Лабораторная работа 1. Вариант 1

Механизм привязки данных в Windows Presentation Foundation

Темы:

- **Объектные ресурсы WPF**
- **Определение привязки в XAML и коде**
- **Привязка к коллекции. Представление коллекции. Поддержка текущего элемента в коллекции**
- **Шаблоны данных (DataTemplate)**

В лабораторной работе надо создать пользовательский интерфейс приложения для работы с коллекцией **TeamObservable** элементов **Person, Researcher** и **Programmer** (из лабораторных работ прошлого семестра). Пользовательский интерфейс приложения дает возможность добавлять в коллекцию новые элементы, удалять элементы, сохранять коллекцию в файле, загружать коллекцию из файла.

В среде VisualStudio создать решение (solution) с двумя проектами:

- тип одного проекта – библиотека классов (class library), в которой находятся типы **TeamObservable, Person, Researcher, Programmer;**
- тип второго проекта – приложение **WPF**.

Класс TeamObservable

Класс **TeamObservable** определяется как производный от класса **System.Collections.ObjectModel.ObservableCollection<Person>**.
Элементы **Person, Programmer** и **Researcher** хранятся в коллекции, которая находится в базовом классе **ObservableCollection<Person>**.

Класс **TeamObservable** содержит

- открытый конструктор **TeamObservable(string name)**;
- открытое свойство (с методами **get** и **set**) типа **string** для названия группы;
- закрытое поле и открытое свойство (с методом **get**) типа **List<string>** с темами исследований, которые выполняются в группе; этот список инициализируется в конструкторе и в дальнейшем не изменяется; при добавлении в коллекцию нового элемента типа **Researcher** пользователь сможет выбрать тему только из этого списка;
- открытое свойство (с методом **get**) типа **double** для информации о проценте исследователей в команде;
- открытое свойство (с методами **get** и **set**) типа **bool**, в котором хранится информация о том, что пользователь внес изменения в коллекцию после того, как коллекция была сохранена в файле;

- открытый метод void **AddPerson (params Person[] persons)** для добавления в список **List<Person>** новых элементов типа **Person**, **Researcher** и **Programmer**;
- открытый метод void **RemovePersonAt (int index)** для удаления из коллекции элемента с индексом **index**;
- открытый метод **AddDefaults()**, в котором в **TeamObservable** добавляется несколько элементов типа **Person**, **Researcher** и **Programmer**;
- открытый метод **AddDefaultProgrammer()**, в котором в **TeamObservable** добавляется один элемент **Programmer** с данными по умолчанию;
- открытый метод **AddDefaultResearcher()**, в котором в **TeamObservable** добавляется один элемент **Researcher** с данными по умолчанию;
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки со значениями всех данных класса, в том числе с данными каждого элемента коллекции;
- открытый статический метод **bool Save(string filename, TeamObservable obj)** для сохранения объекта **TeamObservable** в файле **filename** с помощью сериализации; метод возвращает значение **true**, если сериализация завершилась успешно, и значение **false** в противном случае;
- открытый статический метод **bool Load(string filename, ref TeamObservable obj)** для восстановления объекта **TeamObservable** из файла **filename** с помощью десериализации; метод возвращает значение **true**, если десериализация завершилась успешно, и значение **false** в противном случае.

Класс главного окна приложения содержит закрытое поле типа **TeamObservable**.

Требования к пользовательскому интерфейсу программы

Главное окно приложения содержит элементы управления для вывода коллекции **TeamObservable** и элементы управления для ввода данных при добавлении в коллекцию **TeamObservable** нового элемента типа **Researcher**.

Главное окно приложения содержит следующие элементы управления для информации о **TeamObservable**

- **TextBox** для названия группы;
- **TextBlock** для информации о проценте исследователей в группе;
- **TextBlock** для информации о том, что пользователь внес изменения в коллекцию после того, как коллекция была сохранена в файле.

Главное окно приложения содержит два элемента управления **ListBox:**

- **ListBox** для вывода всей коллекции из класса **TeamObservable**;
- **ListBox** для вывода подмножества элементов, которые имеют тип **Researcher.**

Все перечисленные выше элементы управления надо связать с данными TeamObservable с помощью привязки.

Свойству **DataContext** главного окна приложения надо присвоить ссылку на **TeamObservable** и использовать **DataContext** как источник данных в привязках.

Для привязки к элементам управления **ListBox,** которые используются для вывода подмножества коллекции, состоящего из элементов типа **Researcher,** надо создать **представление коллекции**.

Вывод всех элементов TeamObservable в элемент управления LIstBox

Главное окно приложения содержит две кнопки **RadioButton**, с помощью которых пользователь может выбрать, каким образом отдельные элементы коллекции выводятся в **ListBox** для всей коллекции. Одна из радиокнопок включает использование шаблона **DataTemplate**, другая радиокнопка отвечает случаю, когда для элементов **ListBox** не используется шаблон.

Шаблон данных **DataTemplate** надо определить в объектных ресурсах. Шаблон данных содержит только один элемент управления **TextBlock** и использует в привязке пользовательский преобразователь типа. Пользовательский преобразователь типа для каждого элемента из коллекции **TeamObservable** формирует строку, содержащую имя, фамилию и информацию о том, что это программист или исследователь.

Вывод элементов Researcher

Все элементы из **TeamObservable**, которые имеют тип **Researcher,** выводятся в свой элемент управления **ListBox** с использованием шаблона **DataTemplate**. Шаблон данных содержит только один элемент управления **TextBlock** и использует в привязке пользовательский преобразователь типа. Пользовательский преобразователь типа для каждого элемента **Researcher** формирует строку, содержащую только фамилию и первую букву имени.

Для элемента **Researcher**, который пользователь выбрал в **ListBox** со списком исследователей, приложение выводит дополнительную информацию в следующие элементы управления:

- **TextBlock** для даты рождения;
- **TextBlock** для темы исследований;
- **TextBlock** для числа публикаций.

Добавление в коллекцию нового элемента Researcher

Приложение содержит следующие элементы для ввода данных для нового элемента **Programmer**, который добавляется в коллекцию **TeamObservable**:

- **TextBox** для имени;
- **TextBox** для фамилии;
- **DatePicker** для даты рождения;

- **ComboBox** для выбора темы исследований; в этот элемент управления выводится список тем исследований из класса **TeamObservable;**
- **TextBox** для числа публикаций.

Для того, чтобы добавить в коллекцию **TeamObservable** новый элемент **Researcher**, свойства которого инициализированы данными из элементов управления, в классе главного окна приложения надо определить поле типа **Researcher** и выполнить привязку свойств этого объекта **Researcher** к свойствам соответствующих элементов управления. Все данные будут передаваться в новый объект **Researcher** механизмом привязки.
При добавлении нового элемента в коллекцию **TeamObservable** необходимо либо сделать глубокую копию объекта **Researcher**, либо снова распределить память для поля **Researcher** после добавления объекта в коллекцию.

Меню приложения

Главное окно приложения содержит меню с элементами

- **File** (с элементами **New**, **Open**, **Save**);
- **Edit** (с элементами **Add Default Researcher**, **Add Default Programmer**, **Add Custom Researcher**, **Add Defaults**, **Remove**.

С элементом управления **LIstBox** для вывода элементов коллекции, которые имеют тип **Researcher**, надо связать контекстное меню с элементами

- **Add Default Researcher**;
- **Add Custom Researcher**.

Реакция приложения на выбор пользователем элементов меню File:

New – элементы управления очищаются. Перед этим проверяется значение булевского поля с информацией о том, что в процессе работы приложения пользователь изменил коллекцию **TeamObservable**. Если пользователь внес изменения в коллекцию, выводится стандартный диалог **System.Windows.MessageBox** с запросом на сохранение данных или отказ от сохранения.

Open – пользователь выбирает имя файла в стандартном диалоге **Microsoft.Win32**.**OpenFileDialog**. Если пользователь сделал выбор (закрыл диалог кнопкой **Open**)

- проверяется значение булевского поля с информацией о том, что в процессе работы приложения пользователь изменил коллекцию **TeamObservable**; если пользователь внес изменения в коллекцию, выводится стандартный диалог **System.Windows.MessageBox** с запросом на сохранение данных или отказ от сохранения;
- выполняется десериализация всей коллекции **TeamObservable**;
- десериализованные данные выводятся в элементы управления.

Save – пользователь выбирает имя файла в стандартном диалоге **Microsoft.Win32.SaveFileDialog**. Если пользователь сделал выбор (закрыл диалог кнопкой **Save**), вся коллекция **TeamObservable** сериализуются в файл с именем, который выбрал пользователь, и изменяется значение булевского поля с информацией о том, что пользователь внес изменения в коллекцию.

Реакция приложения на выбор пользователем элементов меню Edit

- **Add Default Researcher** – в коллекцию добавляется элемент **Researcher** с данными по умолчанию;
- **Add Default Programmer** – в коллекцию добавляется элемент **Programmer** с данными по умолчанию;
- **Add Defaults** – в коллекцию добавляется несколько элементов разных типов (вызывается метод **AddDefaults()** из **TeamObservable)**;
- **Add Custom Researcher** – в коллекцию добавляется новый элемент **Researcher,** данные которого инициализируются из элементов управления;
- **Remove** – из коллекции удаляется элемент, который пользователь выбрал в **ListBox** со всей коллекцией.

Элементы контекстного меню для **ListBox** со списком программистов дублируют реакцию на соответствующие элементы меню **Edit**.

Обновление элементов управления TextBlock с информацией о проценте исследователей в группее и об изменении коллекции

Значение свойства класса **TeamObservable** с информацией о проценте исследователей в проекте изменяется при добавлении в коллекцию нового элемента и при удалении элементов коллекции. Для того, чтобы при этом автоматически обновлялся элемент **TextBlock**, необходимо в классе **TeamObservable** реализовать интерфейс **INotifyPropertyChanged.** Свойство класса **TeamObservable** с информацией о проценте исследователей должно уведомлять о том, что его значение изменилось.

Для того, чтобы обновлялся элемент **TextBlock** с информацией о том, что пользователь внес изменения в коллекцию после того, как она была сохранена в файле, булевское свойство с информацией о изменении коллекции также должно уведомлять об изменении своего значения.

Информация об изменении коллекции TeamObservable

Приложение сохраняет в поле булевского типа информацию о том, что в процессе работы приложения пользователь изменил коллекцию **TeamObservable**.

Базовый класс **ObservableCollection<Person>** отслеживает изменение коллекции, т.е. множества ссылок на объекты типа **Person**, **Programmer** и **Researcher**, которые образуют коллекцию. При изменении числа ссылок или значения одной из них базовый класс бросает событие **System.Collections.Specialized.CollectionChanged**. Если

коллекция используется как источник в привязке, **WPF** обновляет данные в элементах управления.

Событие **CollectionChanged** надо использовать для обновления значения булевского поля, в котором хранится информация о том, что коллекция изменилась в процессе работы приложения. Для этого надо подписаться на событие **CollectionChanged**, которое унаследовал класс **TeamObservable** от базового класса, и в обработчике этого события изменять значение булевского поля. В этом случае не придется отслеживать изменения коллекции в обработчиках событий от элементов пользовательского интерфейса. Преимущество такого подхода состоит в том, что его реализация не зависит от конкретной модели пользовательского интерфейса.

После десериализации объекта **TeamObservable** надо снова подписаться на событие **CollectionChanged** (так как значение ссылки изменилось).

Сохранение данных

Если перед выбором элементов меню **New** или **Open** или перед выходом из приложения пользователь изменил коллекцию **TeamObservable** и не сохранил ее (не сериализовал в файл), он получает предупреждение о том, что данные будут потеряны**.** Предупреждение выводится с помощью стандартного диалога **System.Windows.MessageBox**, в котором пользователю предлагается выбрать – сохранить в файле измененные данные или выполнить соответствующую операцию без сохранения результатов. Если пользователь выбрал сохранение данных, то вызывается стандартный диалог **Microsoft.Win32**.**SaveFileDialog** для выбора имени файла, в который будут сериализованы данные объекта **TeamObservable**.

При **завершении работы приложения** проверку того, что пользователь внес изменения в коллекцию, надо выполнить обработчике события **Closed** главного окна приложения.

Обработка исключений

Все исключения, которые могут возникать при обработке некорректного ввода пользователя, должны обрабатываться приложением.
Все операции открытия файла, сериализации и десериализации данных должны находиться в блоках **try-catch-finally**. Независимо от того, корректно были введены данные или при вводе были допущены ошибки, все файловые потоки должны быть закрыты.
Приложение должно оставаться в рабочем состоянии до тех пор, пока пользователь не закроет главное окно приложения.

Замечание: Во всех классах надо закомментировать переопределение виртуальных методов bool Equals (object obj); int GetHashCode(); и операций == и != для того, чтобы не было проблем с выбором (select) элементов в **ListBox** в случае, когда в элементы управления **ListBox** выводятся данные соответствующих типов.

Срок сдачи лабораторной работы

301, 302, 309 группы - 17 марта;

341/2 группа - 23 марта.