**26장 데이터 입력하기, 데이터 보기**

|  |
| --- |
| * 데이터를 리스트형식으로 만들어 관리 하는 방법 * 데이터를 저장하고, 탐색하는 방법 * 저장된 데이터를 불러와 다양한 방식 (정렬, 필터링, 그룹)으로 보는 방법 |

**예제 01\_SingleRecordDataEntry**

**– 여러 개의 파일 저장 가능, but 각 파일에는 한 개의 객체만 포함**

|  |
| --- |
| **<Person.cs>**  🡺 관리하고자 하는 데이터를 클래스로 정리  🡺 공개프로퍼티를 사용해 데이터 필드에 접근할 수 있는 인터페이스를 만듦  🡺 InotifyPropertyChanged 인터페이스를 제공하는 클래스 이므로  프로퍼티를 데이터 바인딩 소스로 사용가능 |
| **<PersonPanel.xaml>**  🡺 데이터 입력폼 제작 (재사용을 위해 클래스를 따로 만듦)  🡺 TextBox, DatePicker 컨트롤 부분에 프로퍼티 바인딩 정보 표시  - Path : Person클래스의 어느 프로퍼티에 바인딩할까  - Mode : 동작모드 (23장 데이터 파인딩 참고) |
| **<PersonPanel.cs>**  🡺 컴파일된 페이지 로드 (초기화) |
| **<SingleRecodeDataEntry.xaml>**  🡺 프로젝트의 프로그램 창에 대한 정의  🡺 메뉴 추가 및 PersonPanel.xaml에서 만든 패널을 불러서 넣음  - <pnl:PersonPanel x:Name = “pnlPerson”/> |
| **<SingleRecodeDataEntry.cs>**  🡺 Executed 이벤트 핸들러 작성 |

**예제 02\_MultiRecordDataEntry**

**– 객체를 집합으로 관리**

|  |
| --- |
| \* Person.cs, PersonPanel.cs, PersonPanel.xaml 파일은 첫번째예제에서 만들어 둔거 사용 |
| **<People.cs>**  🡺 Person 객체들의 컬렉션을 관리하는 역할을 함  🡺 ObservableCollection을 사용해 InotifyPropertyChanged 인터페이스를 노출시킴  - ObservableCollection : 항목이 추가 또는 제거 되거나 전체 목록이 새로 고쳐질때  알림을 제공하는 동적 데이터 컬렉션  🡺 Load, Save 메소드 (메뉴 관련) 정의 |
| **<MultiRecodeDataEntry.xaml>**  🡺 메뉴가 포함된 프로그램 창의 레이아웃 정의  🡺 New, Open, Save 메뉴 제작 및 기능(탐색, 추가, 삭제)별 버튼 만들기  🡺 SingleRecordDataEntry의 SingleRecodeDataEntry.xaml 에 버튼만 추가 |
| **<MultiRecordDataEntry.cs>**  🡺 메뉴 항목과 6개의 버튼에 대한 이벤트 핸들러 작성 |

**예제 03\_DataEntryWithNavigation**

**– 네비게이션 바를 활용한 데이터 탐색 및 리스트를 사용한 데이터 관리**

**※ 네비게이션을 만드는 프로젝트는 ‘클래스라이브러리’ 로 만듦**

|  |
| --- |
| \* Person.cs, PersonPanel.cs, PersonPanel.xaml 파일은 만들어 둔거 사용 |
| **<NavigationBar.xaml>**  🡺 네비게이션 바 컨트롤의 모습을 윈도우폼 컨트롤과 비슷하게 제작 |
| **<NavigationBar.cs>**  🡺 Ilist 타입의 Collection 프로퍼티 정의  - CollectionView객체를 생성하고 객체가 항상 최신상태를 유지할 수 있게 하는 이벤트 등록   |  | | --- | | - IList 인터페이스 :  집합적인 개념과 동기화 지원을 위한 기능을 가지고 있으며,  기본적인 배열 타입이 이 인터페이스를 구현하고 있다.  데이터의 삽입, 삭제, 검색, 인덱스등의 작업을 할 수 있다. |     🡺 ItemType 프로퍼티 정의  - 컬렉션에 저장하는 항목의 타입  - Add, New 버튼의 동작을 위해 사용(MultiRecordDataEntry 예제에서 봤던 코드랑 비슷) |
| **<DataEntryWithNavigation.xaml>**  🡺 DockPanel을 사용한 프로그램창 정의  🡺 PersonPanel.xaml에서 만든 패널과 NavigationBar.xaml 에서 만든 컨트롤을 불러서 넣음 |
| **<DataEntryWithNavigation.cs>**  🡺 New, Open메뉴에 대한 이벤트 핸들러를 InitializeNewPeopleObject 메소드를 호출해서  People객체를 PersonPanel과 네비게이션 바에 연결 |

**예제 04\_DataEntryWithListBox**

**- 리스트 박스를 사용해 저장된 데이터를 전체적으로 파악하기 (03예제에 리스트박스만 추가)**

|  |
| --- |
| \* People.cs, Person.cs, PersonPanel.cs, PersonPanel.xaml 파일은 만들어 둔거 사용 |
| **<DataEntryWithListBox.xaml>**  🡺 데이터베이스에 저장된 항목을 보여주기위한 리스트 박스 정의  - TextBlock을 사용하여 동적으로 텍스트 표시  🡺 PersonPanel.xaml에서 만든 패널을 불러서 넣음 |
| **<DataEntryWithListBox.cs>**  🡺 메뉴, 버튼 관련 이벤트 부분은 전부 이전에 만든 프로젝트와 동일  🡺 InitializeNewPeopleObject() 부분에서 리스트박스와 PersonPanel을 연결 |

**예제 05\_CollectionViewWithFilter**

**- CollectionView의 뷰를 변경 할 수 있는 3가지 수단 중 필터링 사용하기**

|  |
| --- |
| **<CollectionViewWithFilter.xaml>**  🡺 필터링을 위한 라디오 버튼과, 데이터를 보여줄 리스트 박스 정의  🡺 3개의 라디오 버튼 컨트롤은 동일한 이벤트 핸들러를 공유 |
| **< CollectionViewWithFilter.cs>**  🡺 라디오 버튼에 대한 이벤트 핸들러 작성  🡺 라디오 버튼에 할당된 Name 프로퍼티를 기반으로 CollectionView의 Filter프로퍼티를 설정  (살아있는애, 죽은애, 전체보기)  🡺 프로퍼티 설정시 Filter를 null로 함으로써 전체보기 가능  🡺 Refresh메서드를 사용해 리스트뷰 갱신 |