Лабораторная работа 2. Второй уровень. Вариант 1

Дополнительные возможности в механизме привязки данных в Windows Presentation Foundation. Взаимодействие WPF и WindowsForms

Темы:

- Проверка корректности данных в привязке. Интерфейс **System.IDataErrorInfo**
- Маршрутизируемые события и команды
- Элемент управления **Chart** из **WindowsForms**

Требования к программе

В лабораторной работе надо создать пользовательский интерфейс для приложения, который дает возможность просмотреть в графическом виде данные, заданные на равномерной сетке и зависящие от параметра.

В приложении надо определить классы

- **ModelData** для данных;
- **ObservableModelData**, производный от **ObservableCollection<ModelData>,** для коллекции объектов типа **ModelData**;
- **ModelDataView.**

В классе **ModelData** хранятся значения некоторой функции, зависящей от параметра **p,** заданные на отрезке [0,1] на равномерной сетке.
Допустимые значения параметра **p** находятся в заданном диапазоне **[pMin, pMax]**.

Класс **ModelData** содержит следующие открытые свойства

- типа **int** для числа узлов сетки;
- типа **double** для параметра **p**;
- типа **double[]** для узлов сетки;
- типа **double[]** для вычисленных значений функции в узлах сетки.

Класс **ModelData** содержит следующие статические поля и/или свойства

- статическое поле **double pMin** для нижней границы допустимых значений параметра **p**;
- статическое поле **double pMax** для верхней границы допустимых значений параметра **p**;

- статическое поле **double nMin** для нижней границы числа точек на отрезке [0,1], в которых задана функция;
- статическое поле **double nMax** для верхней границы числа точек на отрезке [0,1], в которых задана функция;

В классе **ModelData** определить

- конструктор с двумя параметрами – типа **int** для числа узлов сетки и типа **double** для значения **p**; в конструкторе на заданной сетке вычисляются значения функции, которая зависит от параметра **p**; функция "моделирует" измеренные данные, используется только для того, чтобы получить значения функции на сетке, и вызывается только из конструктора класса **ModelData**;
- Метод **F(x)**, который вычисляет значение функции в любой точке отрезка [0,1]; если точка x находится между узлами сетки x_j и x_{j+1}, для вычисления значения в этой точке функция аппроксимируется на отрезке $[x_j, x_{j+1}]$ с помощью линейной функции.

В классе **ModelData** реализовать интерфейс **System.IDataErrorInfo**. Реализация интерфейса **IDataErrorInfo** должна проверять, что число узлов сетки находится в диапазоне **[nMin, nMax],** а значение параметра **p** – в диапазоне **[pMin, pMax]**.

Класс **ObservableModelData** содержит

- открытое свойство (с методами **get** и **set**) типа **bool**, в котором хранится информация о том, что пользователь внес изменения в коллекцию;
- открытый метод **Add_ModelData(ModelData modelData)**, в котором в коллекцию добавляется элемент **modelData**;
- метод **Remove_At (int index)** для удаления из коллекции элемента с номером **index**;
- открытый метод **AddDefaults ()**, в котором в коллекцию добавляется несколько элементов **ModelData** для разных значений параметров **n** и **p**;
- метод, который для заданного значения **x** из отрезка [0,1] для всех элементов **ModelData** из коллекции вычисляет значения функции в точке **x;**
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с данными класса.

Класс **ModelDataView** содержит

- конструктор с параметром типа **ObservableModelData;**
- свойство типа **double;** это свойство используется в привязке к элементу управления **TextBox,** в который пользователь вводит значение **x** из отрезка [0,1]**;**
- свойство типа **string;** это свойство используется в привязке к элементу **ComboBox,** с помощью которого пользователь выбирает формат вывода чисел при оцифровке осей**;**
- метод **void Draw (Chart chart, IList selectedDataModels).**

В классе **ModelDataView** реализовать интерфейс **System.IDataErrorInfo**. Реализация интерфейса **IDataErrorInfo** должна проверять что значение **x** находятся в диапазоне [0, 1].

Пользовательский интерфейс приложения содержит

- элемент управления **ListBox** для вывода элементов коллекции **ObservableModelData**;
- для элемента управления **ListBox** установить режим, при котором пользователь может выбрать в нем несколько элементов;
- для элемента управления **ListBox** использовать шаблон **DataTemplate**, который содержит элементы для вывода значения параметра **p** и числа узлов сетки **n**;
- элемент управления **Chart** для вывода в графическом виде данных из коллекции **ObservableModelData**;
- элемент управления **TextBox** для ввода значения **x**;
- элемент управления **ComboBox** для выбора формата вывода чисел при оцифровке осей.

Пользовательский интерфейс приложения содержит элементы управления для ввода данных для нового элемента **ModelData**, который добавляется в коллекцию:

- элемент управления **TextBox** для ввода значений параметра **p**;
- элемент управления **TextBox** для ввода числа узлов сетки.

Проверка корректности данных в привязке.

В привязках к элементам управления **TextBox** для ввода значения параметра **p,** числа узлов сетки **n** и значения **x** надо предусмотреть проверку корректности данных с использованием интерфейса **IDataErrorInfo.**
Все элементы управления, в которые пользователь вводит данные, должны быть подписаны.
Должны быть выведены границы допустимых значений для параметра **p**, числа узлов сетки **n** и значения **x**.

Команды

Главное окно приложения содержит

- меню **File** (с элементами **New**, **Open**, **Save**), меню **Edit** (с элементами **Add** и **Remove**), меню **Draw**;
- элемент управления **ToolBar с** кнопками **Open, Save**, **Add, Remove и Draw**.

Реакция на все элементы меню и кнопки панели должна быть реализована с помощью команд.

С элементом меню и кнопкой панели инструментов **New** надо связать команду **ApplicationCommands.New.**

- В обработчике события **Execute** команды проверяется, что пользователь сохранил коллекцию **ObservableModelData**, и создается новая коллекция.

С элементом меню и кнопкой панели инструментов **Open** надо связать команду **ApplicationCommands.Open.**

- В обработчике события **Execute** команды проверяется, что пользователь сохранил коллекцию **ObservableModelData**, и десериализуется коллекция **ObservableModelData** из файла, который пользователь выбрал в стандартном диалоге **OpenFileDialog**.

С элементом меню и кнопкой панели инструментов **Save** надо связать команду **ApplicationCommands.Save.**

- В обработчике события **CanExecute** команды проверяется значение булевского поля класса **ObservableModelData** с информацией о том, что в процессе работы приложения пользователь изменил коллекцию.
- В обработчике события **Execute** команды коллекция **ObservableModelData** сериализуется в файл, который пользователь выбрал в стандартном диалоге **SaveFileDialog**.

С элементом меню и кнопкой панели инструментов **Remove** надо связать команду **ApplicationCommands.Delete**.

- В обработчике события **CanExecute** команды **ApplicationCommands.Delete** проверяется, что в элементе управления **ListBox** с коллекцией есть выбранный элемент.
- В обработчике события **Execute** команды из коллекции удаляются элементы **ModelData**, выбранные пользователем в элементе управления **ListBox**.

Определить пользовательские команды **AddModelCommand и DrawCommand.**

Пользовательская команда **AddModelCommand** связана с элементом меню и кнопкой **Add**.

- Обработчик события **CanExecute** команды должен проверить, что данные, которые пользователь ввел в элементы управления для инициализации нового объекта **ModelData**, не содержат ошибок. В обработчике **CanExecute** надо проверить значение свойства **HasError** привязки для элементов управления, связанных со свойствами **ModelData**, которые контролируется в **IDataErrorInfo**.
- Обработчик события **Execute** команды добавляет элемент **ModelData** в коллекцию **ObservableModelData**.

Пользовательская команда **DrawCommand** связана с элементом меню и кнопкой **Draw**.

- Обработчик события **CanExecute** команды должен проверить, что
  - в элементе управления **ListBox** с коллекцией есть хотя бы один выбранный элемент;
  - пользователь ввел в элемент **TextBox** допустимое значение **x.** В обработчике **CanExecute** надо проверить значение свойства **HasError** привязки для этого элемента управления.
- Обработчик события **Execute** команды вызывает метод **Draw (Chart chart, IList selectedItems)** класса **ModelDataView** для вывода графики в элемент управления **Chart**.

Вывод графики в элемент управления Chart.

В элементе управления **Chart** данные из **ObservableModelData** выводятся в две области вывода (**Chart Area**).

В одной области вывода **Chart Area** выводятся графики значений функции в узлах сетки для всех элементов **ModelData**, которые пользователь выбрал в элементе управления **ListBox**.
В этой области вывода

- графики выводятся без маркеров;
- при оцифровке осей используется формат, выбранный пользователем;
- значения параметра **p** для элементов **ModelData** выводятся в **Legend**.

В другой области вывода **Chart Area** выводится зависимость значения функции в заданной точке **x** от параметра **p**. Для всех значений параметра **p**, которые есть в коллекции, вычисляются значения функции в точке **x**. Горизонтальная ось графика содержит значения параметра **p**.
В этой области вывода

- графики функции выводятся с маркерами для каждой точки ряда данных;
- при оцифровке осей используется формат, выбранный пользователем;
- с каждой точкой ряда данных связывается **ToolTip**, который содержит значение параметра **p** и значение функции в этой точке.

Сохранение данных

Если перед выбором элементов меню **New** или **Open** или перед выходом из приложения пользователь изменил коллекцию **ObservableModelData** и не сохранил их (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 **ObservableModelData**.

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.

Срок сдачи лабораторной работы:

группы 301, 302, 309 - 14 апреля;

группа 341/2 - 20 апреля.