도서관리 프로그램

xml을 활용한 도서관리 프로그램

이진주

Contents


PART
01
“
개요
”

PART 01

개요

"도서 정보들과 도서관 사용자를 등록하여 전체 도서 수와 사용자 수, 대출 중인 도서 수 등을 표시하여 도서관을 관리하는 프로그램을 제작"

목적

1. 도서관에 있는 도서 목록의 검색, 수정, 추가, 삭제
2. 도서관 사용자의 추가 및 삭제

PART

02

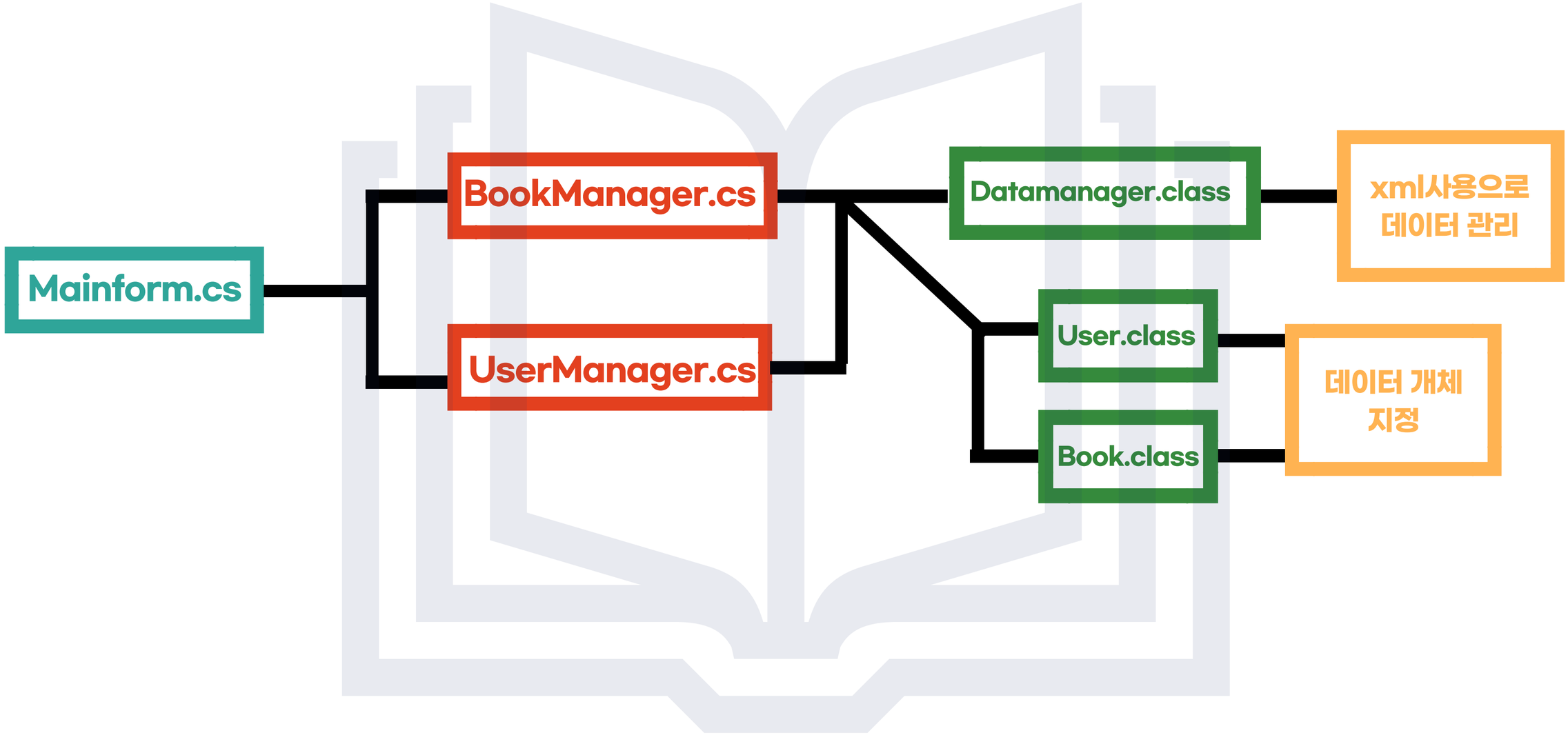
“

프로그램 구조

”

PART 02

프로그램 구조



PART

03

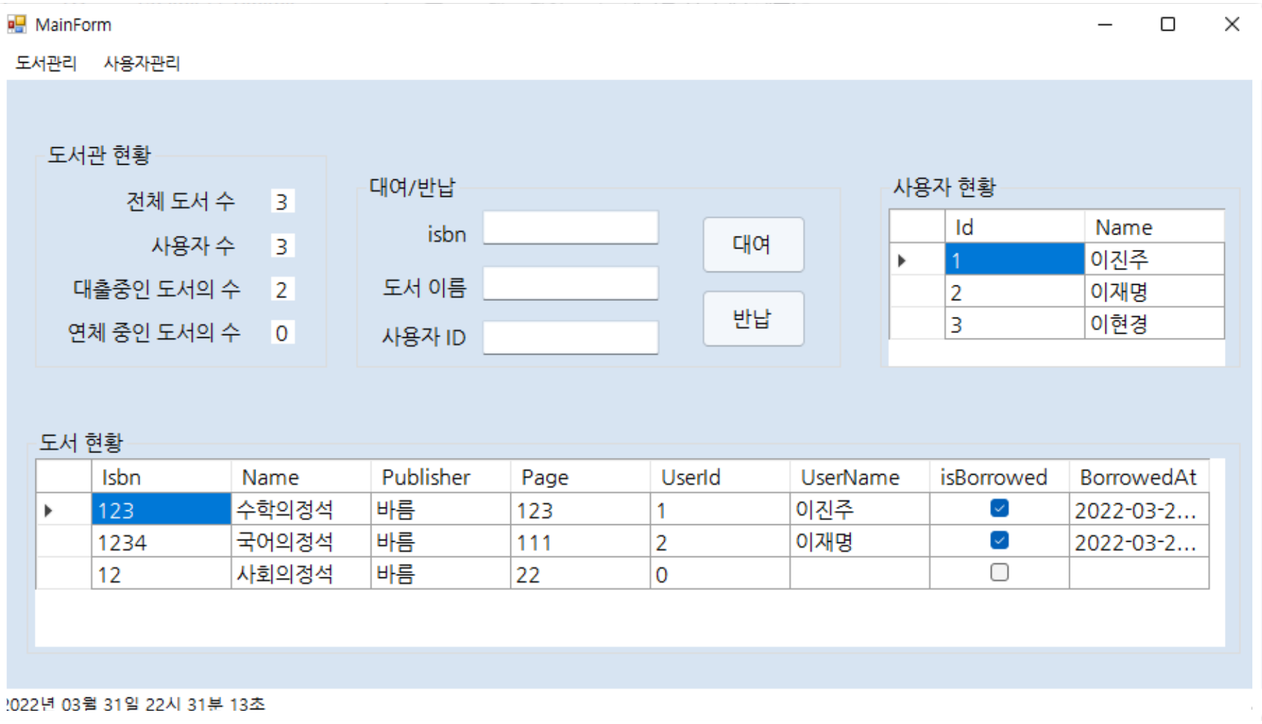
“

프로그램 실행 화면

”

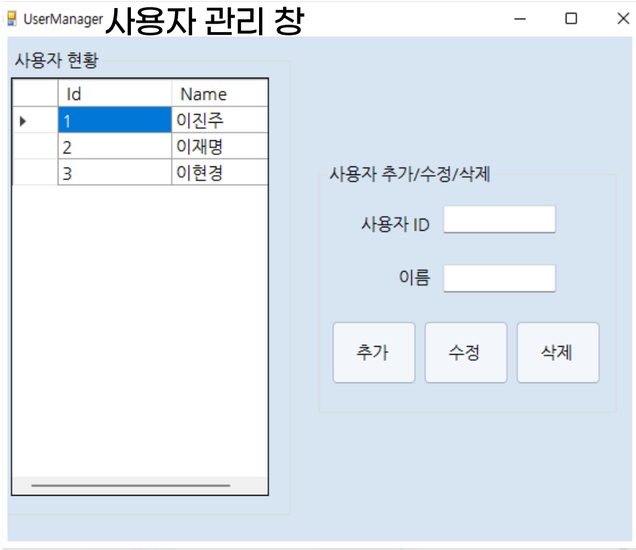
PART 03

프로그램 실행 화면



도서관리 창

사용자 관리 창



PART

04

“

상세 코드

”

Datamanager 코드

```
{
    public static List<Book> Books = new List<Book>();
    public static List<User> Users = new List<User>();


    /// <summary>
    /// 객체명
    /// </summary>
    const string BOOK = "book";
    const string USER = "user";

    /// <summary>
    /// 도서용
    /// </summary>
    const string ISBN = "isbn";
    const string NAME = "name";
    const string PUBLISHER = "publisher";
    const string PAGE = "page";
    const string ISBORROWED = "isBorrowed";
    const string BORROWEDAT = "borrowedAt";
    const string USERID = "userId";
    const string USERNAME = "username";

    /// <summary>
    /// 유저용
    /// </summary>
    const string UNAME = "name";
    const string UID = "id";
```

책 정보와 유저 정보를 담을 리스트 생성

컬럼명, 테이블(=객체)명을 상수로 저장해 가독성을 올리고, 오타를 방지

```
public static void Load()
{
    try
    {
        //Books.xml 파일을 읽어들임
        string booksOutput = File.ReadAllText(@"./Books.xml");
        //Booksxml에 있는 글자들을 읽어 들여서 XElement형태로 변환
        XElement BooksXElement = XElement.Parse(booksOutput);

        //Descendants 자손들, 태그명이 book인 것들을 모은 것
        //Books.xml은 books 태그 안에 여러 개의 book 태그가 있는 형태
        Books = (from item in BooksXElement.Descendants(BOOK)
                 select new Book()
                 {
                     Isbn = item.Element(ISBN).Value,
                     Name = item.Element(NAME).Value,
                     Publisher = item.Element(PUBLISHER).Value,
                     Page = int.Parse(item.Element(PAGE).Value),
                     UserId = int.Parse(item.Element(USERID).Value),
                     UserName = item.Element(USERNAME).Value,
                     BorrowedAt = DateTime.Parse(item.Element(BORROWEDAT).Value),
                     isBorrowed = item.Element(ISBORROWED).Value != "0" ? true : false
                 }).ToList<Book>();

        string usersOutput = File.ReadAllText(@"./Users.xml");
        XElement usersXElement = XElement.Parse(usersOutput);
        Users.Clear();
        foreach(var item in usersXElement.Descendants(USER))
        {
            User temp = new User();
            temp.Name = item.Element(UNAME).Value;
            temp.Id = int.Parse(item.Element(UID).Value);
            Users.Add(temp);
        }

    }
    catch (Exception e)
    {
        //파일이 없으면 이 부분으로 빠지게 되고, 파일을 다시 읽어들임.
        Save(); //만약 Save 실패시 StackOverFlow 에러에 걸림
        Load();
    }
}
```

xml파일 불러오는 함수

DataManager 코드

```
public static void Save()
{
    string booksOutput = "";
    booksOutput += "<books>\n";

    foreach(var item in Books)
    {
        booksOutput += $"\t<{BOOK}>\n";
        booksOutput += $"\t\t<{ISBN}>{item.Isbn}</{ISBN}>\n";
        booksOutput += $"\t\t<{NAME}>{item.Name}</{NAME}>\n";
        booksOutput += $"\t\t<{PUBLISHER}>{item.Publisher}</{PUBLISHER}>\n";
        booksOutput += $"\t\t<{PAGE}>{item.Page}</{PAGE}>\n";
        booksOutput += $"\t\t<{BORROWEDAT}>{item.BorrowedAt}</{BORROWEDAT}>\n";
        booksOutput += $"\t\t<{ISBORROWED}>" + (item.isBorrowed ? 1 : 0) + $"</{ISBORROWED}>\n";
        booksOutput += $"\t\t<{USERID}>{item.UserId}</{USERID}>\n";
        booksOutput += $"\t\t<{USERNAME}>{item.UserName}</{USERNAME}>\n";
        booksOutput += $"\t</{BOOK}>\n";
    }

    booksOutput += "</books>";
    Console.WriteLine(booksOutput);
    File.WriteAllText(@"./Books.xml", booksOutput); //xml파일에 값 넣는 것


    string usersOutput = "";
    usersOutput += "<users>\n";
    foreach(var item in Users)
    {
        usersOutput += $"\t<{USER}>\n";
        usersOutput += $"\t\t<{UID}>{item.Id}</{UID}>\n";
        usersOutput += $"\t\t<{UNAME}>{item.Name}</{UNAME}>\n";
        usersOutput += $"\t</{USER}>\n";
    }
    usersOutput += "</users>";
    Console.WriteLine(usersOutput);
    File.WriteAllText(@"./Users.xml", usersOutput);
}
```

xml파일에 저장하는 함수

책이 대여되면 1, 대여되지 않았다면 0을 넣는 방식으로 진행됨

1 : true, 0 : false

books.xml파일에는 이런식으로 저장

```
<books>
    <book>
        <isbn>1234</isbn>
        <name>국어의정석</name>
        <publisher>바름</publisher>
        <page>111</page>
        <borrowedAt>2022-03-31 오후 11:04:34</borrowedAt>
        <isBorrowed>1</isBorrowed>
        <userId>2</userId>
        <username>이재명</username>
    </book>
    <book>
        <isbn>12</isbn>
        <name>사회의정석</name>
        <publisher>바름</publisher>
        <page>22</page>
        <borrowedAt>0001-01-01 오전 12:00:00</borrowedAt>
        <isBorrowed>0</isBorrowed>
        <userId>0</userId>
        <username></username>
    </book>
</books>
```

MainForm 코드

```
public MainForm()
{
    InitializeComponent();

    label_allBookCount.Text = DataManager.Books.Count.ToString();
    label_allUserCount.Text = DataManager.Users.Count.ToString();
```

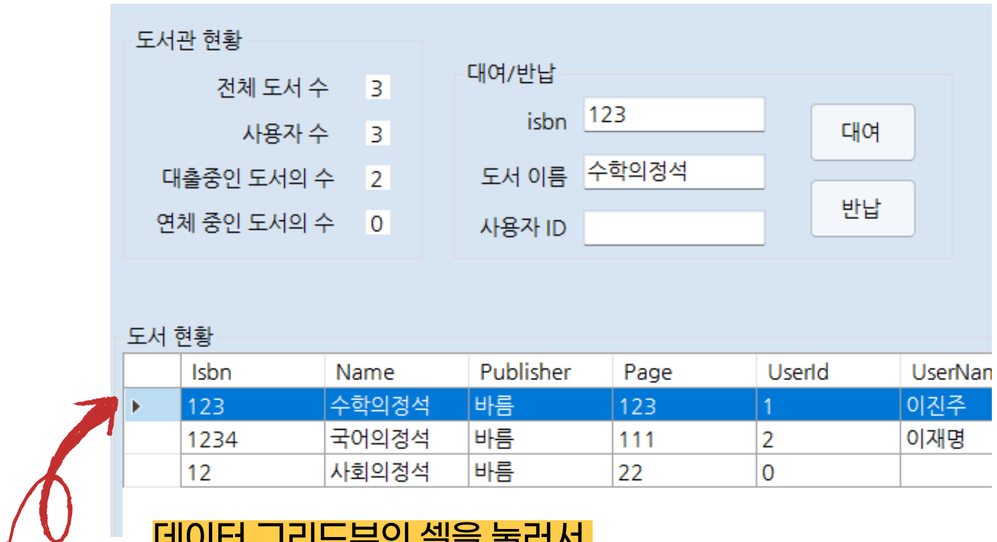
전체 유저수와 도서 수 보여주기

대여중인 도서와 연체된 도서 개수 보여주기

```
    label_allBorrowedBook.Text = DataManager.Books.Where(x=>x.isBorrowed).Count().ToString();

    label_allDelayedBook.Text = DataManager.Books.Where
        (x=> x.isBorrowed&&x.BorrowedAt.AddDays(7)<DateTime.Now).Count().ToString();
```

빌린 날짜에서 7 일이 지나면 연체



데이터 그리드뷰의 셀을 눌러서

대여/반납 안에 있는 텍스트박스(isbn, 도서이름)

들에 값을 채워넣음

```
dataGridView_bookManager.CellClick += (sender, e) =>
{
    Book book = dataGridView_bookManager.CurrentRow.DataBoundItem as Book;
    textBox_isbn.Text = book.Isbn;
    textBox_bookName.Text = book.Name;
};

dataGridView_userManager.CellClick += delegate (object sender, DataGridViewCellEventArgs e)
{
    User user = dataGridView_userManager.CurrentRow.DataBoundItem as User;
    textBox_id.Text = user.Id.ToString();
};

button_borrow.Click += Button_Borrow_Click;
button_return.Click += Button_Return_Click;
```

MainForm 코드

```
private void Button_Borrow_Click(object sender, EventArgs e)
{
    if(textBox_isbn.Text.Trim() == "") //Trim() : 양옆 공백 제거
        MessageBox.Show("isbn 값이 없습니다.");
    else if(textBox_id.Text.Trim() == "")
        MessageBox.Show("사용자 id 입력하세요.");
    else
    {
        try
        {

            Book book = DataManager.Books.Single((x) => x.Isbn == textBox_isbn.Text);
            if(book.isBorrowed)
            {
                MessageBox.Show("이미 빌렸습니다.");
            }
            else
            {

                User user = DataManager.Users.Single((x) => x.Id.ToString() == textBox_id.Text);
                book.UserId = user.Id;
                book.UserName = user.Name;
                book.isBorrowed = true;
                book.BorrowedAt = DateTime.Now;

                dataGridView_bookManager.DataSource = null;
                dataGridView_bookManager.DataSource = DataManager.Books;
                DataManager.Save(); //Books.xml, Users.xml에 내용 덮어쓰기
                MessageBox.Show($"{book.Name}이/가 {user.Name}님께 대여되었습니다.");
            }
        }
        catch (Exception)
        {
            MessageBox.Show("존재하지 않는 도서 혹은 사용자입니다.");
        }
    }

}
```

이미 대여중인 책인지 확인

대여중인 책이 아니라면 대여

MainForm 코드

```
rivate void Button_Return_Click(object sender, EventArgs e)

    if(textBox_isbn.Text.Trim()=="")
        MessageBox.Show("isbn입력하세요.");
    else
    {
        try
        {
            Book book = DataManager.Books.Single((x) => x.Isbn == textBox_isbn.Text);
            if (book.isBorrowed)
            {
                DateTime oldDay = book.BorrowedAt; //반납적에 언제 빌린건지 날짜 받아옴
                book.UserId = 0; //없다고 가정
                book.UserName = "";
                book.isBorrowed = false;
                book.BorrowedAt = new DateTime(); //0001년 과 같은 비어있는 값 들어감

                dataGridView_bookManager.DataSource = null;
                dataGridView_bookManager.DataSource = DataManager.Books;
                DataManager.Save();

                //연체여부 출력
                TimeSpan timeDiff = DateTime.Now - oldDay;
                if(timeDiff.Days>7)
                    MessageBox.Show(book.Name+"은 연체 상태로 반납");
                else
                    MessageBox.Show(book.Name+"은 정상 반납.");

            }
            else
            {
                MessageBox.Show("대여상태 아닙니다.");
            }
        }
        catch (Exception)
        {
            MessageBox.Show("없는 책입니다.");
        }
```

현재날짜에서 대여날짜를 뺀값이 7일보다 클때 연체상태

BookManager 코드

도서 정보 추가

```
private void button_add_Click(object sender, EventArgs e)
{
    //책을 추가하기 전에 해당하는 isbn이 있는지 여부 체크
    bool existBook = false;
    foreach(var item in DataManager.Books)
    {
        if(item.Isbn == textBox_isbn.Text)
        {
            existBook = true;
            break;
        }
    }
    if(existBook)
        MessageBox.Show("이미 존재하는 도서입니다.");
    else //책을 새로 등록할 수 있음
    {
        Book book = new Book();
        book.Isbn = textBox_isbn.Text;
        book.Name = textBox_bookName.Text;
        book.Publisher = textBox_publisher.Text;
        book.Page = int.Parse(textBox_page.Text);
        DataManager.Books.Add(book);

        dataGridView_books.DataSource = null;
        dataGridView_books.DataSource = DataManager.Books;
        DataManager.Save();
    }
}
```

도서 정보 수정

```
private void button_modify_Click(object sender, EventArgs e)
{
    Book book = null;
    for(int i = 0; i<DataManager.Books.Count; i++)
    {
        if(DataManager.Books[i].Isbn == textBox_isbn.Text)
        {
            book = DataManager.Books[i];
            book.Name = textBox_bookName.Text;
            book.Publisher = textBox_publisher.Text;
            book.Page = int.Parse(textBox_page.Text);

            dataGridView_books.DataSource = null;
            dataGridView_books.DataSource = DataManager.Books;
            DataManager.Save();
            break;
        }
    }
    if(book == null)
        MessageBox.Show("존재하지 않는 도서입니다.");

}
```

isbn값이 같을 때 수정/삭제 가능

도서 정보 삭제

```
private void button_delete_Click(object sender, EventArgs e)
{
    bool existBook = false;
    for(int i = 0; i<DataManager.Books.Count; i++)
    {
        if(DataManager.Books[i].Isbn == textBox_isbn.Text)
        {
            DataManager.Books.RemoveAt(i);
            existBook = true;
            break;
        }
    }
    if(existBook == false)
        MessageBox.Show("없는 책입니다.");
```

UserManager 코드

사용자 정보 추가

```
private void button_add_Click(object sender, EventArgs e)
{
    //Exists
    //괄호 안에 있는 조건이 해당되면 true를 반환
    if(DataManager.Users.Exists(x=>x.Id == int.Parse(textBox_ID.Text)))
        MessageBox.Show("해당 ID의 유저 이미 존재");
    else
    {
        //생성자 자체에는 아무것도 없음.
        //선언과 동시에 멤버변수에 값을 부여하는 방식(중괄호 안에 속성이 자동완성으로 나타남)
        User user = new User() { Id = int.Parse(textBox_ID.Text), Name = textBox_Name.Text };
        DataManager.Users.Add(user);

        dataGridView_Users.DataSource = null;
        dataGridView_Users.DataSource = DataManager.Users;
        DataManager.Save(); //바뀐 Users를 xml에 새로 반영시킴
    }
}
```

사용자 정보 수정

```
private void button_modify_Click(object sender, EventArgs e)
{
    try
    {
        User user = DataManager.Users.Single(x => x.Id == int.Parse(textBox_ID.Text));
        user.Name = textBox_Name.Text;
        try
        {
            //만약 유저의 이름을 바꿨는 데, 그 유저가 책을 빌린 유저라면
            //Books의 UserName도 같이 바꿔야 한다.
            Book book = DataManager.Books.Single((x) => x.UserId == int.Parse(textBox_ID.Text));
            book.UserName = textBox_Name.Text;
        }
        catch (Exception)
        {
            //해당 유저ID가 책 빌린 거 없으면 아무 것도 안 한다.
        }
    }
    catch (Exception)
    {
        //해당 아이디가 없으면 아무것도 안 함
        MessageBox.Show("이 아이디는 없습니다.");
    }

    dataGridView_Users.DataSource = null;
    dataGridView_Users.DataSource = DataManager.Users;
    DataManager.Save();
```

Thank You!