

Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисципліни

“Розробка офісних застосувань”

для студентів спеціальностей

121 Інженерія програмного забезпечення, 122 Комп’ютерні
науки та інформаційні технології
(всіх форм навчання)

2017

Методичні вказівки до виконання лабораторних робіт з дисципліни “Розробка офісних застосувань” для студентів спеціальностей 121 Інженерія програмного забезпечення, 122 Комп’ютерні науки та інформаційні технології (всіх форм навчання) / А. О. Олійник, В. М. Льовкін, Є. М. Федорченко. – Запоріжжя : ЗНТУ, 2017. – 70 с.

Автори: А. О. Олійник, к.т.н., доцент
В. М. Льовкін, к.т.н., доцент
Є. М. Федорченко, ст. викладач

Рецензент: С. О. Субботін, д.т.н., професор

Відповідальний
за випуск: С. О. Субботін, д.т.н., професор

Затверджено
на засіданні кафедри
“Програмні засоби”

Протокол № 1
від “28” серпня 2017 р.

ЗМІСТ

Вступ	6
1 Лабораторна робота № 1 Створення проектів Microsoft Office....	7
1.1 Мета роботи	7
1.2 Основні теоретичні відомості.....	7
1.2.1 Загальні відомості про автоматизацію Office	7
1.2.2 Об'єктна модель офісних застосувань	8
1.2.3 Розробка консольного застосування, яке виконує автоматизацію офісного застосування	10
1.2.4 Розробка налаштування рівня документа	11
1.2.5 Розробка надбудови рівня застосування	12
1.3 Порядок виконання роботи	13
1.4 Зміст звіту.....	13
1.5 Контрольні запитання	14
2 Лабораторна робота № 2 Налаштування стрічки у проектах Microsoft Office	15
2.1 Мета роботи	15
2.2 Основні теоретичні відомості.....	15
2.2.1 Використання стрічки у програмах Microsoft Office	15
2.2.2 Об'єктна модель стрічки.....	17
2.2.3 XML-стрічки	19
2.3 Порядок виконання роботи	21
2.4 Зміст звіту.....	22
2.5 Контрольні запитання	23
3 Лабораторна робота № 3 Створення панелі дій і області задач у проектах Office	24
3.1 Мета роботи	24
3.2 Основні теоретичні відомості.....	24
3.2.1 Налаштування панелі дій документа або книги	24
3.2.2 Створення налаштовуваної області задач	26
3.3 Порядок виконання роботи	28
3.4 Зміст звіту.....	29
3.5 Контрольні запитання	29

4 Лабораторна робота № 4 Використання елементів керування ведучого застосування у проектах Word.....	30
4.1 Мета роботи	30
4.2 Основні теоретичні відомості.....	30
4.2.1 Ведучий елемент документа Word	30
4.2.2 Елементи керування вмістом у документах Word	31
4.2.3 Закладки у документах Word	33
4.3 Порядок виконання роботи	35
4.4 Зміст звіту.....	37
4.5 Контрольні запитання	37
5 Лабораторна робота №5 Використання елементів керування ведучого застосування у проектах Excel	39
5.1 Мета роботи	39
5.2 Основні теоретичні відомості.....	39
5.2.1 Ведучі елементи книги, аркуша, аркуша діаграми.....	39
5.2.2 Елементи керування діаграмою	41
5.2.3 Елементи керування ListObject	42
5.2.4 Елемент керування NamedRange	43
5.3 Порядок виконання роботи	45
5.4 Зміст звіту.....	46
5.5 Контрольні запитання	47
6 Лабораторна робота №6 Використання елементів керування Windows Forms у документах Office.....	49
6.1 Мета роботи	49
6.2 Основні теоретичні відомості.....	49
6.2.1 Загальні відомості про ведучі елементи й елементи керування ведучої програми	49
6.2.2 Загальні відомості про використання елементів керування Windows Forms в документах Office	52
6.2.3 Додавання елементів керування в документи Office під час виконання	54
6.3 Порядок виконання роботи	55
6.4 Зміст звіту.....	56
6.5 Контрольні запитання	57
7 Лабораторна робота №7 Робота з PDF-документами	58
7.1 Мета роботи	58

7.2 Основні теоретичні відомості.....	58
7.2.1 Загальний огляд бібліотек	58
7.2.2 Робота з бібліотекою iTextSharp	59
7.3 Порядок виконання роботи	61
7.4 Зміст звіту.....	62
7.5 Контрольні запитання	62
8 Лабораторна робота № 8 Створення дистрибутиву для	
Windows-застосувань	63
8.1 Мета роботи	63
8.2 Основні теоретичні відомості.....	63
8.2.1 Загальні відомості про дистрибутиви.....	63
8.2.2 Створення дистрибутиву для Windows-застосувань у	
Visual Studio 2010	63
8.3 Порядок виконання роботи	65
8.4 Зміст звіту.....	66
8.5 Контрольні запитання	66
Література.....	67
Додаток А Приклади індивідуальних завдань.....	68

ВСТУП

Дане видання призначене для вивчення та практичного освоєння студентами усіх форм навчання основ розробки офісних застосунків.

Відповідно до графіка студенти перед виконанням лабораторної або самостійної роботи повинні ознайомитися з конспектом лекцій та рекомендованою літературою. Звичайно, в дані методичні вказівки неможливо було внести весь матеріал, необхідний для виконання та захисту лабораторних робіт. Тому тут містяться основні, базові теоретичні відомості, необхідні для виконання лабораторних робіт. Таким чином для виконання лабораторної роботи та при підготовці до її захисту необхідно ознайомитись з конспектом лекцій та проробити весь матеріал, наведений в переліку рекомендованої літератури. При цьому не варто обмежуватись лише наведеним списком.

Для одержання заліку по кожній роботі студент здає викладачу цілком оформлений звіт. Звіт має містити:

- титульний аркуш;
- мету, варіант і завдання роботи;
- лаконічний опис теоретичних відомостей;
- текст програми, що обов'язково містить коментарі;
- вхідні та вихідні дані програми;
- змістовний аналіз отриманих результатів та висновки.

Звіт виконують на білому папері формату А4 (210 x 297 мм). Текст розміщують тільки з однієї сторони аркуша. Поля сторінки з усіх боків — 20 мм. Аркуші скріплюють за допомогою канцелярських скріпок. Для набору тексту звіту використовують редактор MS Word 97: шрифт Times New Roman, 14 пунктів. Міжрядковий інтервал: полуторний — для тексту звіту, одинарний — для лістингів програм, таблиць і роздруківок даних.

Під час співбесіди студент повинен виявити знання за метою роботи, теоретичним матеріалом, методами виконання кожного етапу роботи, змістом основних розділів розробленого звіту з демонстрацією результатів на конкретних прикладах. Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і здачі роботи студент повинен відповісти на контрольні питання, приведені наприкінці опису відповідної роботи. Загальний залік студент одержує після виконання і здачі останньої роботи.

1 ЛАБОРАТОРНА РОБОТА № 1

СТВОРЕННЯ ПРОЕКТІВ MICROSOFT OFFICE

1.1 Мета роботи

Вивчити основи створення проектів Microsoft Office.

1.2 Основні теоретичні відомості

1.2.1 Загальні відомості про автоматизацію Office

Функції застосування Microsoft Office можуть бути включені в рішення шляхом написання коду, який отримує доступ до об'єктної моделі застосування.

Завдяки використанню Microsoft Office у якості інтерфейсу для рішень розробники можуть скористатися перевагою звичних для користувача інтерфейсів системи Microsoft Office і таких засобів, як обробка текстів в програмі Word, функції аналізу даних програми Excel і функції керування електронною поштою Outlook.

Для налаштування застосувань Office і додавання особливих функціональних можливостей, необхідних для бізнес-процесів, можна розробляти рішення у Visual Studio.

У Visual Studio є наступні типи шаблонів проектів для розробки Office:

- налаштування рівня документа (Document-Level Customizations), пов'язані з конкретним документом;
- надбудови рівня застосування (Application-Level Add-ins), пов'язані з самим застосуванням.

Налаштування рівня документа складаються зі збірки, пов'язаної з окремим документом, книгою або шаблоном в Microsoft Office Word або Microsoft Office Excel. Налаштування не можуть провадити зміни на рівні програми, такі як відображення нового елемента меню або вкладки стрічки при відкритті будь-якого документа.

Коли користувач відкриває документ, застосування Microsoft Office завантажує відповідну збірку. Після завантаження збірки налаштування може відповідати на події у відкритому документі. Коли документ відкритий, налаштування може також звертатися до об'єктної моделі автоматизації і розширювати застосування, а також

може використовувати будь-які класи .NET Framework.

Надбудова рівня застосування складається зі збірки, пов'язаної з застосуванням Microsoft Office. Як правило, надбудова виконується при запуску пов'язаного застосування, хоча користувачі можуть також завантажувати збірки в ході роботи програми. Можливості створюваних надбудов доступні самому застосуванню незалежно від типу документів, що відкриваються.

1.2.2 Об'єктна модель офісних застосувань

При розробці рішень Word в Visual Studio виконується взаємодія з об'єктною моделлю Word, що складається з класів і інтерфейсів, які надаються в основній збірці взаємодії для Word і задаються в просторі імен Microsoft.Office.Interop.Word. На рис. 1.1 показано ієрархію об'єктної моделі Word.

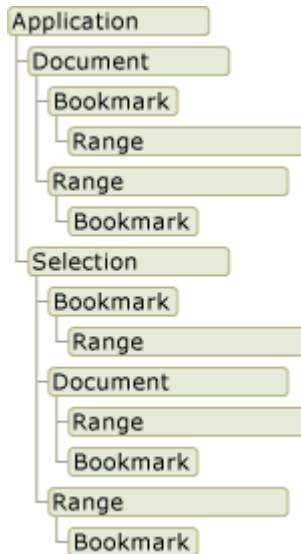


Рисунок 1.1 — Короткий довідник за об'єктною моделлю Word

Об'єкти верхнього рівня:

— об'єкт застосування Application представляє застосування Word і є батьківським для всіх інших об'єктів. Його члени зазвичай

застосовуються до Word у цілому, а властивості і методи можна використовувати для керування середовищем Word. У проектах надбудов рівня застосування (або документа) можна отримувати доступ до об'єкта Application за допомогою поля Application класу ThisAddIn (або ThisDocument);

- об'єкт документа Microsoft.Office.Interop.Word.Document;

- об'єкт вибору Selection представляє область, обрану в поточний момент. Якщо нічого не виділено, то цей об'єкт представляє точку вставки;

- об'єкт діапазону Range представляє безперервну область документа і задається положеннями початкового і кінцевого символів. Немає обмеження на кількість об'єктів Range.;

- об'єкт закладки Microsoft.Office.Interop.Word.Bookmark представляє безперервну область документа з початковою і кінцевою позиціями. Закладки можна використовувати для позначення якогось розташування в документі або як контейнер для тексту в документі.

Об'єкт документа є основою для програмування в Word. Він представляє документ і весь його вміст. При відкритті документа або створенні нового документа створюється новий об'єкт, який додається в колекцію Documents об'єкта Application. Документ, у якому знаходиться фокус, називається активним документом. Він представлений властивістю ActiveDocument об'єкта Application.

Засоби розробки Office в Visual Studio розширюють об'єкт Microsoft.Office.Interop.Word.Document, надаючи тип Microsoft.Office.Tools.Word.Document. Цей тип є ведучим елементом, містить додаткові події, а також дозволяє додавати керовані елементи керування і смарт-теги.

При створенні проекту рівня документа розробник може звернутися до членів Microsoft.Office.Tools.Word.Document за допомогою створеного класу ThisDocument у проекті. Доступ до членів ведучого елементу Microsoft.Office.Tools.Word.Document можна отримати за допомогою ключових слів Me або this з коду в класі ThisDocument або за допомогою Globals.ThisDocument з коду поза класом ThisDocument. Наприклад, для вибору параграфа документа використовується код: `this.Paragraphs[1].Range.Select()`;

У проектах рівня застосування можна створювати ведучі елементи Microsoft.Office.Tools.Word.Document під час виконання. Створений ведучий елемент можна використовувати для додавання

елементів керування у пов'язаний документ.

Об'єктна модель Excel точно відповідає інтерфейсу користувача. Об'єкт Application представляє застосування в цілому, і кожен з об'єктів Workbook містить колекцію об'єктів Worksheet. Звідси випливає, що основна абстракція, що представляє комірки, є об'єктом Range, що дозволяє працювати з окремими комірками або групою комірок.

При створенні нового проекту рівня застосування для Excel середовище Visual Studio автоматично створює файл коду ThisAddIn.cs. Доступ до об'єкту застосування можна отримати за допомогою властивості Me.Application або this.Application.

При створенні нового проекту рівня документа для Excel можна створити новий проект книги Excel або шаблону Excel. Середовище Visual Studio автоматично створює наступні файли коду в новому проекті Excel як для проектів книги, так і для проектів шаблону: ThisWorkbook.cs, Sheet1.cs, Sheet2.cs, Sheet3.cs.

Оскільки дані в документі Excel добре структуровані, модель об'єкта ієрархічна й пряма. До числа основних об'єктів відносяться:

а) застосування (Application): представляє безліч відомостей про виконувани додатки, параметри цього екземпляра і поточні об'єкти користувача, відкриті в екземплярі;

б) книга: об'єкт Microsoft.Office.Interop.Excel.Workbook представляє одну книгу в Excel;

в) аркуш: об'єкт Microsoft.Office.Interop.Excel.Worksheet є членом колекції Worksheets. Excel надає колекцію Sheets як властивість об'єкта Microsoft.Office.Interop.Excel.Workbook;

г) діапазон (об'єкт Microsoft.Office.Interop.Excel.Range) представляє комірку, рядок, стовпець і набір комірок, що містить один або декілька блоків комірок (які можуть бути суміжними або несуміжними) або групу комірок на декількох аркушах.

1.2.3 Розробка консольного застосування, яке виконує автоматизацію офісного застосування

Для створення консольного застосування у Visual Studio, яке виконує автоматизацію Word, загалом необхідно виконати наступну послідовність кроків:

а) у меню Файл виберіть пункти Создать та Проект;

б) у панелі шаблонів розверніть вузол Visual C#, оберіть вузол Windows, зі списку шаблонів проекту оберіть Консольное приложение;

в) у браузері рішень натисніть правою кнопкою миші на вузлі Ссылки та виберіть пункт меню Добавить ссылку...

г) у вікні, що відкриється, виберіть вкладку.NET та файл Microsoft.Office.Interop.Word.dll.

Приклад тексту файлу Program.cs:

```
using Word = Microsoft.Office.Interop.Word;
namespace AutomationWordConsoleApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Word.Application word = new Word.Application();
            Word.Document document = word.Documents.Add();
            Word.Range range = document.Range();
            range.Text = "Hello, World!";
            word.Visible = true;
        }
    }
}
```

1.2.4 Розробка налаштування рівня документа

Для того щоб створити новий проект документа Word:

- виберіть Файл → Создать → Проект;
- у панелі шаблонів розгорніть вузол Visual C#, а потім – Office;
- у розгорнутому вузлі Office виберіть версію Word;
- у списку шаблонів проекту виберіть Документ Word (2007, 2010 або 2013 відповідно);
- виберіть Создать новый документ.

Після виконання наведених дій Visual Studio створює проект FirstDocumentCustomization і додає документ FirstDocumentCustomization і файл коду ThisDocument у проект. Документ FirstDocumentCustomization відкривається в конструкторі автоматично.

Можна розробити користувацький інтерфейс налаштування, змінивши документ, відкритий у конструкторі.

Наприклад, можна додати текст, таблиці або елементи керування Word.

Для додавання тексту в документ програмними засобами необхідно у браузері рішення натиснути правою кнопкою миші на файл ThisDocument, потім вибрати Смотреть код та замінити обробник подій ThisDocument_Startup наступним кодом:

```
private void ThisDocument_Startup(object sender,
System.EventArgs e)
{
    this.Paragraphs[1].Range.InsertParagraphAfter();
    this.Paragraphs[2].Range.Text = "Цей текст доданий за
допомогою коду.";
}
```

1.2.5 Розробка надбудови рівня застосування

Для створення нового проекту надбудови Word в Visual Studio необхідно виконати дії, аналогічні налаштуванню рівня документа, обравши при цьому у списку шаблонів проекту Надстройка Word.

За замовчуванням файл коду ThisAddIn містить часткове визначення класу ThisAddIn та обробники подій ThisAddIn_Startup і ThisAddIn_Shutdown.

Для прикладу у файлі коду ThisAddIn додайте в клас ThisAddIn наступний код:

```
void Application_DocumentBeforeSave(Word.Document Doc, ref
bool SaveAsUI, ref bool Cancel)
{
    Doc.Paragraphs[1].Range.InsertParagraphBefore();
    Doc.Paragraphs[1].Range.Text = "Цей текст доданий за
допомогою коду.";}

```

А в обробник подій ThisAddIn_Startup необхідно додати наступний код, що використовується для підключення до обробника подій Application_DocumentBeforeSave події DocumentBeforeSave:

```
this.Application.DocumentBeforeSave += new
Word.ApplicationEvents4_DocumentBeforeSaveEventHandler
(Application_DocumentBeforeSave);
```

1.3 Порядок виконання роботи

1.3.1 Ознайомитися з основними теоретичними відомостями та рекомендованою літературою за темою роботи.

1.3.2 Відповідно до індивідуального завдання з додатку А для Word (підрозділ А.1) розробити:

- консольне застосування, що виконує автоматизацію Word;
- налаштування рівня документа;
- надбудову рівня застосування.

1.3.3 Пояснити необхідність та особливості використання даних підходів для рішення, що розробляється.

1.3.4 Відповідно до індивідуального завдання з додатку А для Excel (підрозділ А.2) розробити:

- консольне застосування, що виконує автоматизацію Excel;
- налаштування рівня документа;
- надбудову рівня застосування.

1.3.5 Пояснити необхідність та особливості використання даних підходів для рішення, що розробляється.

1.3.6 Виконати тестування розробленого програмного забезпечення.

1.3.7 Оформити звіт з роботи.

1.3.8 Відповісти на контрольні запитання.

1.4 Зміст звіту

1.4.1 Тема та мета роботи.

1.4.2 Короткі теоретичні відомості.

1.4.3 Завдання на лабораторну роботу.

1.4.4 Текст розробленого програмного забезпечення з коментарями.

1.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

1.4.6 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

1.5 Контрольні запитання

1.5.1 Що таке автоматизація застосувань Office?

1.5.2 Які типи шаблонів проектів для розробки Office пропонує Visual Studio?

1.5.3 Чим відрізняється документ від шаблону документа?

1.5.4 У чому полягає відмінність налаштування рівня документа від надбудови рівня застосування?

1.5.5 Яка архітектура у налаштування рівня документа?

1.5.6 Яка архітектура у надбудови рівня застосування?

1.5.7 Опишіть процес завантаження налаштування рівня документа?

1.5.8 Опишіть процес завантаження надбудови рівня застосування?

1.5.9 Які налаштовувані властивості документа використовуються при завантаженні налаштування рівня документа?

1.5.10 Які розділи реєстру використовуються при завантаженні надбудови рівня застосування?

1.5.11 Які способи реєстрації надбудови рівня застосування існують?

1.5.12 Які файли середовище Visual Studio автоматично створює при створенні нового проекту рівня застосування для Word?

1.5.13 Які файли середовище Visual Studio автоматично створює при створенні нового проекту рівня документа для Word?

1.5.14 Яким чином можна отримати доступ до об'єкту застосування у проекті рівня застосування для Word?

1.5.15 Для доступу до яких об'єктів можна використовувати клас Globals у проекті рівня документа Word?

1.5.16 Які об'єкти верхнього рівня існують в об'єктній моделі Word?

1.5.17 Які файли середовище Visual Studio автоматично створює при створенні нового проекту рівня застосування для Excel?

1.5.18 Які файли середовище Visual Studio автоматично створює при створенні нового проекту рівня документа для Excel?

1.5.19 Яким чином можна отримати доступ до об'єкту застосування у проекті рівня застосування для Excel?

1.5.20 Які об'єкти верхнього рівня існують в об'єктній моделі Excel?

2 ЛАБОРАТОРНА РОБОТА № 2 НАЛАШТУВАННЯ СТРІЧКИ У ПРОЕКТАХ MICROSOFT OFFICE

2.1 Мета роботи

Навчитися налаштовувати стрічку у проектах Microsoft Office засобами Microsoft Visual Studio.

2.2 Основні теоретичні відомості

2.2.1 Використання стрічки у програмах Microsoft Office

У випуску 2007 системи Microsoft Office з'явився новий елемент користувацького інтерфейсу – стрічка, що служить для впорядкування пов'язаних команд, щоб полегшити їх пошук. Команди представлені на стрічці у вигляді елементів керування, які об'єднані в групи вздовж горизонтальної смуги у верхній частині вікна програми. Близькі групи елементів керування розташовані на вкладках.

Щоб налаштувати стрічку програми Microsoft Office, необхідно додати функцію Стрічка (візуальний конструктор) або елемент Стрічка (XML) в проект Office.

Щоб додати стрічку в проект, необхідно у меню Проект вибрати команду Додати новий елемент, а у діалоговому вікні вибрати функцію Стрічка (візуальний конструктор) або Стрічка (XML).

Елемент Стрічка (візуальний конструктор) надає розширені засоби, що полегшують проектування і розробку власної стрічки, і використовується для налаштування стрічки наступними способами:

- додавання на стрічку користувацьких або вбудованих вкладок;
- додавання на призначену для користувача або вбудовану вкладку користувацьких груп;
- додавання елементів керування користувача в призначену для користувача групу;
- додавання елементів керування користувача в меню Office.

Елемент Стрічка (XML) використовується в тому випадку, коли необхідно налаштувати стрічку способом, що не підтримується елементом Стрічка (візуальний конструктор). Даний елемент

використовується для налаштування стрічки такими способами:

- додавання вбудованих груп на призначену для користувача або вбудовану вкладку;
- додавання вбудованих елементів керування в призначену для користувача групу;
- додавання користувацького вузла, що перевизначає обробники подій вбудованих елементів керування;
- налаштування панелі швидкого доступу;
- спільне використання параметрів настроювання стрічки декількома надбудовами за допомогою повного ідентифікатора.

Конструктор стрічок є засобом візуальної розробки і використовується для додавання користувацьких вкладок, груп і елементів керування на стрічку. Конструктор можна використовувати для виконання розробки макета стрічки, обробки подій і встановлення властивостей елементів керування та налаштування меню кнопки Microsoft Office.

При додаванні нового елемента Стрічка (візуальний конструктор) Visual Studio автоматично додає в проект наступні файли: файл коду стрічки, коду конструктора стрічок та файл ресурсів, який містить значення властивостей кожного елемента керування на стрічці.

За наявності елемента Стрічка (візуальний конструктор) з іншого проекту, його можна повторно використовувати в поточному проекті, викликавши діалогове вікно Додавання існуючого елемента.

Конструктор стрічок за замовчуванням містить вкладку і групу. У конструктор стрічок також можна додати користувацькі вкладки, групи та елементи керування. Ці елементи керування знаходяться на Панелі елементів у групі Елементи керування стрічки Office.

Елементи керування не можна перетягувати з однієї вкладки на стрічці на іншу вкладку. Для цього необхідно скористатися командами Вирізати та Вставити, а для підключення обробника подій скористатися вікном Властивості.

Вбудована вкладка – це вкладка, яка вже є на стрічці програми Microsoft Office. Наприклад, вкладка Дані є вбудованою вкладкою Excel. На вбудовану вкладку можна додавати групи і елементи керування. Видалити вбудовані групи не можна.

Події та властивості стрічки знаходяться в просторі імен

Microsoft.Office.Tools.Ribbon. Елемент Стрічка (візуальний конструктор) автоматично додає в проект посилання на цю збірку і вставляє необхідний оператор using або Imports в початок файлу коду стрічки.

2.2.2 Об'єктна модель стрічки

Середовище виконання Visual Studio Tools for Office надає суворо типізовану об'єктну модель, яку можна використовувати для отримання та встановлення значень властивостей елементів керування стрічки під час виконання. Наприклад, можна динамічно наповнювати елементи керування меню, а також приховувати або показувати елементи керування залежно від контексту. Також на стрічку можна додавати вкладки, групи та елементи керування, але тільки до тих пір, поки стрічка не буде завантажена застосуванням Office.

Клас стрічки містить 3 події:

- RibbonBase.Load: виникає при завантаженні налаштування стрічки застосуванням Office. У файл коду стрічки автоматично додається обробник подій Load, який використовується для виконання користувацького коду при завантаженні стрічки;

- RibbonBase.LoadImage: дозволяє кешувати зображення в налаштуванні стрічки при її завантаженні. Якщо код буде налаштований на кешування зображень стрічки в цьому обробнику подій, це дозволить домогтися деякого поліпшення продуктивності;

- RibbonBase.Close: виникає при закритті екземпляра стрічки.

Простір імен Microsoft.Office.Tools.Ribbon містить типи для кожного елемента керування, що відображається у групі Елементи керування стрічки Office в панелі Панель елементів: RibbonBox (поле), RibbonButton, RibbonButtonGroup, RibbonCheckBox, RibbonComboBox, RibbonDropDown, RibbonEditBox, RibbonGallery (колекція), RibbonGroup, RibbonLabel, RibbonMenu, RibbonSeparator, RibbonSplitButton, RibbonTab, RibbonToggleButton.

При додаванні елемента керування в конструктор стрічки конструктор оголошує клас для цього елемента керування як поле у файлі коду конструктора стрічки.

Деякі властивості елементів керування стрічки можна встановити тільки до завантаження стрічки. Існує три місця для

встановлення цих властивостей: у вікні Властивості Visual Studio, в конструкторі класу стрічки та в методі CreateRibbonExtensibilityObject класу ThisAddin, ThisWorkbook або ThisDocument проекту.

Динамічні меню надають деякі винятки. Можна створювати нові елементи керування, задавати їх властивості і потім додавати їх в динамічне меню під час виконання, навіть після завантаження стрічки, яка містить меню. Властивості елементів керування, що додаються в динамічне меню, можна задавати в будь-який момент часу.

Наступний приклад додає до групи нову кнопку, якщо поточний час дорівнює 17:00 за тихоокеанським часом (UTC-8) або пізніше:

```
public Ribbon1 ()
: Base (Globals.Factory.GetRibbonFactory ())
{InitializeComponent ();
    if (System.DateTime.Now.Hour > 16)
    {
        Microsoft.Office.Tools.Ribbon.RibbonButton button =
            this.Factory.CreateRibbonButton ();
        button.Label = "New Button";
        group1.Items.Add (button);
    }
}
```

Властивості елемента керування стрічки можна задати шляхом перевизначення методу CreateRibbonExtensibilityObject в класах проекту ThisAddin, ThisWorkbook або ThisDocument.

Кожен клас елемента керування містить одну або декілька подій: Click, TextChanged, ItemsLoading, ButtonClick, SelectionChanged, DialogLauncherClick. Обробники даних подій мають два параметри: sender та e (об'єкт RibbonControlEventArgs, що містить властивість Microsoft.Office.Core.IRibbonControl і використовується для доступу до будь-якої недоступної в об'єктній моделі стрічки властивості).

Для зміни положення вкладки на стрічці використовується властивість PositionType, значення якої BeforeOfficeId розміщує групу перед зазначеною вбудованою вкладкою, AfterOfficeId – після зазначеної вбудованої вкладки. Для додавання груп на вбудовану вкладку використовується властивість ControlIdType.

У конструкторі стрічок можна додати елементи керування в меню, яке відкривається при виборі вкладки Файл (для Microsoft

Office 2010) або натисканні кнопки Microsoft Office (для Microsoft Office 2007). У проектах Office 2010 елементи керування, що додаються на вкладку Файл, відображаються у групі Надбудови. У проектах Office 2007 елементи керування можна розмістити поряд з будь-яким вбудованим елементом керування.

У проектах Office 2010 неможливо розмістити елементи керування до або після вбудованих елементів керування, скориставшись конструктором стрічки в Visual Studio. Якщо потрібно розмістити елементи керування до або після вбудованих елементів керування, необхідно скористатися XML-стрічкою.

2.2.3 XML-стрічки

Елемент Стрічка (візуальний конструктор) підтримує не всі типи налаштувань стрічки. Щоб отримати доступ до додаткових налаштувань стрічок, можна експортувати стрічку з конструктора в XML-стрічку і внести зміни безпосередньо в XML.

Щоб виконати експорт стрічки з конструктора стрічок в XML-стрічку необхідно в браузері рішень натиснути правою кнопкою миші файл коду стрічки та вибрати пункт Конструктор уявлень, після чого натиснути правою кнопкою миші на конструктор стрічок і вибрати команду Експортувати стрічку в XML-стрічку.

Visual Studio додає XML-файл стрічки і файл XML-коду стрічки в проект. У класі коду стрічки розміщують коментарі, що починаються з TODO:. У файлі XML-коду у області Ribbon Callbacks записуються методи зворотного виклику для обробки дій користувача.

Елемент Стрічка (XML) дозволяє налаштовувати стрічку за допомогою XML. При додаванні елементу Стрічка (XML) Visual Studio автоматично додає в проект XML-файл стрічки, де визначається користувацький інтерфейс стрічки, та файл коду стрічки, що містить клас стрічки, а додатки Microsoft Office використовують екземпляр цього класу для завантаження користувацької стрічки.

За замовчуванням ці файли додають призначену для користувача групу на вкладку стрічки Надбудови.

Після додавання в проект елемента Стрічка (XML), в клас ThisAddin, ThisWorkbook або ThisDocument необхідно додати код, який перекриє метод CreateRibbonExtensibilityObject і поверне застосуванням Office екземпляр класу XML-стрічок.

Методи зворотного виклику дозволяють реагувати на дії користувача і схожі на події елементів керування Windows Forms, але вони ідентифікуються за атрибутом у XML користувача інтерфейсу елемента. Методи створюються в класі стрічки, а елемент керування викликає той метод, ім'я якого збігається зі значенням атрибута. Методи зворотного виклику створюються в два етапи: призначити елементу керування в XML-файлі стрічки атрибут, який ідентифікує метод зворотного виклику в коді, та визначити метод зворотного виклику в класі стрічки. Наприклад: `<toggleButton id="toggleButton1" onAction="OnToggleToggleButton1" />`

До методу зворотного виклику пред'являється ряд вимог: він повинен бути оголошений як відкритий, ім'я має відповідати імені методу зворотного виклику, призначеного елементу керування в XML-файлі стрічки, а сигнатура повинна відповідати сигнатурі методу зворотного виклику, доступного для відповідного елемента керування стрічки.

Всі методи зворотного виклику мають параметр `Microsoft.Office.Core.IRibbonControl`, який представляє елемент керування, що викликає метод. Цей параметр дозволяє використовувати один метод зворотного виклику для декількох елементів керування.

Визначити призначену для користувача стрічку можна, додавши необхідні елементи і атрибути в XML-файл стрічки. Типово XML-файл стрічки має такий зміст (group – група елементів керування на вкладці tab стрічки ribbon):

```
<? Xml version = "1.0" encoding = "UTF-8"?>
<customUI xmlns="http://schemas.microsoft.com/office/2006/
01/customui" onLoad="OnLoad">
  <ribbon>
    <tabs>
      <tab idMso="TabAddIns">
        <Group id = "MyGroup"
          label = "My Group">
          </ Group>
        </ Tab>
      </ Tabs>
    </ Ribbon>
  </ CustomUI>
```

У таблиці 2.1 описані стандартні атрибути XML-файла стрічки.

Таблиця 2.1 – Стандартні атрибути XML-файла стрічки

Атрибут	Батьківський елемент	Опис
onLoad	customUI	метод, що викликається застосуванням при завантаженні стрічки
idMso	tab	вбудована вкладка, яка відображається на стрічці
id	group	група
label	group	текст, який відображається для групи

Visual Studio створює клас стрічки у файлі коду стрічки. У нього вручну необхідно додати методи зворотного виклику для елементів керування, що знаходяться на стрічці. Даний клас реалізує інтерфейс Microsoft.Office.Core.IRibbonExtensibility. У таблиці 2.2 описані методи за замовчуванням даного класу.

Таблиця 2.2 – Методи за замовчуванням класу Стрічка

Метод	Опис
GetCustomUI	Повертає вміст XML-файла стрічки. Викликається в для отримання XML-стрічки, що визначається користувацьким інтерфейсом. Даний метод реалізує метод IRibbonExtensibility.GetCustomUI(String).
OnLoad	Присвоює параметр Microsoft.Office.Core.IRibbonControl полю ribbon. Викликається при завантаженні користувацької стрічки. Можна використовувати для динамічного оновлення користуницької стрічки.
GetResource Text	Викликається методом GetCustomUI для отримання вмісту XML-файла стрічки.

2.3 Порядок виконання роботи

2.3.1 Ознайомитися з основними теоретичними відомостями та рекомендованою літературою за темою роботи.

2.3.2 Створити проект рівня документа Word.

2.3.3 Додати в проект елемент Стрічка. Її налаштування за наступними пунктами завдання виконувати за допомогою конструктора у відповідності з узгодженим з викладачем завданням.

2.3.4 Додати на стрічку вкладку, а на вкладці розташувати необхідні елементи керування, розмістивши їх у відповідних групах.

2.3.5 Додати елементи керування в меню, яке відкривається при виборі вкладки Файл.

2.3.6 Додати кнопки запуску діалогового вікна в кожен групу стрічки.

2.3.7 Призначити обробники подій для елементів керування стрічки та кнопок запуску діалогового вікна в групах стрічки.

2.3.8 Виконати експорт стрічки з конструктора стрічок в XML-стрічку.

2.3.9 Створити проект рівня застосування Word.

2.3.10 Додати у проект рівня застосування копії файлів, отриманих в результаті експорту у пункті 2.3.8.

2.3.11 Додати в проект код для завантаження експортованої XML-стрічки.

2.3.12 Створити методи зворотного виклику для обробки дій користувача зі стрічкою.

2.3.13 Виконати тестування розробленого програмного забезпечення.

2.3.14 Оформити звіт з роботи.

2.3.15 Відповісти на контрольні питання.

2.4 Зміст звіту

2.4.1 Тема та мета роботи.

2.4.2 Короткі теоретичні відомості.

2.4.3 Завдання на лабораторну роботу.

2.4.4 Текст розробленого програмного забезпечення з коментарями.

2.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

2.4.6 Висновки, що містять відповіді на контрольні запитання, та відображують результати виконання роботи та їх критичний аналіз.

2.5 Контрольні запитання

2.5.1 Як налаштувати перегляд помилок інтерфейсу користувача надбудови?

2.5.2 Як увімкнути відображення вкладки розробника на стрічці?

2.5.3 Що таке стрічка?

2.5.4 Яким чином можна додати стрічку в проект?

2.5.5 Які типи елементів стрічок можна додати в проект?

2.5.6 Які можливості надає елемент «Стрічка (візуальний конструктор)»?

2.5.7 Які можливості надає елемент «Стрічка (XML)»?

2.5.8 Чим вбудована вкладка (група) стрічки відрізняється від користувацької?

2.5.9 Які файли Visual Studio автоматично додає в проект при додаванні елемента «Стрічка (візуальний конструктор)»?

2.5.10 Якими способами можна додавати елементи керування в конструкторі стрічки?

2.5.11 Яким чином можна додати вкладку користувача на стрічку перед або після вбудованої вкладки?

2.5.12 Яким чином можна додати групу користувача на вкладку стрічки перед або після вбудованої групи?

2.5.13 Яким чином можна додати елемент керування в групу вкладки стрічки перед або після вбудованого пункту меню?

2.5.14 Яким чином можна додати елементи керування в меню, яке відкривається при виборі вкладки Файл?

2.5.15 Яким чином можна додати кнопку запуску діалогового вікна в групу стрічки і назначити для неї обробник?

2.5.16 Яким чином можна виконати експорт стрічки з конструктора стрічок в XML-стрічку?

2.5.17 Які файли Visual Studio автоматично додає в проект при додаванні елемента «Стрічка (XML)»?

2.5.18 Яким чином можна задати обробники подій для елементів керування XML-стрічки?

2.5.19 Які обмеження накладено на методи зворотного виклику для обробників подій елементів керування XML-стрічки?

2.5.20 Опишіть стандартні елементи XML-файлу стрічки та їх атрибути.

3 ЛАБОРАТОРНА РОБОТА № 3

СТВОРЕННЯ ПАНЕЛІ ДІЙ І ОБЛАСТІ ЗАДАЧ У ПРОЕКТАХ OFFICE

3.1 Мета роботи

Навчитися створювати панелі дій і області задач у проектах Office.

3.2 Основні теоретичні відомості

3.2.1 Налаштування панелі дій документа або книги

Панель дій – це налаштовувана область задач «Дії з документами», що прикріплюється до окремого документу Microsoft Office Word або книги Microsoft Office Excel. Вона розміщується всередині області задач Office разом з іншими областями завдань, такими як Джерело XML в Excel або Стили та форматування в Word. Для розробки інтерфейсу панелі дій можна використовувати елементи керування Windows Forms або WPF.

Панель дій можна створити тільки в налаштуванні рівня документа для Word або Excel. Створити панель дій в надбудові рівня застосування неможливо.

Панель дій відрізняється від налаштовуваних областей задач. Налаштовувана область задач пов'язана з застосуванням, а не з окремим документом. Для деяких додатків Microsoft Office можна створювати власні області задач у надбудові рівня застосування.

Панель дій представлена класом ActionsPane. Щоб відобразити панель дій, слід додати елемент керування Windows Forms у властивість Controls поля ActionsPane. Після відображення панелі дій можна динамічно додавати або видаляти елементи керування у відповідь на дії користувача. Зазвичай для відображення панелі дій слід додати код в обробник подій Startup класу ThisDocument або ThisWorkbook, так що панель дій стає видимою, коли користувач відкриває документ вперше.

При додаванні декількох елементів на панель дій в більшості випадків необхідно об'єднати елементи керування в елемент керування користувача, а потім додати елемент керування

користувача у властивість Controls. Дана процедура включає наступні кроки:

- створення інтерфейсу користувача в панелі дій за допомогою додавання в проект елемента «Елемент керування панелі дій» або елемент керування користувача, які розширюють клас Windows Forms UserControl;

- додавання елемента керування Windows Forms в елемент UserControl за допомогою конструктора або написання коду;

- додавання екземпляра настроюваного елемента керування користувача в елементи керування, які містяться в полі ActionsPane класу ThisWorkbook (для Excel) або ThisDocument (для Word).

Хоча клас ActionsPane містить метод Hide і властивість Visible, не можна видалити панель дій з користувацького інтерфейсу за допомогою членів класу ActionsPane. Оскільки це приховує тільки елементи керування в панелі дій, але не саму панель.

Для того щоб приховати панель завдань, у рішенні Word використовується властивість Visible об'єкта TaskPane, а Excel – властивість DisplayDocumentActionTaskPane об'єкта Application.

Якщо користувач зберігає документ, коли панель дій видима, то панель видно при кожному відкритті документа незалежно від того, чи містить вона які-небудь елементи керування. Якщо потрібно керувати відображенням панелі дій, слід викликати метод Clear поля ActionsPane в обробнику подій Startup класу ThisDocument або ThisWorkbook, щоб переконатися, що панель дій невидима при відкритті документа.

Можна додати декілька елементів керування на панель дій і написати код, який відповідає на події документа, відображаючи і приховуючи елементи керування користувача. Якщо зіставити елементи схеми XML з документом, можна відобразити певні елементи керування користувача в панелі дій, якщо точка вставки знаходиться всередині одного з XML-елементів.

Елементи керування в панелі дій мають такі ж можливості прив'язки даних, як і елементи керування в Windows Forms. Можна виконувати прив'язку елементів керування до наборів даних, типізованих наборів даних і XML.

Можна виконати прив'язку елементів керування в панелі дій і елементів керування в документі до одного набору даних. Наприклад,

можна створити відношення "основний / подробиці" між елементами керування в панелі дій та елементами на аркуші.

При відображенні вікна повідомлення в обробнику подій Validating елемента керування в панелі дій подію може бути викликано вдруге при переміщенні фокуса від елемента керування на вікно повідомлення. Щоб цьому запобігти слід використовувати елемент керування ErrorProvider для відображення повідомлень про помилку.

Якщо користувач змінює розмір панелі дій під час виконання, розмір елементів керування також може змінюватися разом з панеллю дій. Можна використовувати властивість Anchor елемента керування Windows Forms, щоб закріпити елементи керування в панелі дій. Можна закріпити елементи керування Windows Forms на призначеному для користувача елементі точно так само, як і інші елементи керування.

Напряму змінити розмір ActionsPane не можна, тому що клас ActionsPane вбудований в область завдань. Однак можна програмними засобами змінити ширину області задач, задавши значення властивості Width класу CommandBar, що представляє область завдань. Можна змінити висоту області задач, якщо вона закріплена горизонтально або є переміщуваною. Не рекомендовано змінювати розмір області задач програмними засобами, тому що користувач повинен мати можливість вибирати розмір області, який найкращим чином відповідає його потребам.

Щоб додати панель дій в документ Microsoft Office Word, спочатку створіть елемент керування Windows Forms. Потім додайте елемент керування користувача у властивість Controls поля ThisDocument.ActionsPane в проєкті.

3.2.2 Створення налаштовуваної області задач

Області задач – це області користувацького інтерфейсу, які зазвичай прикріплюються до однієї сторони вікна програми Microsoft Office. За допомогою налаштовуваних областей задач можна створити власну область завдань і надати користувачам знайомий інтерфейс для доступу до функціональних можливостей рішення. Наприклад, інтерфейс може містити елементи керування, які виконують код для зміни документів або відображення даних з джерела.

До переваг налаштовуваної області задач належить те, що вона дозволяє інтегрувати функції в знайомий користувальницький інтерфейс, який дозволяє використовувати конструктор Windows Forms, а інструменти Visual Studio дозволяють швидко створювати персоналізовану область завдань.

Створити основну персоналізовану область завдань можна в два етапи:

- створити користувальницький інтерфейс для налаштовуваної області задач, додавши елементи керування Windows Forms до об'єкта UserControl;

- створити персоналізовану область завдань шляхом передачі елементу керування користувача об'єкту CustomTaskPaneCollection в надбудові. Ця колекція повертає новий об'єкт CustomTaskPane, який може використовуватися для зміни зовнішнього вигляду області задач і відповіді на користувальницькі події.

Елемент керування користувача можна створювати під час розробки або під час виконання. Якщо користувацький елемент керування створюється під час розробки, для створення користувацького інтерфейсу для області задач може використовуватися конструктор Windows Forms.

Після створення елементу керування користувача, що включає користувальницький інтерфейс області задач, що налаштовується, необхідно створити CustomTaskPane. Для цього необхідно передати елемент керування користувача CustomTaskPaneCollection в надбудові шляхом виклику методу Add.

Методи Add повертають новий об'єкт CustomTaskPane. Цей об'єкт може використовуватися для зміни зовнішнього вигляду області задач і відповіді на користувальницькі події.

Налаштовувані області задач у додатках пов'язані з вікном фрейму документа, що відображає документ або елемент. Область завдань видно тільки при відображенні відповідного вікна.

Для того щоб визначити, яке з вікон відображає персоналізовану область завдань, використовуйте відповідний метод Add, що завантажується при створенні області задач.

При необхідності автоматизації додатки з елементу керування користувача можна отримати прямий доступ до об'єктної моделі за допомогою Globals.ThisAddIn.Application.

Після створення області задач можна використовувати властивості та події об'єкта `CustomTaskPane`, щоб контролювати користувацький інтерфейс області задач і реагувати на її зміни.

Якщо налаштовувана область задач створюється в надбудові, необхідно також створити елемент користувацького інтерфейсу, щоб користувачі могли з його допомогою відобразити або приховати персоналізовану область завдань.

При створенні налаштовуваної області задач у застосуванні Microsoft Office, яке не підтримує налаштування стрічки, можна додати елемент `CommandBarButton`, що відображує або приховує персоналізовану область задач.

Керувати розміром або розташуванням налаштовуваної області задач можна за допомогою властивостей об'єкта `CustomTaskPane`. Зокрема щоб змінити розташування налаштовуваної області задач, використовуйте `DockPosition`, а для того щоб запобігти змінам розташування області задач – `DockPositionRestrict`. Інші зміни у зовнішній вигляд налаштовуваної області задач можна вносити за допомогою властивості об'єкта `UserControl`.

За допомогою відповідних параметрів можна активувати відповідь надбудови на зміни в налаштовуваній області задач. Наприклад: `DockPositionChanged` та `VisibleChanged`.

Після створення налаштовуваної області задач об'єкт `CustomTaskPane` зберігається в пам'яті на час виконання надбудови. Об'єкт буде зберігатися в пам'яті і при закритті області задач.

3.3 Порядок виконання роботи

3.3.1 Ознайомитися з основними теоретичними відомостями та рекомендованою літературою за темою роботи.

3.3.2 Створити проект рівня документа Word.

3.3.3 Додати в проект елемент керування користувача.

3.3.4 Додати елементи керування в елемент керування користувача у відповідності з узгодженим з викладачем завданням.

3.3.5 Додати створений елемент керування користувача у панель дій.

3.3.6 Створити проект рівня застосування Excel.

3.3.7 Додати в проект елемент керування користувача.

3.3.8 Додати елементи керування в елемент керування

користувача у відповідності з узгодженим з викладачем завданням.

3.3.9 Додати створений елемент керування користувача у область задач.

3.3.10 Виконати тестування розробленого програмного забезпечення.

3.3.11 Оформити звіт з роботи.

3.3.12 Відповісти на контрольні питання.

3.4 Зміст звіту

3.4.1 Тема та мета роботи.

3.4.2 Короткі теоретичні відомості.

3.4.3 Завдання на лабораторну роботу.

3.4.4 Текст розробленого програмного забезпечення з коментарями.

3.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

3.4.6 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

3.5 Контрольні запитання

3.5.1 Що таке панель дій?

3.5.2 Які можливості надає панель дій?

3.5.3 Чи можна створити панель дій у проекті рівня документа? У проекті рівня застосування?

3.5.4 Що таке елемент керування користувача?

3.5.5 Як додати у проект елемент керування користувача?

3.5.6 Як додати елемент керування користувача в панель дій?

3.5.7 Що таке налаштовувана область задач?

3.5.8 Які можливості надає налаштовувана область задач?

3.5.9 Чи можна створити налаштовувану область задач у проекті рівня документа? У проекті рівня застосування?

3.5.10 Як додати елемент керування користувача в налаштовувану область задач?

3.5.11 Чим панель дій відрізняється від налаштовуваної області задач?

4 ЛАБОРАТОРНА РОБОТА № 4

ВИКОРИСТАННЯ ЕЛЕМЕНТІВ КЕРУВАННЯ ВЕДУЧОГО ЗАСТОСУВАННЯ У ПРОЕКТАХ WORD

4.1 Мета роботи

Навчитися використовувати елементи керування ведучого застосування у проектах Word.

4.2 Основні теоретичні відомості

4.2.1 Ведучий елемент документа Word

Ведучий елемент Document є типом, який розширює тип Microsoft.Office.Interop.Word.Document і містить додаткові події, виконує функцію контейнера для ведучих елементів керування і елементів керування Windows Forms.

У проектах рівня документа існує ведучий елемент Document, який представляє документ у проекті. У проектах рівня застосування можна створювати ведучі елементи Document під час виконання.

Для звернення до документа в проекті використовується клас ThisDocument. При створенні проекту рівня документа Visual Studio створює клас ThisDocument, який використовується для зв'язку між Word і кодом налаштування. Клас ThisDocument забезпечує доступ до членів класу ведучого елементу Document і призначений для виконання основних завдань з налаштування, наприклад, для запуску коду при відкритті або закритті документа. Клас ThisDocument надає місце розташування, з якого можна почати написання коду в проекті.

Проект рівня документа може містити лише один ведучий елемент Document (клас ThisDocument). Під час розробки не допускається додавання нових ведучих елементів класу Document у проект. Також не допускається створення нових ведучих елементів класу Document під час виконання з налаштування рівня документа.

При створенні нового документа Word під час виконання йому присвоюється тип Microsoft.Office.Interop.Word.Document. Оскільки такий документ не є ведучим елементом, він не може містити елементи керування ведучого застосування або Windows Forms.

У проектах рівня застосування можна створювати ведучий

елемент Document під час виконання для будь-якого документа, відкритого у Word. Можна використовувати ведучий елемент Document для додавання елементів керування у зв'язаний документ або для обробки подій, недоступних в об'єктах Microsoft.Office.Interop.Word.Document. Щоб створити ведучий елемент Document, використовуйте метод GetVstoObject.

4.2.2 Елементи керування вмістом у документах Word

Елементи керування вмістом дозволяють розробляти документи та шаблони, які мають наступні функції:

- інтерфейс користувача з керованим введенням, подібним введенню, яке використовується у формах;
- обмеження, що забороняють користувачам редагувати захищені розділи документа або шаблону;
- прив'язка даних до джерела даних.

У проектах рівня документа можна додавати елементи керування вмістом у документ під час розробки або під час виконання. У проектах рівня застосування елементи керування вмістом додаються в будь-який відкритий елемент під час виконання. При додаванні елемента керування вмістом в документ, він визначається межами, заголовком і тимчасовим текстом, який може надавати інструкції.

До типів елементів керування вмістом, які можна додавати в документи, належать:

- колекція стандартних блоків: частини вмісту, створені для багаторазового використання (загальний титульний аркуш, відформатована таблиця);
- CheckBox: не надається спеціальний тип, створити можна за допомогою універсального шаблону;
- ComboBox (поле зі списком), що на відміну від списку, який розкривається, дозволяє користувачам додавати власні елементи;
- керуючий елемент вибору дати, що надає календар;
- список, що розкривається, на відміну від поля з таким списком, не дозволяє користувачам додавати або змінювати елементи;
- елемент керування групою визначає захищену область документа, яку користувачі не зможуть змінювати або видаляти, і може містити будь-які елементи документа;
- рисунок;

- текст у форматі RTF містить текст або інші елементи, такі як таблиці, зображення або інші елементи керування вмістом;

- звичайний текст: весь текст однаково відформатований;

- універсальний шаблон елемента керування вмістом ContentControl, який може представляти будь-який з доступних типів елементів керування вмістом, при цьому можна змінити поведінку об'єкта так, щоб вона була схожа на поведінку іншого типу елемента керування, за допомогою властивості Type. Об'єкти ContentControl можна створити тільки під час виконання, а не під час розробки.

До загальних функцій елементів керування вмістом належать:

- Text – текст, що відображається в елементі керування;

- PlaceholderText (окрім PictureContentControl) – тимчасовий текст, що відображається в елементі керування, поки користувач не змінить елемент керування, або він не буде заповнений даними з джерела даних або вміст елемента керування не буде видалено;

- Title – заголовок, який відображається на границі елемента керування вмістом, при натисканні на нього користувачем;

- події Entering, Exiting призводять до виконання коду, коли користувач натискає мишею всередині (за межами) елемента керування вмістом, або за програмного переміщення курсору;

- події Added, Deleting призводять до виконання коду після додавання в документ (безпосередньо перед видаленням) елемента керування вмістом в результаті операції скасування або повтору.

При захисті частини документа користувачам забороняється змінювати або видаляти вміст в цій частині документа. Якщо область, яку слід захистити, знаходиться всередині елемента керування, можна використовувати властивості елемента керування вмістом, щоб заборонити користувачам правку або видалення елемента керування: властивість LockContents забороняє правку вмісту користувачами, а властивість LockContentControl забороняє видалення елемента керування користувачами.

Якщо область, яку слід захистити, знаходиться поза елементом керування, або, якщо слід захистити область, яка містить елементи керування вмістом і інші типи вмісту, можна помістити всю область в GroupContentControl. Якщо даний елемент містить впроваджені елементи керування вмістом, останні не захищаються автоматично. Необхідно використовувати властивість LockContents кожного впровадженого елемента керування, щоб заборонити користувачам

правити їх вміст.

Елементи керування вмістом можна прив'язувати до полів баз даних або керованих об'єктів за допомогою однакових моделей прив'язки даних, таких як Windows Forms, а також їх можна зв'язати з елементами в XML-частинах, які впроваджено в документ.

Існує декілька способів додавання елементів керування вмістом в документ у проєкті рівня документа під час розробки: з вкладки Елементи керування Word в панелі елементів, як звичайний елемент керування вмістом Word, або перетягнути елемент керування вмістом у документ з вікна Джерела даних.

Можна програмно додати елемент керування вмістом в документ під час виконання за допомогою методів властивості Controls класу ThisDocument. Кожен метод має три переваження, що дозволяє додати елемент керування в місце поточного виділення, в зазначений діапазон, а також додати елемент керування, заснований на вихідному елементі керування вмістом в документі.

Динамічно створені елементи керування вмістом не зберігаються в документі при його закритті. Однак, власні елементи керування вмістом залишаються в документі. Можна заново створити елемент керування вмістом, заснований на власному елементі керування вмістом при наступному відкритті документа.

Для додавання елемента керування вмістом в місце поточного виділення (у вказаний діапазон) використовуйте метод ControlCollection, який має ім'я Add <класу елементів керування> і єдиний параметр для імені нового елемента керування (Microsoft.Office.Interop.Word.Range). Для додавання елемента керування вмістом, заснованого на власному елементі керування вмістом, використовуйте метод ControlCollection, який має ім'я Add <класу елементів керування> і параметр Microsoft.Office.Interop.Word.ContentControl.

4.2.3 Закладки у документах Word

Елемент керування Microsoft.Office.Tools.Word.Bookmark є закладкою з унікальним ім'ям, подіями і можливістю прив'язки до даних. Закладку можна використовувати в якості прототипу елемента або розміщення в документі. Даний елемент керування є поєднанням об'єктів Microsoft.Office.Interop.Word.Bookmark і Microsoft.Office.

Interop.Word.Range.

У проектах рівня документа можна додавати закладки в документ під час розробки або під час виконання. У проектах рівня застосування закладки додаються в будь-який відкритий елемент під час виконання.

Елемент керування Bookmark підтримує просту прив'язку даних. Закладку слід прив'язати до джерела даних за допомогою властивості DataBindings. Властивістю прив'язки даних закладки за замовчанням є Text. Зміни даних у зв'язаному наборі даних відображаються в елементі керування Microsoft.Office.Tools.Word.Bookmark. У проектах рівня документа прив'язати дані до закладок можна також за допомогою вікна Джерела даних.

Форматування, що застосовується до елемента Microsoft.Office.Interop.Word.Bookmark, можна застосовувати і до елементу керування Microsoft.Office.Tools.Word.Bookmark. До таких функцій форматування відносяться шрифти, відступи, інтервали, нумерація списку, а також стилі.

Додаткову відмінність між об'єктом Microsoft.Office.Interop.Word.Bookmark і елементом керування Microsoft.Office.Tools.Word.Bookmark складає їх поведінка при присвоєнні тексту закладці. Якщо першому присвоюється текст нульової довжини, він буде вказаний праворуч від закладки, і довжина закладки залишиться нульовою. Однак, якщо присвоюється текст нульової довжини другому об'єкту, його буде додано до закладки, а довжина закладки збільшиться відповідно до загальної кількості доданих знаків.

Властивість Bookmark.Text використовується, щоб відобразити текст в межах закладки і залишити закладку в документі. Range.Text використовується, щоб відобразити текст в розташуванні закладки і автоматично видалити її.

Закладки можуть перекривати одна одну. При присвоєнні нового тексту одній з закладок, що перекриваються, вона буде містити тільки новий текст. Якщо закладка повністю заповнена в межах іншої закладки і необхідно змінити текст зовнішньої закладки, внутрішня закладка не видаляється. Однак внутрішня закладка стає порожньою і переміщається в кінець зовнішньої закладки.

Для елемента керування Microsoft.Office.Tools.Word.Bookmark доступні наступні події: BeforeDoubleClick, BeforeRightClick, BindingContextChanged, Deselected, Disposed, Selected,

SelectionChange.

Існує декілька способів додавання елементів керування Bookmark у документ у проєкті рівня документа під час розробки: з Панелі елементів Visual Studio, з додатку Word тим же самим способом, що і звичайну закладку, з вікна Джерела даних.

Можна програмно додати в документ елемент керування Bookmark під час виконання за допомогою методів властивості Controls класу ThisDocument в проєкті. Кожен метод має два переважання, що дозволяє додавати елемент керування Bookmark такими способами: додати Bookmark в заданому діапазоні та елемент керування Bookmark, заснований на власній закладці в документі.

Елементи керування Bookmark можна додавати в будь-який відкритий документ під час виконання програмно або з використанням надбудови рівня застосування. Для цього слід створити ведучий елемент Document, заснований на відкритому документі, і використовувати метод властивості Controls цього ведучого елементу. Кожен метод має також два переважання.

Для додавання елемента керування Bookmark у вказаний діапазон слід використовувати метод ControlCollection.AddBookmark (Range, String) і передати в діапазон Range, куди необхідно додати елемент керування Bookmark.

Для додавання елемента керування Bookmark, заснованого на власному елементі керування Bookmark, слід використовувати метод ControlCollection.AddBookmark (Bookmark, String) і передати в існуючий елемент керування Microsoft.Office.Interop.Word.Bookmark, який буде використовуватися в якості основи для нового елемента керування Bookmark.

Розмір елемента керування Bookmark встановлюється при додаванні його в документ Microsoft Office Word. Передбачено три способи зміни розмірів закладки:

- додавання або видалення тексту в елементі керування Bookmark: Розмір автоматично змінюється;
- зміна властивостей Start і End елемента керування Bookmark;
- повторне створення елемента керування Bookmark;

4.3 Порядок виконання роботи

4.3.1 Ознайомитися з рекомендованою літературою за темою

роботи.

4.3.2 Зробити шаблон документу (наприклад, анкети) у відповідності з узгодженим з викладачем завданням, використовуючи для захисту частин документа елементи керування вмістом. Додавання елементів керування вмістом і їх налаштування виконувати під час розробки.

4.3.3 Створити проект рівня документа Word.

4.3.4 Додати в проект елемент «Стрічка (візуальний конструктор)».

4.3.5 Додати на стрічку вкладки користувача, використовуючи своє прізвище для заголовку вкладки.

4.3.6 Додати на вкладку користувача дві групи: «Елементи керування вмістом» і «Закладки».

4.3.7 У групу «Елементи керування вмістом» додати елементи керування і призначити обробники їх подій для роботи з елементами керування вмістом. Аналогічно — для групи «Закладки».

4.3.8 Додати в проект два елемента керування користувача `ContentControlsActionsPaneControl` і `BookmarksActionsPaneControl`. Додати елементи керування в елементи керування користувача і призначити обробники їх подій так, щоб це дозволило виконувати сценарії роботи з елементами керування вмістом та закладками відповідно, як це було зроблено для груп вкладки стрічки.

4.3.9 Додати кнопки запуску діалогового вікна в групи вкладки стрічки.

4.3.10 Призначити обробники подій для кнопок запуску діалогового вікна груп таким чином, щоб натискання цих кнопок призводило до відображення відповідного елемента керування користувача у панелі дій.

4.3.11 Продемонструвати додавання елемента керування вмістом у місце поточного виділення документа під час виконання.

4.3.12 Продемонструвати додавання елемента керування вмістом у місце поточного діапазону документа під час виконання.

4.3.13 Продемонструвати додавання елемента керування вмістом, заснованого на власному елементі керування вмістом в документ під час виконання.

4.3.14 Продемонструвати взяття під захист і скасування захисту від редагування елемента керування вмістом під час виконання.

4.3.15 Продемонструвати взяття під захист і скасування захисту

від вилучення елемента керування вмістом під час виконання.

4.3.16 Додати деякий вміст до колекції стандартних блоків.

4.3.17 Створити елемент керування вмістом, який відображає стандартний блок, створений у попередньому пункті.

4.3.18 Продемонструвати додавання елемента керування вмістом, який відображає стандартний блок, у документ під час виконання.

4.3.19 Продемонструвати додавання закладки в документ під час виконання.

4.3.20 Продемонструвати зміну розміру закладки шляхом додавання або видалення тексту під час виконання.

4.3.21 Продемонструвати зміну розміру закладки шляхом зміни її властивостей Start і End під час виконання.

4.3.22 Продемонструвати зміну розміру закладки шляхом повторного створення закладки під час виконання.

4.3.23 Створити контекстне меню для закладок і додати дії для його пунктів.

4.3.24 Виконати тестування розробленої програми.

4.3.25 Оформити звіт з роботи.

4.3.26 Відповісти на контрольні питання.

4.4 Зміст звіту

4.4.1 Тема та мета роботи.

4.4.2 Короткі теоретичні відомості.

4.4.3 Завдання на лабораторну роботу.

4.4.4 Текст розробленого програмного забезпечення з коментарями.

4.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

4.4.6 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

4.5 Контрольні запитання

4.5.1 Які існують ведучі елементи керування для Word?

4.5.2 Що таке ведучий елемент керування документа Word?

4.5.3 Які можливості надає ведучий елемент керування документа Word?

4.5.4 Як отримати доступ до ведучого елемента керування документа Word?

4.5.5 Які існують обмеження ведучого елемента документа Word у проектах рівня документа?

4.5.6 Що таке елемент керування вмістом?

4.5.7 Які можливості надає елемент керування вмістом?

4.5.8 Які існують типи елементів керування вмістом?

4.5.9 Загальні властивості елементів керування вмістом.

4.5.10 Як захистити документ за допомогою елементів керування вмістом?

4.5.11 Якими способами можна додавати елементи керування вмістом під час розробки?

4.5.12 Якими способами можна додавати елементи керування вмістом під час виконання у проектах рівня документа?

4.5.13 Якими способами можна додавати елементи керування вмістом під час виконання у проектах рівня застосування?

4.5.14 Що таке стандартні блоки?

4.5.15 Як додати деякий вміст у колекцію стандартних блоків?

4.5.16 Як створити елемент керування вмістом, що відображає стандартні блоки?

4.5.17 Що таке закладка?

4.5.18 Які відмінності між властивостями Bookmark.Text та Bookmark.Range.Text?

4.5.19 Які особливості роботи із закладками, котрі перекриваються?

4.5.20 Які події доступні для елемента керування «Закладка»?

4.5.21 Якими способами можна додавати закладки під час розробки?

4.5.22 Якими способами можна додавати закладки під час виконання у проектах рівня документа?

4.5.23 Якими способами можна додавати закладки під час виконання у проектах рівня застосування?

4.5.24 Якими способами можна змінити розмір закладки?

4.5.25 Як створити контекстне меню для закладок?

5 ЛАБОРАТОРНА РОБОТА №5

ВИКОРИСТАННЯ ЕЛЕМЕНТІВ КЕРУВАННЯ ВЕДУЧОГО ЗАСТОСУВАННЯ У ПРОЕКТАХ EXCEL

5.1 Мета роботи

Навчитися використовувати елементи керування ведучого застосування у проектах Excel.

5.2 Основні теоретичні відомості

5.2.1 Ведучі елементи книги, аркуша, аркуша діаграми

Ведучий елемент Workbook є типом, який розширює тип Microsoft.Office.Interop.Excel.Workbook з основної збірки взаємодії для Excel. Ведучий елемент Workbook надає ті ж властивості, методи і події, що і об'єкт Microsoft.Office.Interop.Excel.Workbook, однак має ряд додаткових можливостей.

У проектах рівня документа існує ведучий елемент Workbook, який представляє книгу в проекті. У проектах рівня застосування можна створювати ведучі елементи Workbook під час виконання.

Для звернення до книги в проекті використовується клас ThisWorkbook, що забезпечує доступ до членів класу ведучого елементу Workbook і призначений для виконання основних завдань з налаштування, наприклад, для запуску коду при відкритті або закритті книги. Клас ThisWorkbook надає місце розташування, в якому можна почати написання коду в проекті.

Двічі натиснувши на елемент проекту ThisWorkbook у браузері рішень, можна відкрити конструктор книг і відобразити властивості та події робочої книги у вікні Властивості.

Проект рівня документа може містити лише один ведучий елемент Workbook. Під час розробки не допускається додавання нових ведучих елементів класу Workbook в проект. Також не допускається створення нових ведучих елементів класу Workbook під час виконання з налаштування рівня документа.

При створенні нової книги Excel під час виконання присвоюється тип Microsoft.Office.Interop.Excel.Workbook. Оскільки такий документ не є ведучим елементом, він не може містити

елементи керування ведучої програми або Windows Forms.

Ведучий елемент Workbook не функціонує в якості контейнера для елементів керування ведучої програми. Таким чином, не можна додавати на аркуш будь-які видимі елементи керування, але можна додавати компоненти, такі як DataSet, оскільки ці компоненти можуть спільно використовуватися всіма аркушами. У проекті рівня документа компоненти, доступні для робочої книги, знаходяться на вкладках панелі елементів Компоненти, Дані та Всі форми Windows Forms.

У проектах рівня застосування можна створювати ведучий елемент Workbook під час виконання для будь-якої книги, відкритої в Excel. Щоб створити ведучий елемент Workbook, використовуйте метод GetVstoObject.

Ведучий елемент Worksheet є типом, який розширює тип Microsoft.Office.Interop.Excel.Worksheet з основної збірки взаємодії для Excel. Ведучий елемент Worksheet надає всі ті ж властивості, методи і події, що і об'єкт Microsoft.Office.Interop.Excel.Worksheet, проте також містить додаткові події і виконує функцію контейнера для ведучих елементів керування і елементів керування Windows Forms.

У проектах рівня документа ведучі елементи керування Worksheet можна додавати в документ під час розробки. У проектах рівня застосування можна створювати ведучі елементи Worksheet під час виконання.

При створенні проекту рівня документа для Excel Visual Studio автоматично створює три ведучих елемента Worksheet в проекті. За замовчуванням аркуші отримують імена Sheet1, Sheet2 і Sheet3.

Класи аркушів забезпечують доступ до учасників класу ведучого елемента Worksheet і призначені для виконання основних завдань з налаштування, наприклад, для зміни вмісту книги. Ці класи також можна використовувати для додавання в книгу елементів керування. Використовуючи різні поєднання елементів керування та коду, можна пов'язувати елементи керування з даними, збирати відомості, що вводяться користувачем, і реагувати на дії користувача.

У проект рівня документа можна додати додаткові ведучі елементи Worksheet під час розробки. Для цього необхідно додати новий аркуш книги в конструкторі.

У проекті рівня документа неможливе створення нових ведучих

елементів Worksheet. При створенні нового аркуша Excel під час виконання йому присвоюється тип `Microsoft.Office.Interop.Excel.Worksheet`. Оскільки такий документ не є ведучим елементом, він не може містити елементи керування ведучої програми або `Windows Forms`.

У проектах рівня застосування можна створювати ведучий елемент Worksheet під час виконання для будь-якого аркуша, відкритого в Excel. Ведучий елемент Worksheet можна використовувати для додавання елементів керування та смарт-тегів у зв'язаний аркуш або для обробки подій, недоступних в об'єктах `Microsoft.Office.Interop.Excel.Worksheet`.

Щоб створити ведучий елемент Worksheet, використовуйте метод `GetVstoObject`.

5.2.2 Елементи керування діаграмою

Елемент керування Chart – це об'єкт діаграми, що надає події. При додаванні діаграми на аркуш Visual Studio створює об'єкт Chart, який можна запрограмувати безпосередньо, не звертаючись до об'єктної моделі Microsoft Office Excel. Можна додати елементи керування Chart на аркуш Microsoft Office Excel під час розробки або під час виконання в проектах на рівні документа. У надбудові рівня застосування елементи керування Chart додаються на аркуш під час виконання.

Всі форматування, що застосовуються до елемента `Microsoft.Office.Interop.Excel.Chart`, можна також застосовувати і до елементу керування `Microsoft.Office.Tools.Excel.Chart`. Це включає межі, шрифти, тип діаграми, сітки, позначення та підписи даних.

Для елемента керування Chart доступні наступні події: `ActivateEvent`, `BeforeDoubleClick`, `BeforeRightClick`, `BindingContextChanged`, `Calculate`, `Deactivate`, `Disposed`, `DragOver`, `DragPlot`, `MouseDown`, `MouseMove`, `MouseUp`, `Resize`, `SelectEvent`, `SeriesChange`

Для додавання елемента керування діаграми на аркуш Excel на вкладці Вставлення у групі Діаграми натисніть на пункт Стовпець, виберіть категорію діаграми і потім потрібний тип діаграми, на вкладці Конструктор у групі Дані натисніть на пункт Вибір даних. Потім у діалоговому вікні виберіть діапазон комірок, який містить дані для діаграми.

5.2.3 Елементи керування ListObject

Елемент керування ListObject є списком з подіями і можливістю прив'язки до даних. При додаванні списку на аркуш Visual Studio створює елемент керування ListObject, який можна запрограмувати безпосередньо, не звертаючись до об'єктної моделі Microsoft Office Excel. У проектах рівня документа можна додавати елементи керування ListObject на аркуш під час розробки або під час виконання. У проектах рівня застосування елементи керування ListObject додаються на аркуші тільки під час виконання.

Елемент керування ListObject підтримує просту і складну прив'язку даних, його можна прив'язати до джерела даних за допомогою властивостей DataSource і DataMember під час розробки або за допомогою методу SetDataBinding під час виконання.

Якщо елемент керування ListObject прив'язаний до джерела даних, при зміні даних у якому події не виникають, то для оновлення елемента керування ListObject необхідно викликати метод RefreshDataRow або RefreshDataRows.

При додаванні в комірку аркуша елемента керування ListObject шляхом зіставлення з цією коміркою повторюваного елемента схеми Visual Studio автоматично зіставляє ListObject зі створеним набором даних. Однак елемент керування ListObject не прив'язується до даних автоматично. Можна прив'язати елемент керування ListObject до набору даних в проекті рівня документа під час розробки або під час виконання. Можна програмно виконати прив'язку елемента керування ListObject до набору даних в надбудові рівня застосування під час виконання.

Оскільки дані відокремлені від елемента керування ListObject, додавати і видаляти дані слід через прив'язаний набір даних, а не безпосередньо через елемент керування ListObject. Якщо дані в прив'язаному наборі оновлюються, зміни відображаються в елементі керування ListObject автоматично.

Якщо необхідно заповнити даними елемент керування ListObject і дозволити користувачеві змінювати дані в елементі керування ListObject, не змінюючи при цьому джерело даних, викличте метод Disconnect, щоб відключити елемент керування ListObject від джерела даних.

Під час виконання не можна вручну додавати або видаляти

стовпці з прив'язаного до даних елемента керування ListObject. У прив'язаний до даних елемент керування ListObject можна вручну додавати і видаляти рядки за умови, що джерело даних дозволяє додавати нові рядки і не призначено тільки для читання. Можна написати код для таких подій, як BeforeAddDataBoundRow, який буде перевіряти дані.

Для елемента керування ListObject доступні наступні події: BeforeAddDataBoundRow, BeforeDoubleClick, BeforeRightClick, BindingContextChanged, Change, DataBindingFailure, DataMemberChanged, DataSourceChanged, Deselected, ErrorAddDataBoundRow, OriginalDataRestored, Selected, SelectedIndexChanged, SelectionChange.

Можна додати елементи керування ListObject на аркуш Microsoft Office Excel під час розробки і під час виконання в проектах на рівні документа. Також можна додати елементи керування ListObject під час виконання в проектах надбудов рівня застосування.

Існує декілька способів додавання елементів керування ListObject на аркуш в проекті рівня документа під час розробки: з Excel (на вкладці Вставлення у групі Таблиці натисніть кнопку Таблиця та виділіть клітинки, які необхідно включити до списку), з панелі елементів Visual Studio (зі вкладки Елементи керування Excel панелі елементів перетягніть елемент керування ListObject в клітинку на аркуш і виділіть діапазон клітинок) і з вікна Джерела даних (відкрийте вікно Джерела даних, створіть джерело даних для проекту, перетягніть таблицю з вікна Джерела даних на аркуш).

Можна додавати елементи керування ListObject динамічно під час виконання. Це дозволяє створювати елементи керування ведучої програми при виникненні певних подій.

У проектах рівня документа можна змінити розміри елементів керування ListObject під час розробки або під час виконання. У проектах рівня застосування розміри елементів керування ListObject можна змінювати під час виконання.

Можна змінити розмір елемента керування ListObject під час виконання в проекті рівня документа, викликавши метод Resize.

5.2.4 Елемент керування NamedRange

Елемент керування NamedRange є діапазоном з унікальним

ім'ям, подіями і можливістю прив'язки до даних.

У проектах на рівні документа можна додати елементи керування NamedRange на аркуш Microsoft Office Excel під час розробки або під час виконання. У надбудові рівня застосування елементи керування NamedRange додаються на аркуш під час виконання.

Елементи керування NamedRange можуть складатися тільки з діапазонів на окремих аркушах (не діапазонах) і не можуть мати відносних імен, які можна застосувати для всіх аркушів.

Елементи керування NamedRange підтримують тільки просту прив'язку даних і можуть бути прив'язані до джерела даних за допомогою властивостей DataBindings. Типовою властивістю прив'язки даних елемента керування NamedRange є Value2. Якщо дані в прив'язаному наборі оновлюються яким-небудь способом, зміни відображаються в елементі керування NamedRange.

Форматування, яке може застосовуватися до елемента Microsoft.Office.Interop.Excel.Range, може застосовуватися до елемента керування Microsoft.Office.Tools.Excel.NamedRange. Це включає границі, шрифти, числовий формат та стилі.

Для елемента керування NamedRange доступні наступні події: BeforeDoubleClick, BeforeRightClick, BindingContextChanged, Change, Deselected, Disposed, Selected, SelectionChange

Існує декілька способів додавання елементів керування NamedRange на аркуш в проекті рівня документа під час розробки: з Excel, з панелі елементів Visual Studio і з вікна Джерела даних.

Елементи керування NamedRange можна додавати на аркуш програмними засобами під час виконання. Це дозволяє створювати елементи керування ведучої програми при виникненні певних подій. Динамічно створені іменовані діапазони не зберігаються як ведучі елементи керування на аркуші при його закритті.

Елемент керування NamedRange можна програмними засобами додавати на будь-який відкритий аркуш в проекті рівня застосування:

```
private void AddNamedRange ()
{
    Microsoft.Office.Tools.Excel.NamedRange textInCell;
    Worksheet worksheet = Globals.Factory.GetVstoObject (
        Globals.ThisAddIn.Application.ActiveWorkbook.Worksheets[1]);
    Excel.Range cell = worksheet.Range["A1", missing];
```

```

textInCell = worksheet.Controls.AddNamedRange (cell,
"MyNamedRange");
textInCell.Value2 = "Hello World";
}

```

5.3 Порядок виконання роботи

5.3.1 Ознайомитися з рекомендованою літературою за темою роботи.

5.3.2 Створити проект рівня документа Excel.

5.3.3 Додати в проект елемент «Стрічка (візуальний конструктор)».

5.3.4 Додати на стрічку вкладки користувача, використовуючи своє прізвище для заголовку вкладки.

5.3.5 Додати на вкладку користувача три групи: «Chart», «ListObject» і «NamedRange», заповнюючи їх згідно з наступними пунктами завдання у відповідності з узгодженим з викладачем індивідуальним завданням.

5.3.6 У групу «Chart» додати елементи керування і назначити обробники їх подій так, щоб це дозволило виконувати сценарії демонстрації роботи з діаграмами. Аналогічно для групи «ListObject» — для демонстрації роботи з елементами ListObject та для групи «NamedRange» — для демонстрації роботи з елементами NamedRange.

5.3.7 Додати в проект три елемента керування користувача «ChartUserControl», «ListObjectUserControl» і «NamedRangeUserControl». Додати елементи керування в елементи керування користувача і призначити обробники їх подій так, щоб це дозволило виконувати сценарії демонстрації роботи з діаграмами, з елементами ListObject та з елементами NamedRange відповідно, як це було зроблено для груп вкладки стрічки.

5.3.8 Додати кнопки запуску діалогового вікна в групи вкладки стрічки.

5.3.9 Призначити обробники подій для кнопок запуску діалогового вікна груп таким чином, щоб натискання цих кнопок призводило до показу відповідного елемента керування користувача у панелі дій.

5.3.10 Виконати додавання діаграми на аркуш під час розробки.

5.3.11 Продемонструвати додавання діаграми на аркуш під час

виконання.

5.3.12 Виконати додавання аркуша в книгу під час розробки.

5.3.13 Продемонструвати додавання аркуша в книгу під час виконання.

5.3.14 Виконати додавання аркуша діаграми (ChartSheet) в книгу під час розробки.

5.3.15 Продемонструвати додавання аркуша діаграми (ChartSheet) в книгу під час виконання.

5.3.16 Виконати додавання елемента керування ListObject на аркуш під час розробки.

5.3.17 Продемонструвати додавання елемента керування ListObject на аркуш під час виконання у проекті рівня документа.

5.3.18 Виконати прив'язку елемента керування ListObject до джерела даних (колекції або таблиці) під час розробки у проекті рівня документа. Виконати зіставлення стовпців елементу керування ListObject із даними джерела. Виконувати перевірку даних, введених користувачем при додаванні рядка.

5.3.20 Виконати відключення елемента керування ListObject від джерела даних під час виконання.

5.3.21 Виконати додавання та видалення рядків та стовпців елемента керування ListObject під час виконання.

5.3.22 Виконати зміну розмірів елемента керування ListObject під час розробки.

5.3.23 Продемонструвати зміну розмірів елемента керування ListObject під час виконання.

5.3.24 Виконати додавання елемента керування NamedRange під час розробки.

5.3.25 Продемонструвати додавання елемента керування NamedRange під час виконання.

5.3.26 Виконати обробку подій елемента керування NamedRange.

5.3.27 Виконати тестування розробленої програми.

5.3.28 Оформити звіт з роботи.

5.3.29 Відповісти на контрольні питання.

5.4 Зміст звіту

5.4.1 Тема та мета роботи.

5.4.2 Короткі теоретичні відомості.

5.4.3 Завдання на лабораторну роботу.

5.4.4 Текст розробленого програмного забезпечення з коментарями.

5.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

5.4.6 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

5.5 Контрольні запитання

5.5.1 Які існують ведучі елементи керування для Excel?

5.5.2 Яким чином можна змінити коротке ім'я та програмне ім'я аркуша під час розробки?

5.5.3 Чим аркуш діаграми (ChartSheet) відрізняється від звичайного аркуша (Sheet)?

5.5.4 Які можливості надає елемент керування Chart?

5.5.5 Як додати діаграму на аркуш під час розробки у проекті рівня документа?

5.5.6 Що представляє собою елемент керування ListObject?

5.5.7 Які можливості надає елемент керування ListObject?

5.5.8 Як виконати прив'язку елемента керування ListObject до джерела даних під час розробки? Під час виконання?

5.5.9 Як додати елемент керування ListObject на аркуш під час розробки у проекті рівня документа? Під час виконання у проекті рівня документа? Під час виконання у проекті рівня застосування?

5.5.10 Як змінити розміри елемента керування ListObject під час розробки у проекті рівня документа? Під час виконання у проекті рівня документа? Під час виконання у проекті рівня застосування?

5.5.11 Що представляє собою елемент керування NamedRange?

5.5.12 Які можливості надає елемент керування NamedRange?

5.5.13 Як відрізняється прив'язка даних у елементів керування ListObject і NamedRange?

5.5.14 Як додати елемент керування NamedRange на аркуш під час розробки у проекті рівня документа? Під час виконання у проекті рівня документа? Під час виконання у проекті рівня застосування?

5.5.15 За яких умов виникає подія Change у елементі керування

NamedRange?

5.5.16 Які події надає елемент керування NamedRange?

6 ЛАБОРАТОРНА РОБОТА №6

ВИКОРИСТАННЯ ЕЛЕМЕНТІВ КЕРУВАННЯ WINDOWS FORMS У ДОКУМЕНТАХ OFFICE

6.1 Мета роботи

Навчитися використовувати елементи керування Windows Forms у документах Office.

6.2 Основні теоретичні відомості

6.2.1 Загальні відомості про ведучі елементи й елементи керування ведучої програми

Засоби розробки Office в Visual Studio можна використовувати для додавання елементів керування Windows Forms і елементів керування ведучої програми до документів Word і книг Excel. Ведучі елементи розширюють різні об'єкти користувацького інтерфейсу в об'єктних моделях Word і Excel.

При розробці проектів рівня застосування для Excel або Word можна програмним способом додавати такі елементи керування в будь-який відкритий документ або аркуш під час виконання.

У проектах рівня документа під час розробки в документи Word або аркуші Excel можна додавати ведучі елементи керування одним із таких способів: додати ведучі елементи керування в документ під час розробки таким же чином, як додається власний об'єкт, перетягнути ведучі елементи керування з Панелі елементів або з вікна Джерела даних у документи і аркуші.

Ведучі елементи і елементи керування ведучої програми є типами, які надають модель програмування для рішень Office, що створюються за допомогою засобів розробки Office в Visual Studio. Вони забезпечують можливість взаємодії з об'єктними моделями додатків, заснованими на COM за аналогією зі взаємодією з керованими об'єктами.

Для проектів Excel доступні наступні ведучі елементи: Chart, ListObject, NamedRange, XmlMappedRange, для проектів Word: Bookmark, XMLNode, XMLNodes та елементи керування вмістом.

Ведучі елементи керування, що додаються в документи Office,

функціонують як власні об'єкти Office, однак мають додаткові функції, що включають події та можливість прив'язки до даних. Наприклад, якщо потрібно перехоплювати події власного об'єкта `Microsoft.Office.Interop.Excel.Range` в Excel, необхідно спочатку обробити подію зміни аркуша, потім визначити, чи відбулася зміна в `Microsoft.Office.Interop.Excel.Range`. На противагу цьому, елемент керування ведучої програми `Microsoft.Office.Tools.Excel.NamedRange` має подію `Change`, яку може бути оброблено безпосередньо.

При додаванні в документ елемента керування ведучої програми під час проектування його не можна видалити програмними засобами під час виконання, оскільки при спробі наступного використання в коді елемента керування виникає виняток. Метод `Delete` елемента керування ведучої програми видаляє тільки елементи керування ведучої програми, додані в документ під час виконання.

Динамічно створені ведучі елементи керування можна також видалити за допомогою передачі імені елемента керування в метод `Remove` властивості `Worksheet.Controls` або `Document.Controls`.

Одним із способів розширення елементами керування ведучої програми об'єктів Office є додавання подій. Наприклад, об'єкт `Microsoft.Office.Interop.Excel.Range` в Excel і об'єкт `Microsoft.Office.Interop.Word.Bookmark` у Word не мають подій, але середовище виконання Visual Studio Tools for Office розширяє ці об'єкти, додаючи програмовані події. Можна мати доступ до цих подій і записувати код для реагування на них через сторінку властивостей.

Кожен ведучий елемент і елемент керування ведучої програми створюється, щоб працювати аналогічно власному об'єкту `Microsoft Office Word` або `Microsoft Office Excel` з додатковими функціями. Але існує ряд фундаментальних відмінностей між поведінкою під час виконання ведучих елементів і елементів керування ведучої програми з одного боку і власних об'єктів Office з іншого боку.

При програмному створенні або відкритті документа, книги або аркуша під час виконання з використанням об'єктної моделі Word або Excel елемент не є ведучим. Замість цього новий об'єкт є власним об'єктом Office. Наприклад, при використанні методу `Documents.Add` для створення нового документа Word під час виконання він буде власним об'єктом `Microsoft.Office.Interop.Word.Document`, а не `Microsoft.Office.Tools.Word.Document`.

У проектах рівня документа не можна створити ведучі елементи

під час виконання. Ведучі елементи можуть бути створені в проектах рівня документа тільки під час розробки. У проектах рівня застосування можна створити ведучі елементи час виконання.

Під час виконання можна програмними засобами додавати елементи керування ведучої програми на ведучий елемент `Microsoft.Office.Tools.Word.Document` або `Microsoft.Office.Tools.Excel.Worksheet`. Не можна додавати елементи керування ведучої програми у власний документ `Microsoft.Office.Interop.Word.Document` або аркуш `Microsoft.Office.Interop.Excel.Worksheet`.

Для кожного ведучого елемента і елемента керування ведучої програми існує власний об'єкт `Word` або `Excel`. Для доступу до основного об'єкта використовується властивість `InnerObject` ведучого елемента або елемента керування ведучої програми. Однак неможливо привести власний об'єкт `Office` до типу ведучого елемента або елемента керування ведучої програми.

У застосуванні `Word` неможливо передати елемент керування ведучої програми в метод або властивість, для якого потрібний власний об'єкт `Word` як обов'язковий параметр. Для повернення до основного власного об'єкту `Word` необхідно використовувати властивість `InnerObject` елемента керування ведучої програми.

Більшість методів і властивостей ведучих елементів повертають власні об'єкти `Office`, що лежать в основі ведучих елементів. Наприклад, властивість `Parent` елемента керування ведучої програми `NamedRange` в `Excel` повертає об'єкт `Microsoft.Office.Interop.Excel.Worksheet`, а не ведучий елемент `Microsoft.Office.Tools.Excel.Worksheet`.

Середовище виконання `Visual Studio Tools for Office` не надає окремих колекцій для кожного типу елементів керування ведучої програми. Замість цього використовується властивість `Controls` ведучого елемента для перебору всіх керованих елементів керування (як елементів керування ведучої програми або елементів керування `Windows Forms`) в документі або аркуші, а потім необхідно знайти елементи, які відповідають типу потрібного елемента керування ведучої програми.

Об'єктні моделі `Word` і `Excel` включають властивості, які надають колекції власних елементів керування для документів і аркушів. За допомогою цих властивостей не можна отримати доступ до керованих елементів керування. Наприклад, неможливо

перерахувати кожен Microsoft.Office.Tools.Word.Bookmark елемент керування ведучої програми за допомогою властивості Bookmarks об'єкта Microsoft.Office.Interop.Word.Document. Ці властивості включають в документ лише елементи керування Microsoft.Office.Interop.Word.Bookmark, вони не включають в документ елементи керування ведучої програми Microsoft.Office.Tools.Word.Bookmark.

6.2.2 Загальні відомості про використання елементів керування Windows Forms в документах Office

Елементи керування Windows Forms – об'єкти, з якими можуть взаємодіяти користувачі для введення даних або керування ними. Їх можна додати в документи і в налаштовувані елементи інтерфейсу користувача, включаючи панелі дій, налаштовувані області задач і форми Windows Forms. При розробці інтерфейсу слід враховувати наступні варіанти використання елементів керування Windows Forms:

- у документі (якщо необхідно відображати елементи керування весь час, якщо потрібно, щоб користувачі вводили дані прямо в документ);
- у панелі дій або в налаштовуваній області задач (якщо необхідно надавати користувачеві контекстно-залежну інформацію, якщо необхідно, щоб у документі з'являвся тільки результат, а не дані і елементи керування запиту);
- у формі Windows Forms (якщо необхідно управляти розмірами елементів інтерфейсу користувача, якщо необхідно, щоб користувачі не могли приховувати або видаляти елементи керування).

Розробник може додати елементи керування Windows Forms в документи Word і аркуші Excel під час виконання. Середовище виконання надає допоміжні методи для додавання найбільш поширених елементів керування Windows Forms, що дозволяють швидко додавати елементи керування в документи Office і отримувати доступ до об'єднаних функціональних можливостей Office і Windows Forms, реалізованих у цих елементах керування.

Для додавання елементів керування Windows Forms в документ у проєкті рівня документа *під час розробки* перетягніть в документ необхідний елемент керування зі вкладки Стандартні елементи

керування панелі елементів.

Можна додавати елементи керування Windows Forms в документ *під час виконання програмними засобами*. У Word слід використовувати методи властивості Controls класу ThisDocument. У Excel слід використовувати методи властивості Controls класу Sheet n.

Якщо елемент керування Windows Forms додається в документ під час розробки, то цей елемент керування не зберігається в документі після його закриття.

У проектах надбудов рівня застосування необхідно додати посилання на Microsoft.Office.Tools.Word.v4.0.Utilities.dll, перш ніж отримувати доступ до методів Add <клас_управління>.

Відмінності, які існують для функціональних можливостей елементів керування Windows Forms в документах Office:

- не можна здійснювати перехід між елементами керування в робочому аркуші Excel або документі Word;

- не можна використовувати елемент керування GroupBox з метою розміщення інших елементів у документ Office;

- елементи керування Windows Forms, що використовуються в документах, упаковуються в клас, який надається середовищу виконання, що забезпечує елементи додатковими функціональними можливостями, характерними для аркуша або документа;

- розмір і позиція елемента керування визначаються властивостями, які є складовими контейнера елемента керування ActiveX. Властивості елемента керування ActiveX приймають значення, відмінні від значень еквівалентних властивостей елемента Windows Forms. Властивості Top, Left, Height або Width задаються в пунктах, а не в точках;

- при додаванні елементів керування в потоковий макет необхідно враховувати, що елементи керування будуть зміщуватися разом зі вмістом відповідно до його змін;

- при виборі елемента керування події виникають в наступному порядку: Enter, GotFocus, а при скасуванні вибору елемента керування: Leave Validating Validated LostFocus;

- не можна використовувати конструктор елемента керування для додавання елемента керування до документа під час виконання. Слід використовувати допоміжні методи, передбачені в середовищі виконання Visual Studio Tools for Office. Наприклад, метод

AddButton, щоб додати кнопку в робочий аркуш.

Сторонні елементи керування повинні мати атрибут ComVisibleAttribute, що дорівнює значенню true, щоб ці елементи можна було використовувати у рішенні Office.

6.2.3 Додавання елементів керування в документи Office під час виконання

Елементи керування можна додавати в документи Microsoft Office Word і книги Microsoft Office Excel, а також видаляти з них як під час розробки, так і під час виконання. В останньому випадку вони називаються динамічними елементами керування. Спосіб звертання до колекцій елементів керування залежить від типу проекту, що розробляється:

- у проекті рівня документа для Excel використовується властивість Worksheet.Controls класів Sheet1, Sheet2 і Sheet3;

- у проекті рівня документа для Word використовується властивість Document.Controls класу ThisDocument;

- у проекті рівня застосування для Excel або Word використовується властивість Controls класів Microsoft.Office.Tools.Excel.Worksheet або Microsoft.Office.Tools.Word.Document, створених під час виконання.

Класи Microsoft.Office.Tools.Excel.ControlCollection та Microsoft.Office.Tools.Word.ControlCollection містять допоміжні методи, які можна використовувати для додавання елементів керування ведучої програми та спільних елементів керування Windows Forms в документи і на аркуші. У кожному методі використовується формат <клас елемента керування>.

За допомогою властивості Controls забезпечується ітерація за всіма елементами керування в документі, включаючи додані під час розробки.

Видалити динамічні елементи керування можна шляхом виклику методу Delete елемента керування або шляхом виклику методу Remove кожної колекції Controls. У разі програмного додавання елементів керування ведучої програми в документ необхідно присвоювати кожному з них унікальне ім'я і вказати, де цей елемент керування має бути доданий в документі.

Щоб додати елемент керування System.Windows.Forms.Control,

який не підтримується доступними допоміжними методами, слід використовувати один з методів `AddControl` об'єкта `Microsoft.Office.Tools.Excel.ControlCollection` у `Excel`. Метод `AddControl` повертає об'єкт `Microsoft.Office.Tools.Excel.ControlSite` (для `Excel`) або `Microsoft.Office.Tools.Word.ControlSite` (для `Word`), який визначає, як елемент керування взаємодіє з аркушем або документом.

Після додавання елемента керування на аркуш або в документ за допомогою методів `AddControl` з'являється два різних об'єкта елементів керування: об'єкт `System.Windows.Forms.Control`, що представляє елемент керування користувача, та об'єкт `ControlSite`, `OLEObject` або `OLEControl`, що представляє елемент керування після додавання на аркуш або в документ. Ці елементи керування спільно використовують багато властивостей і методів. Для звернення до властивостей та методів, що належить тільки для елемента керування користувача, використовуйте `System.Windows.Forms.Control`, а для звернення до загальних методів та властивостей елементів керування використовуйте об'єкти `OLEObject` або `OLEControl`.

Коли документ зберігається, а потім закривається, всі динамічні ведучі елементи керування `Windows Forms` видаляються, залишаються тільки власні базові об'єкти `Office`. Власні об'єкти `Office` не пов'язані з подіями ведучих елементів керування і не мають функції прив'язки даних, як у випадку з ведучими елементами керування. Динамічний ведучий елемент керування можна відтворювати на місці існуючого власного елемента керування при кожному відкритті документа. Щоб створити ведучий елемент керування `Microsoft.Office.Tools.Excel.ListObject` на основі існуючого власного об'єкта `Microsoft.Office.Interop.Excel.ListObject` при відкритті документа, слід використовувати метод `AddListObject (ListObject)` і передати йому існуючий елемент керування `Microsoft.Office.Interop.Excel.ListObject`.

Хоча метод `GetVstoObject` використовується в основному для створення нових ведучих елементів керування під час виконання, за допомогою нього можна також видалити всі оболонки `ActiveX` з документа при першому виклику методу для конкретного документа.

6.3 Порядок виконання роботи

6.3.1 Ознайомитися з рекомендованою літературою за темою роботи.

6.3.2 Створити проект рівня документа Word.

6.3.3 Додати в проект стрічку і виконати її налаштування за допомогою конструктора.

6.3.4 Створити елемент керування користувача.

6.3.5 Додати елемент керування користувача у панель дій документа.

6.3.6 Додати елементи керування на стрічку у вкладку користувача та в елемент керування користувача, які дозволять запускати сценарії демонстрації роботи з елементами керування Windows Forms у відповідності з завданням, узгодженим з викладачем.

6.3.7 Додати елементи керування Windows Forms у документ під час розробки.

6.3.8 Продемонструвати додавання елементів керування Windows Forms у документ під час виконання.

6.3.9 Виконати повторне створення динамічних ведучих елементів керування при відкритті документів.

6.3.10 Виконати видалення оболонок елементів керування ActiveX при закритті документа.

6.3.11 Налаштувати елемент керування таким чином, щоб він не змінював розмір і не переміщувався разом із коміркою у аркуші Excel (для проекту Excel).

6.3.12 Приховати елементи керування на аркушах під час друку.

6.3.13 Виконати тестування розробленої програми.

6.3.14 Оформити звіт з роботи.

6.3.15 Відповісти на контрольні питання.

6.4 Зміст звіту

6.4.1 Тема та мета роботи.

6.4.2 Короткі теоретичні відомості.

6.4.3 Завдання на лабораторну роботу.

6.4.4 Текст розробленого програмного забезпечення з коментарями.

6.4.5 Результати роботи програмного забезпечення (розроблені проекти, зображення форм інтерфейсу користувача).

6.4.6 Висновки, що містять відповіді на контрольні запитання, та відображують результати виконання роботи та їх критичний аналіз.

6.5 Контрольні запитання

6.5.1 Де можна розмістити елементи керування Windows Forms у документах Office?

6.5.2 Чому в процесі завантаження документа або книги можлива поява старих елементів керування Windows Forms?

6.5.3 Якими способами можна додати елементи керування Windows Forms у документ або книгу під час розробки у проекті рівня документа?

6.5.5 Як додати елемент керування Windows Forms у документ Office під час виконання у проекті рівня документа? У проекті рівня застосування?

6.5.6 Які обмеження у методів і властивостей елементів керування Windows Forms, які використовуються у документах Office?

6.5.7 Які особливості поведінки у елементів керування Windows Forms, які використовуються у документах Office?

6.5.8 Які елементи керування Windows Forms, що використовуються у документах Office, називаються статичним, а які динамічними?

6.5.9 У чому відмінність статичних та динамічних елементів керування Windows Forms, що використовуються у документах Office?

6.5.10 Чи можна видалити статичні елементи керування з документів Office?

6.5.11 Як виконати збереження динамічних елементів керування в документах Office?

6.5.12 Чому необхідно видаляти оболонки ActiveX елементів керування перед закриттям документа?

6.5.13 Які особливості елементів керування Windows Forms у аркушах Excel?

6.5.14 Як зробити, щоб елемент керування не змінював свій розмір і не переміщався разом з коміркою у аркуші Excel?

6.5.15 Як приховати елементи керування на аркушах під час друку?

7 ЛАБОРАТОРНА РОБОТА №7 РОБОТА З PDF-ДОКУМЕНТАМИ

7.1 Мета роботи

Навчитися працювати з PDF документами.

7.2 Основні теоретичні відомості

7.2.1 Загальний огляд бібліотек

7.2.1.1 Бібліотека iTextSharp

Бібліотека iTextSharp підходить для веб та прикладних додатків. Дозволяє динамічно генерувати та/або маніпулювати PDF документами.

Розробники можуть використовувати iText для того, щоб:

- передати PDF у браузер;
- генерувати динамічні документи з XML файлів або бази даних;
- використовувати інтерактивні можливості PDF;
- додавати закладки, кількість сторінок, водяні знаки тощо;
- розділяти, об'єднувати та маніпулювати PDF сторінками;
- автоматизувати заповнення PDF форм;
- додавати цифровий підпис до PDF файлу.

7.2.1.2 Бібліотека Report.NET

Бібліотека Report.NET – потужна бібліотека, яка просто та зручно дозволяє генерувати PDF документи. Документ може бути створений за допомогою даних, які були отримані з будь-якого ADO.NET data set.

Можливості бібліотеки:

- дуже компактний код;
- підтримує графічні об'єкти: текст, лінії, прямокутники, зображення jpeg;
- легкість вирівнювання та трансформації графічних об'єктів;
- ASP.NET може генерувати динамічні PDF сторінки;
- XML документація (Comment Web Pages).

Наступні бібліотеки мають схожу з описаними функціональність:

- PDFsharp;
- SharpPDF;
- PDFjet Open Source Edition (підтримує багату функціональність для малювання);
- ASP.NET FO PDF;
- PDF Clown.

7.2.2 Робота з бібліотекою iTextSharp

Для роботи з бібліотекою iTextSharp необхідно виконати наступні дії:

- завантажити останню версію бібліотеки за посиланням <http://sourceforge.net/projects/itextsharp/?source=navbar>, видобути архів itextsharp-dll-core.zip, видобути файл itextsharp.dll;
- створити проект Windows Forms;
- додати посилання на бібліотеку.

Для цього у вікні **Обозреватель решений** натисніть правою кнопкою миші по **Ссылки** та оберіть пункт **Добавить ссылку**.

У вікні, що з'явиться, оберіть вкладку **Обзор** та знайдіть попередньо завантажений файл.

Додайте кнопку на форму та текст обробника її натискання:

```
// створимо документ
var doc = new Document();
// відкриємо файловий потік. Створимо файл, якщо не існує
PdfWriter.GetInstance(doc, new FileStream
(Application.StartupPath + @"\Document.pdf",
FileMode.Create));
doc.Open();
// додаємо зображення, вирівнюємо по центру
iTextSharp.text.Image jpg =
iTextSharp.text.Image.GetInstance(Application.StartupPath +
@"\images.jpg");
jpg.Alignment = Element.ALIGN_CENTER;
doc.Add(jpg);
// створимо та заповнимо таблицю
PdfPTable table = new PdfPTable(3);
```

```

PdfPCell cell = new PdfPCell(new Phrase("Simple table",
new iTextSharp.text.Font
(iTextSharp.text.Font.FontFamily.TIMES_ROMAN, 16,
iTextSharp.text.Font.NORMAL, new BaseColor(Color.Orange))));
cell.BackgroundColor = new BaseColor(Color.Wheat);
cell.Padding = 5;
cell.Colspan = 3;
cell.HorizontalAlignment = Element.ALIGN_CENTER;
table.AddCell(cell);
table.AddCell("Col 1 Row 1");
table.AddCell("Col 2 Row 1");
table.AddCell("Col 3 Row 1");
table.AddCell("Col 1 Row 2");
table.AddCell("Col 2 Row 2");
table.AddCell("Col 3 Row 2");
jpg = iTextSharp.text.Image.GetInstance
(Application.StartupPath + @"/left.jpg");
cell = new PdfPCell(jpg);
cell.Padding = 5;
cell.HorizontalAlignment = PdfPCell.ALIGN_LEFT;
table.AddCell(cell);
cell = new PdfPCell(new Phrase("Col 2 Row 3"));
cell.VerticalAlignment = PdfPCell.ALIGN_MIDDLE;
cell.HorizontalAlignment = PdfPCell.ALIGN_CENTER;
table.AddCell(cell);
jpg = iTextSharp.text.Image.GetInstance
(Application.StartupPath + @"/right.jpg");
cell = new PdfPCell(jpg);
cell.Padding = 5;
cell.HorizontalAlignment = PdfPCell.ALIGN_RIGHT;
table.AddCell(cell);
doc.Close();
button1.Text = "Success";

```

Додаткові бібліотеки, необхідні для виконання програми:

```

using iTextSharp.text;
using System.IO;
using iTextSharp.text.pdf;

```

Додайте необхідні зображення до проекту. Це можуть бути будь-які 3 зображення. Помістіть їх до папки bin/Debug.

Запустіть проект та натисніть на кнопку. Після того, як напис на кнопці змінився на “Success”, у директорії, що відповідає конфігурації, з’явиться файл Document.pdf (рис. 7.1).

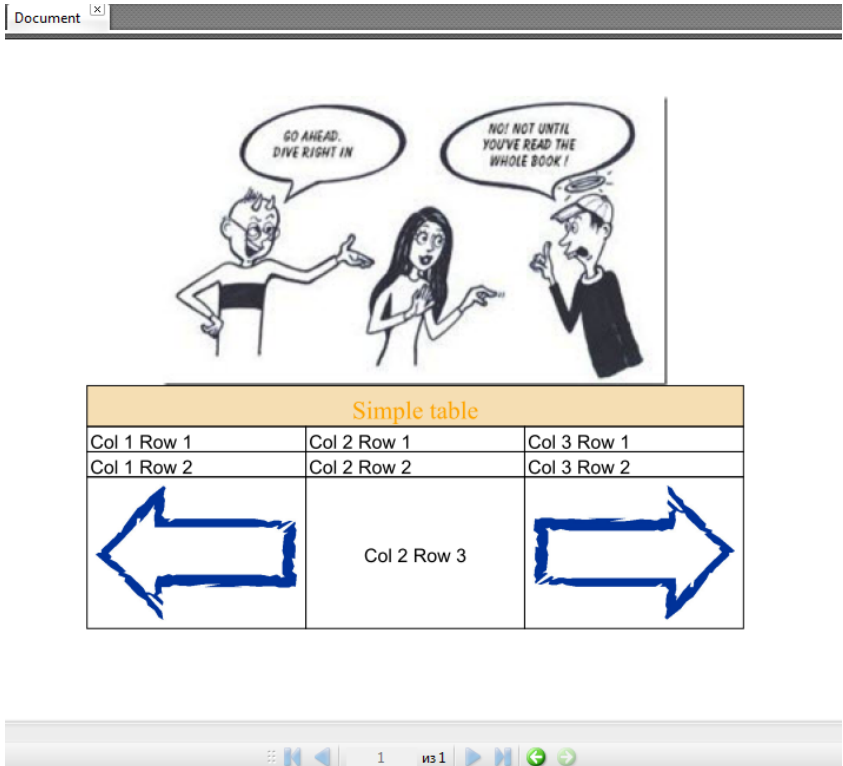


Рисунок 7.1 — Згенерований файл Document.pdf

7.3 Порядок виконання роботи

7.3.1 Ознайомитися з основними теоретичними відомостями та рекомендованою літературою за темою роботи.

7.3.2 Виконати індивідуальне завдання з додатку А (підрозділ А.3).

7.3.3 Виконати тестування розробленого застосування.

7.3.4 Оформити звіт з роботи.

7.3.5 Відповісти на контрольні запитання.

7.4 Зміст звіту

7.4.1 Тема та мета роботи.

7.4.2 Короткі теоретичні відомості.

7.4.3 Завдання на лабораторну роботу.

7.4.4 Текст розробленого програмного забезпечення з коментарями.

7.4.5 Результати роботи програмного забезпечення (зображення форми та PDF файлу).

7.4.6 Висновки, що містять відповіді на контрольні запитання, а також відображають результати виконання роботи та їх критичний аналіз.

7.5 Контрольні запитання

7.5.1 Які існують бібліотеки для роботи з PDF файлами у C#?

7.5.2 Як підключити dll файл до проекту?

7.5.3 Як додати зображення до PDF файлу?

7.5.4 Як додати таблицю до PDF файлу?

7.5.5 Яку функцію потрібно викликати, щоб записати документ на диск?

8 ЛАБОРАТОРНА РОБОТА № 8 СТВОРЕННЯ ДИСТРИБУТИВУ ДЛЯ WINDOWS- ЗАСТОСУВАНЬ

8.1 Мета роботи

8.1.1 Ознайомитися з основами створення дистрибутивів для розроблених застосунків у Visual Studio.

8.1.2 Навчитися створювати пов'язаний та не пов'язаний з проектом дистрибутив застосування у Visual Studio.

8.2 Основні теоретичні відомості

8.2.1 Загальні відомості про дистрибутиви

Створення дистрибутиву для Windows-застосунків – це перший етап процесу розгортання (розповсюдження готового застосунка на інші комп'ютери). У Visual Studio розгортання застосунків ґрунтується на використанні технології інсталяції Microsoft Windows, яка також включає засоби для створення виконуваних файлів.

Існує два шляхи створення даних проектів: створення зв'язаного з проектом та розділеного рішення проекту встановлення.

8.2.2 Створення дистрибутиву для Windows-застосунків у Visual Studio 2010

8.2.2.1 Створення пов'язаного проекту встановлення

Після створення Windows-застосунка (далі використано проект рішення з ім'ям Hello), створимо відповідні інсталяційні файли. Для цього, у контекстному меню проекту (верхній вузол у дереві Обозреватель решений) виберемо пункти Додати та Створити проект.

У лівій частині майстра Додати новий проект вибираємо папку Другі типи проектів / Установка и развертывание / Установщик Visual Studio, а у правій — Проект установки. Вказуємо назву проекту та його майбутнє місцезнаходження (назва не може співпадати з ім'ям проекту застосунка — проекти повинні мати різні назви). На даному етапі можна скористатися і Майстром встановлення, але при цьому втрачається гнучкість.

Серед властивостей проекту: AddRemoveProgramsIcon, Author, Description, DetectNewerInstalledVersion, Keywords, Localization, Manufacturer, ManufacturerUrl, ProductCode, ProductName, RemovePreviousVersions, SearchPath, Subject, SupportPhone, SupportUrl, Title, UpgradeCode, Version. У даному проекті встановимо: Manufacturer — zntu, Author — zntu, ProductName — SetupHello.

Властивість AddRemoveProgramsIcon дозволяє встановити іконку для опису програми у панелі Видалення програми. Для встановлення іконки треба вибрати у випадяючому списку Обзор, потім у вікні Выбор элемента в проекте двічі натиснути на Папка приложения та у наступному вікні Добавить файл.

Додамо вихідні файли застосування Hello до проекту встановлення SetupHello. Для цього у вікні Файловая система натисніть правою кнопкою миші на Папка приложения — Добавить — Выходной элемент проекта... або у меню оберіть Проект — Добавить — Выходной элемент проекта...

Далі у вікні Добавление выходной группы проекта обираємо проект Hello та Основные выходные файлы. Після підтвердження у директорії Папка приложения та у рішенні Hello з'явиться пункт, який означає, що у проект встановлення додані необхідні файли з проекту.

На даному етапі до проекту можна додавати файли, папки та збірки. Якщо, наприклад, програма працює з деякими зображеннями, які знаходяться у директорії рішення, то при розробці застосування ці зображення будуть знаходитись у Debug або Release директорії (в залежності від збірки), але інсталяційний проект не підхопить ці файли. У такому випадку потрібно додати до інсталяційного проекту ці зображення через пункт Добавить — Файл.

Тепер залишилося лише додати необхідні ярлики та іконки. Додамо посилання на папку, куди буде встановлене застосування (C:\Program Files\zntu\Hello), в меню Пуск/Програми та на робочий стіл. Розглянемо цей процес на прикладі додавання папки на робочий стіл. Для цього необхідно:

а) у властивостях вузла Рабочий стол пользователя встановити властивість AlwaysCreate у true. Це завжди необхідно, коли додаємо ярлик для спеціальної папки;

б) у контекстному меню вибрати Создать ярлык для Рабочий стол пользователя. У вікні властивостей ярлика натиснути на кнопку з трьома точками у властивості Target. У вікні, що з'явиться обрати

Папка приложення.

Назвіть ярлик, наприклад, за назвою програми. Якщо на даному етапі перебудувати рішення і встановити його, то на робочому столі з'явиться ярлик, але зі стандартною іконкою. Створимо іконки для посилань на робочому столі та у меню Пуск/Програми. Іконку для Встановлення та видалення програм вже було додано. Тепер у властивостях ярликів змініть властивість Icon – додайте іконки.

Перекомпілюйте проект (у контекстному меню Обозреватель решений вузла SetupHello пункт Перестроить). У результаті будуть створені файли setup.exe, SetupHello.msi.

Після встановлення програми запис про застосування буде додано у меню Пуск/Програми, до панелі Установлення та видалення програм, на робочому столі з'явиться ярлик.

8.2.2.2 Створення розділеного проекту встановлення

Попередній спосіб встановлення не дозволяє створити ярлик, який посилається на exe файл. Але це не найголовніша відмінність — попередній спосіб потребує збереження проекту застосування та проекту інсталяції разом, що незручно.

Створіть проект, аналогічний попередньому, Конфигурация решения встановіть у Release, налаштуйте властивості проекту.

Встановіть властивість Папка приложення AlwaysCreate у true, додайте ...\\Hello\\Hello\\bin\\Release\\Hello.exe до Папка приложення. Якщо проект використовує додаткові файли, додати потрібно їх усі. Даний етап аналогічний попередньому способу за однією відмінністю — способом додавання вихідних файлів. Іконки встановлюються також аналогічно першому способу.

Додавання ярликів на робочий стіл та у меню Пуск/Програми трохи відрізняється від першого способу. У контекстному меню файлу Hello.exe обираємо Создать ярлык для Hello.exe. Виберіть назви ярликів та перетягніть їх до потрібних папок.

Перебудуйте рішення. Після встановлення програми на робочому столі з'явиться ярлик.

8.3 Порядок виконання роботи

8.3.1 Ознайомитися з основними теоретичними відомостями та

рекомендованою літературою за темою роботи.

8.3.2 Для застосування з попередньої лабораторної роботи створити дистрибутив, пов'язаний з проектом.

8.3.3 Для застосування з попередньої лабораторної роботи створити дистрибутив, не пов'язаний з проектом.

8.3.4 Виконати тестування розроблених дистрибутивів.

8.3.5 Оформити звіт з роботи.

8.3.6 Відповісти на контрольні питання.

8.4 Зміст звіту

8.4.1 Тема та мета роботи.

8.4.2 Короткі теоретичні відомості.

8.4.3 Завдання на лабораторну роботу.

8.4.4 Результати роботи програмного забезпечення (зображення дистрибутивів та результатів установлення дистрибутивів).

8.4.5 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

8.5 Контрольні запитання

8.5.1 Що таке дистрибутив?

8.5.2 Які типи дистрибутивів можна створити за допомогою Visual Studio?

8.5.3 Чим відрізняються зв'язаний з проектом дистрибутив від не зв'язаного?

8.5.4 Як зробити, щоб при встановленні дистрибутиву на робочому столі з'являвся ярлик додатку?

8.5.5 Як додати файли до дистрибутиву?

8.5.6 Де можна змінити ім'я автора програми?

8.5.7 Які три іконки можна встановити для дистрибутиву?

8.5.8 Як визначити версію умови запуску .NET Framework для проекту?

8.5.9 Які властивості проекту можна визначити?

8.5.10 Чи можливо інсталиувати додаток з вилученням попередньої версії?

8.5.11 Як змінити шлях, за яким інсталиується додаток?

ЛІТЕРАТУРА

1. Разработка приложений Office в Visual Studio. – Режим доступа до ресурсу : [http://msdn.microsoft.com/ru-ru/library/d2tx7z6d\(v=vs.100\).aspx](http://msdn.microsoft.com/ru-ru/library/d2tx7z6d(v=vs.100).aspx)
2. iTextSharp. An open source C# PDF library. – Режим доступа до ресурсу: <http://sourceforge.net/projects/itextsharp/?source=navbar>
3. Практика программирования на C# для Windows и Web в Microsoft Visual Studio. – Режим доступа до ресурсу: http://wladm.narod.ru/C_Sharp/index.html
4. Лабор В.В. Си Шарп Создание приложений для Windows / В. В. Лабор. – Минск : Харвест, 2003. – 385 с.
5. Джонсон Б. Основы Microsoft Visual Studio .Net 2003 / Б. Джонсон. – Москва : Русская редакция, 2003. – 463 с.
6. Бишоп Д. C# в кратком изложении / Д. Бишоп. – Москва : БИНОМ Лаборатория знаний, 2005. – 467 с.
7. Фризен И. Офисное программирование / И. Фризен. – М. : Феникс, 2010. – 240 с.
8. Carter E. Visual Studio Tools for Office : Using C# with Excel, Word, Outlook, and InfoPath / E. Carter, E. Lippert. – Indianapolis : Addison-Wesley Professional, 2005. – 1008 p.
9. Назаров С. В. Программирование в пакетах MS Office / С. В. Назаров, П. П. Мельников, Л. П. Смольников. – М. : Финансы и статистика, 2007. – 656 с.
10. Гарнаев А. Ю. Microsoft Office Excel 2010 : разработка приложений / А. Ю. Гарнаев. – СПб : БХВ-Петербург, 2011. – 528 с.
11. Буллен С. Excel и VBA : Профессиональная разработка приложений Excel / С. Буллен, Р. Боуви, Д. Грин. – М. : Вильямс, 2007. – 736 с.
12. Walkenbach J. Excel 2013 : профессиональное программирование на VBA / John Walkenbach. – New Jersey : John Wiley & Sons, 2013. – 1104 p.
13. Слепцова Л.Д. Программирование на VBA в Microsoft Office 2010 / Л.Д. Слепцова. – М. : ООО “И. Д. Вильямс”, 2010. – 432 с. : ил.

ДОДАТОК А

ПРИКЛАДИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ

А.1 Створення проектів Microsoft Word

1. В абзацах зі стилем «Код» ключові слова мови програмування C# виділити напівжирним шрифтом синього кольору.
2. В абзацах зі стилем «Код» виконати заміну символів табуляції на початку кожного рядка символами пропуску.
3. У визначеному фрагменті тексту замінити числа на відповідні символи (за кодом).
4. В абзацах зі стилем «Код» оформити коментарі курсивом, а розмір шрифту всього іншого тексту збільшити на 2 пт.
5. Замінити всі послідовності символів пробіл на одинарний пробіл.
6. У визначеному фрагменті тексту підняти всі заголовки на один рівень.
7. У визначеному фрагменті тексту видалити всі гіперпосилання, залишивши лише їхній текст та оформивши його курсивом.
8. У визначеному фрагменті тексту видалити весь текст, залишивши лише абзаци зі стилями «Заголовок 1», ..., «Заголовок 9».
9. У всіх абзацах зі стилем «Звичайний» збільшити розмір шрифту великих літер на 2 пт, збільшити міжрядковий інтервал в абзацах.
10. Вилучити всі знаки переносу з тексту. Внесені зміни виділити кольором.
11. Замінити в абзацах зі стилем «Звичайний» символи англійської розкладки на українську.
12. Визначити кількість символів в абзацах зі стилем «Код» та кількість символів у всіх інших абзацах.
13. Вивести на окрему сторінку весь текст зі стилями «Заголовок 1», «Заголовок 2», «Заголовок 3».
14. Виконати заміну деякого набору слів на інший у визначеному фрагменті тексту.
15. У визначеному фрагменті тексту видалити текст всіх стилів окрім деякого заданого.

A.2 Створення проектів Microsoft Excel

1. Заповнити таблицю результатами контролю успішності студентів та підрахувати загальний бал.
2. Заповнити таблицю результатами продаж товарів за місяцями та виділити кольором клітинки з найгіршими результатами за кожен місяць.
3. Створити таблицю, в якій кожен третій рядок виділено кольором та відформатувати її границі.
4. Заповнити таблицю результатами контролю успішності студентів та виділити студентів, які мають поточний результат менше 60 балів.
5. Заповнити таблицю результатами голосування за регіонами за кожним з кандидатів та визначити середній результат.
6. Заповнити таблицю результатами футбольних матчів, розфарбувавши діагональні елементи різними кольорами.
7. Створити дві таблиці та розрахувати елементи третьої як різницю між їх елементами.
8. Заповнити таблицю результатами голосування за регіонами за кожним з кандидатів та виділити кольором регіони, в яких переміг перший кандидат.
9. Заповнити таблицю даними та розрахувати діагональні елементи як суму всіх елементів рядка та стовпця, на перетині яких вони знаходяться.
10. Заповнити таблицю результатами відвідування занять студентами та розрахувати кількість пропусків.
11. Створити таблиці, заповнивши їх даними про продаж товарів за місяцями за декількома магазинами. Обчислити загальні результати продажів та виділити кольорами найгірший та найкращий магазини.
12. Заповнити таблицю результатами футбольних матчів, виділивши кольором переможців та переможених у кожній грі.
13. Заповнити таблицю результатами контролю успішності студентів та визначити студентів, що не склали сесію.
14. Заповнити таблицю даними таким чином: перший рядок та стовпчик – довільним чином, а всі інші елементи розрахувати як суму елементів, що розташовані зліва та згори.
15. Створити таблицю, в якій кожен кожен стовпчик виділити

кольором у залежності від суми елементів у даному стовпчику.

A.3 Робота з PDF-документами

1. Розробіть програму, яка додає в файл зображення, обрізане за допомогою деякої геометричної фігури (передбачити декілька варіантів).

2. Розробіть програму, яка додає до документу зображення, встановлюючи його рівень стиснення.

3. Розробіть програму, яка створює кольорову палітру в документі.

4. Розробіть програму, яка з'єднає два PDF-документи у один, додаючи інформацію про з'єднання у документ.

5. Розробіть програму, яка заповнює документ зображеннями декількох форматів.

6. Розробіть програму, яка в деяких комірках таблиці розміщує інші таблиці.

7. Розробіть програму, яка розміщує в документі таблицю з комірками, які мають різний колір тла.

8. Розробіть програму, яка додає до документу посилання, яке призводить до відкриття нового файлу в текстовому редакторі.

9. Розробіть програму, яка створює захищений паролем документ, додає захист до існуючого документу, встановлюючи пароль та забороняючи друк.

10. Розробіть програму, яка створює документ, заповнюючи його метадані, та змінює метадані існуючого документу.

11. Розробіть програму, яка створює новий документ з даного PDF-документа за заданими параметрами (зокрема номерами сторінок).

12. Розробіть програму, яка розміщує в документі таблицю з комірками, у яких розташовані задані зображення.

13. Розробіть програму, яка створює з даного PDF-документу буклет.

14. Розробіть програму, яка з'єднає частини двох PDF-документів у один.

15. Розробіть програму, яка створює документ, у якому розташовано визначені частини PDF-документа у вигляді зображень.