<Book> data, string fileName, FileFormat format, bool append)

{

if (format == FileFormat.CSV)

SaveList(new CSVFileIO(fileName), data, append);

else

SaveList(new XMLFileIO<Book>(fileName), data, append);

}

private static void SaveList(CSVFileIO writer, List<Book> data, bool append)

{

List<string> lines = new List<string>();

foreach (Book item in data)

{

string formattedRecord;

formattedRecord = item.Title;

formattedRecord += string.Join(";", item.Authors);

formattedRecord += item.Isbn.BarCode;

lines.Add(formattedRecord);

}

writer.WriteAllLines(lines, append);

}

private static void SaveList(XMLFileIO<Book> writer, List<Book> data, bool append)

{

writer.SaveAll(data, append);

}

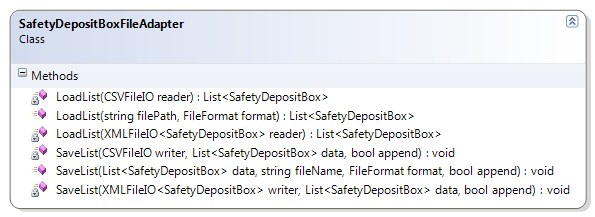
#endregion

}

# SafetyDepositBoxFileAdapter

This reads a text file to fill a list of SafetyDepositBox data. Note that in this particular example, some records may not have an account number for the box number.

* Record Structure:
  1. Box Number : Integer
  2. Account Number : Integer – may not exist



## Code Solution

/// <summary>

/// SafetyDepositBoxFileAdapter provides a way to read and write

/// SafetyDepositBox information to and from text files.

/// </summary>

/// <remarks>

/// This class requires that the file structure be in the following format:

/// <ol><li>StudentId : Integer

/// <li>StudentName : String

/// <li>StudentGender : GenderType

/// </ol>

/// </remarks>

public class SafetyDepositBoxFileAdapter

{

#region Read from a file

public static List<SafetyDepositBox> LoadList(string filePath, FileFormat format)

{

List<SafetyDepositBox> data;

if (format == FileFormat.CSV)

data = LoadList(new CSVFileIO(filePath));

else

data = LoadList(new XMLFileIO<SafetyDepositBox>(filePath));

return data;

}

private static List<SafetyDepositBox> LoadList(CSVFileIO reader)

{

List<SafetyDepositBox> data = new List<SafetyDepositBox>();

List<string> lines = reader.ReadAllLines();

foreach (string individualLine in lines)

{

// code specifics here..

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

int boxNumber, accountNumber;

boxNumber = Convert.ToInt32(fields[0]);

if (fields.Length > 1)

{

accountNumber = Convert.ToInt32(fields[1]);

data.Add(new SafetyDepositBox(boxNumber, accountNumber));

}

else

{

data.Add(new SafetyDepositBox(boxNumber));

}

}

return data;

}

private static List<SafetyDepositBox> LoadList(XMLFileIO<SafetyDepositBox> reader)

{

return reader.LoadAll();

}

#endregion

#region Write to a file

public static void SaveList(List<SafetyDepositBox> data, string fileName, FileFormat format, bool append)

{

if (format == FileFormat.CSV)

SaveList(new CSVFileIO(fileName), data, append);

else

SaveList(new XMLFileIO<SafetyDepositBox>(fileName), data, append);

}

private static void SaveList(CSVFileIO writer, List<SafetyDepositBox> data, bool append)

{

List<string> lines = new List<string>();

foreach (SafetyDepositBox item in data)

{

string lineOfText = item.BoxNumber.ToString();

if (item.IsLeased)

lineOfText += "," + item.AccountNumber;

lines.Add(lineOfText);

}

writer.WriteAllLines(lines, append);

}

private static void SaveList(XMLFileIO<SafetyDepositBox> writer, List<SafetyDepositBox> data, bool append)

{

writer.SaveAll(data, append);

}

#endregion

}