Задание №5. Логика главного окна

Описание задания:

В предыдущих заданиях были реализованы отдельно структуры данных и верстка приложения. В данном задании необходимо связать интерфейс и бизнес-логику. Для этого нужно:

1. организовать хранение объекта класса Project в главном окне;
2. реализовать добавление, отображение и удаление данных проекта в списке главного окна;
3. реализовать отображение выбранного в списке элемента на правой панели главного окна.

Пользовательский интерфейс должен быть связан с объектом класса Project, т.е. любые действия пользователя приводят к изменению в проекте, а изменения в проекте должны отображаться на пользовательском интерфейсе.

Поскольку в рамках данного задания еще не реализована логика второго окна (где пользователь вручную может вводить данные), в этом задании мы воспользуемся программной генерацией данных. Программная генерация бывает удобна для отладки приложений или на начальных этапах разработки, когда часть функциональности еще не готова. В следующих заданиях мы заменим программную генерацию на ручной ввод данных.

В ходе работы не забывайте делать коммиты в репозиторий.

Последовательность работы:

1. Создайте новую ветку в локальном репозитории под названием «features/5_add_mainform_logic». Перейдите в новую ветку.
2. Добавьте в класс MainForm закрытое поле _project типа Project. Поле должно инициализироваться новым объектом Project при объявлении.
3. Создайте закрытый метод UpdateListBox(). Метод должен очищать все элементы списка ListBox, а затем с помощью цикла for или foreach добавлять в него данные из коллекции внутри объекта _project. Например, если в проекте хранятся контакты, то в список ListBox надо добавить фамилию каждого контакта из проекта; а если в проекте хранятся фильмы – то добавить название каждого фильма и т.д.

4. Создайте закрытый метод Add***Object***() (где в качестве ***Object*** подставьте название вашей структуры данных, например, Contact, Note, Movie и т.д.). Поскольку у нас пока нет второго окна для создания данных, в данном методе необходимо прописать логику по программной генерации новых данных. Для этого используйте стандартный класс Random. Он предоставляет возможность генерации случайных целых и вещественных чисел. Генерация строк может быть основа на выборе случайной строки из заранее созданного массива строк. Метод должен завершаться добавлением нового объекта в _project.

5. Создайте обработчик события Click для кнопки добавления новых данных (согласно макету, она должна располагаться под списком в левой колонке пользовательского интерфейса). В обработчик добавьте последовательные вызовы метода Add***Object***() и UpdateListBox().

6. Запустите программу и убедитесь, что добавление новых данных выполняется правильно, и список ListBox в интерфейсе заполняется при каждом нажатии на кнопку. Если список не обновляется или генерация данных не работает, найдите и исправьте ошибки. Для проверки генерации данных используйте точки останова (breakpoints).

7. Добавление данных в программе осуществляется как через кнопку добавления, так и через пункт в главном меню Edit -> Add. Создайте обработчик события Click для пункта меню Add и также добавьте в него последовательные вызовы методов создания данных и обновления списка. Убедитесь, что пункт меню работает.

8. Теперь реализуем удаление выбранного элемента списка. При этом важно удалять данные как в списке ListBox, так и из хранилища пользовательских данных Project.

9. Создайте закрытый метод Remove***Object***(int index), который принимает на вход индекс выбранного элемента и удаляет его из проекта.

10. Создайте обработчик события Click для кнопки удаления выбранных данных (согласно макету, она также располагается под списком ListBox). Добавьте в обработчик вызов метода Remove***Object***(), при вызове передайте в него значение текущего выбранного элемента списка (ListBox.SelectedIndex). После вызова метода Remove***Object***() добавьте вызов метода UpdateListBox().

11. Запустите программу и убедитесь, что удаление работает правильно. Убедитесь, что удаляются именно выбранные элементы списка.

12. Аналогично реализуйте удаление данных через пункт главного меню.

13. При тестировании удаления вы могли обратить внимание, что программа падает в тех случаях, когда в списке ListBox ничего не выбрано или список пуст. Чтобы программа не падала, необходимо добавить в метод Remove***Object***() проверку входного значения – если значение индекса равно -1, то метод должен немедленного завершиться (если ничего не выбрано, то и не надо ничего удалять).

14. Протестируйте работу проверки на выбранный элемент.

15. Добавьте в метод Remove***Object***() вывод пользовательского сообщения (MessageBox.Show()) «Do you really want to remove …?», где в качестве «…» подставляется текущий выбранный элемент. Если пользователь выберет в окне OK, то удаление выбранного элемента выполнится, а если Cancel – ничего не произойдет.

16. Теперь реализуем логику, при которой при выборе любого элемента списка, его подробные данные отобразятся на панели справа.

17. Добавьте в главное окно закрытый метод UpdateSelected***Object***(int index), где в качестве входного аргумента будет передаваться текущий индекс выбранного элемента. Метод должен получить текущий объект по заданному индексу из поля _project и заполнить данные на правой панели главного окна согласно макету.

18. Создайте обработчик события SelectedIndexChanged для ListBox. Добавьте в обработчик вызов метода UpdateSelected***Object***(). Обновление списка вызывать не требуется, так как данные в нём не менялись. Запустите программу и убедитесь, что правая панель обновляется при выборе новых элементов в списке.

19. Когда в списке нет выбранного элемента или в списке нет данных, необходимо очищать правую панель. Для этого создайте еще один метод.

20. Добавьте в главное меню закрытый метод ClearSelected***Object***(). Метод должен очищать все элементы управления на правой панели главного окна.

21. Добавьте в обработчик события SelectedIndexChanged условие: если выбранный индекс равен -1, тогда вызвать метод ClearSelected***Object***(), иначе вызвать UpdateSelected***Object***().

22. Альтернативный вариант – сделать проверку на отрицательный индекс прямо в методе UpdateSelected***Object***() с последующим вызовом метода очистки. В данной реализации это не имеет принципиального значения.

23. Напоследок, добавьте обработчик события для пункта меню Exit и реализуйте в обработчике закрытие приложения. Программа обязательно должна спрашивать пользователя, действительно ли он хочет закрыть программу.

24. Убедитесь, что вся реализованная функциональность по добавлению, отображению и удалению данных работает правильно, в том числе через главное меню. Устраните ошибки, если они есть. Проверьте оформление кода, грамматические ошибки и наличие всех xml-комментариев в коде. Если всё верно, сделайте коммит и синхронизируйтесь.

25. В Visual Studio перейдите в ветку develop. Затем выполните слияние (merge) ветки features/5_add_mainform_logic в ветку develop. Теперь все изменения из ветки с добавленным решением должны переместиться в ветку develop. Убедитесь в этом с помощью GitHub.

Дополнительные задачи по желанию:

* - технически, при добавлении или удалении одного элемента в списке ListBox вызывается полное очищение и перезаполнение списка вновь. Такой подход применен в задании для унификации методов и может применяться, когда количество отображаемых элементов в списке небольшое, до нескольких сотен (либо, если это не приводит к продолжительным пересчетам данных в приложении). Однако, приложение может быть реализовано и без перезаполнения списка – это потребует от вас реализации чуть более сложных алгоритмов в будущем (например, когда добавится поиск и сортировка), но при разработке высокоэффективных серверных приложений разработчик должен стремиться к подобным оптимизациям.

** - к программной генерации данных можно относиться как к отдельной функции внутри программы. С точки зрения объектно-ориентированного кода, и с точки зрения чистоты кода внутри главного окна, любая отдельная функция должна выноситься в самостоятельный класс. Добавьте в пространстве имен View класс ***Object***Factory (можно сделать статическим классом), создайте в нём метод GenerateRandom() и перенесите в него логику генерации данных. Это избавит главное окно от лишнего кода и оставит в неё только работу данными.

*** - запрос, хочет ли пользователь закрыть программу, должен появляться не только при нажатии на пункт меню Exit, но также и при закрытии программы через крестик или через сочетание клавиш Alt+F4. Измените логику программы так, чтобы запрос на закрытие программы появлялся при любом варианте закрытия, при этом код не должен дублироваться.