

## Лабораторная работа № 35-36

**Тема:** Формирование отчетов.

**Программное обеспечение:** MS Visual Studio 2015, библиотека NPOI.

**Цель:** научиться выполнять автоматическую генерацию отчетов: экспорт данных в XLS формат.

**Время на выполнение:** 2 часа.

Отчет – документ, содержащий информацию в структурированном удобочитаемом виде. Ранее уже организовывались данные из базы данных для представления в таблице. Далее будем понимать отчет как офисный документ, содержащий выдержку из некоторой таблицы, как описание ситуации, и элементы статистики, как предпосылку к выводам и принятию решений по сложившейся ситуации.

**Задание 1:** Реализовать формирование отчета на примере списка абитуриентов. Для выполнения задания откройте проект, созданный в предыдущей лабораторной работе.

### Описание выполнения задания:

Сформируем документ Microsoft Office (таблицу Excel), в котором представим следующую информацию:

- количество записей в данной таблице БД (список всех абитуриентов);
- список абитуриентов;
- количество зачисленных абитуриентов;
- количество не зачисленных абитуриентов;
- количество абитуриентов, забравших документы;
- дата формирования отчета.

Обязанности по подготовке такого документа возложим на методиста (форма 1). Для этого добавим в Основное рабочее окно методиста (Form1) еще одну кнопку, при нажатии которой будет отображаться диалоговое окно выбора места сохранения отчета и выполняться генерация отчета (см. Рисунок 1-2):

| №  | Фамилия   | Имя        | Отчество     | Статус           | Город          |
|----|-----------|------------|--------------|------------------|----------------|
| 22 | Иванов    | Петр       | Сергеевич    | не зачислен      | Чебоксары      |
| 23 | Попова    | Людмила    | Валентиновна | не зачислен      | Сургут         |
| 24 | Сыроежкин | Сергей     | Павлович     | зачислен         | Ханты-Мансийск |
| 25 | Шариков   | Егор       | Андреевич    | не зачислен      | Казань         |
| 26 | Каренина  | Анна       | Васильевна   | забрал документы | Ханты-Мансийск |
| 27 | Трубецкой | Петр       | Фомич        | не зачислен      | Сургут         |
| 28 | Романова  | Анастасия  | Юрьевна      | зачислен         | Ханты-Мансийск |
| 35 | Иванова   | Александра | Петровна     | зачислен         | Казань         |
| 36 | Иванова   | Анастасия  | Сергеевна    | зачислен         | Чебоксары      |
| 37 | Абрамов   | Артем      | Сергеевич    | зачислен         | Москва         |
| 38 | Сайкова   | Альбина    | Сергеевна    | зачислен         | Чебоксары      |

Рисунок 1 - Внешний вид формы

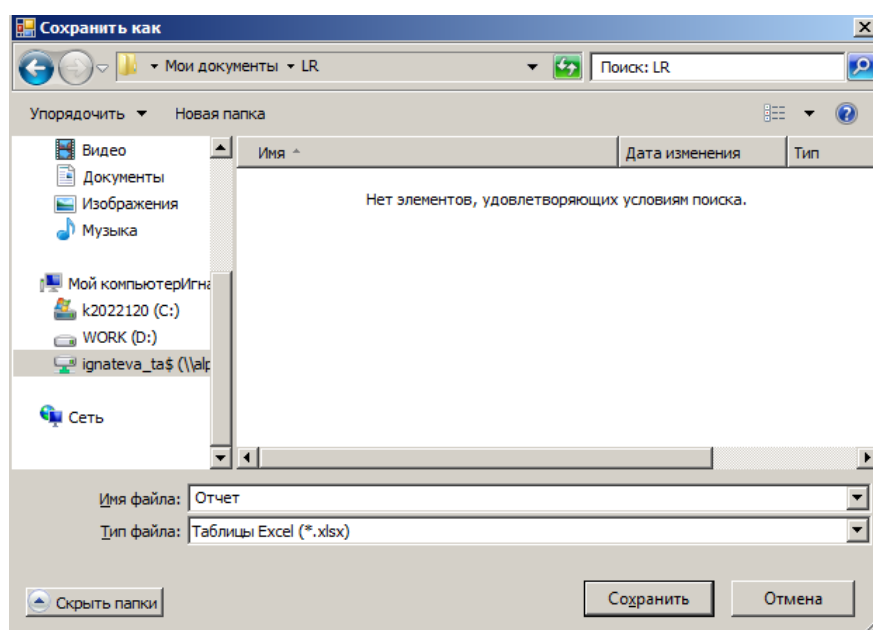


Рисунок 2 - Диалоговое окно сохранения отчета

Логика работы будет следующая:

```
private void button4_Click(object sender, EventArgs e)
{
    SaveFileDialog dialog = new SaveFileDialog();
    dialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
    dialog.DefaultExt = ".xlsx";
    dialog.Filter = "Таблицы Excel (*.xlsx)|*.xlsx|Все файлы (*.*)|*.*";
    dialog.FilterIndex = 1;
    dialog.FileName = "Отчет";

    if (dialog.ShowDialog() == DialogResult.OK)
    {
        var file = new FileStream(dialog.FileName,
            FileMode.Create,
            FileAccess.ReadWrite);
        ...
    }
}
```

Здесь создается новое диалоговое окно SaveFileDialog dialog, указывается, что по умолчанию необходимо сохранять файл Отчет.xlsx на Рабочем столе как Лист MS Excel,

и создается соответствующий FileStream. Осталось написать логику самого генератора, которая будет реализована с помощью библиотеки NPOI - .NET-версии Java Apache POI (the Java API for Microsoft Documents), позволяющей работать с документами Word и Excel. Получить библиотеку также просто, как и EF, – через Диспетчер пакетов NuGet (ИД: NPOI) (см. Рисунок 3).

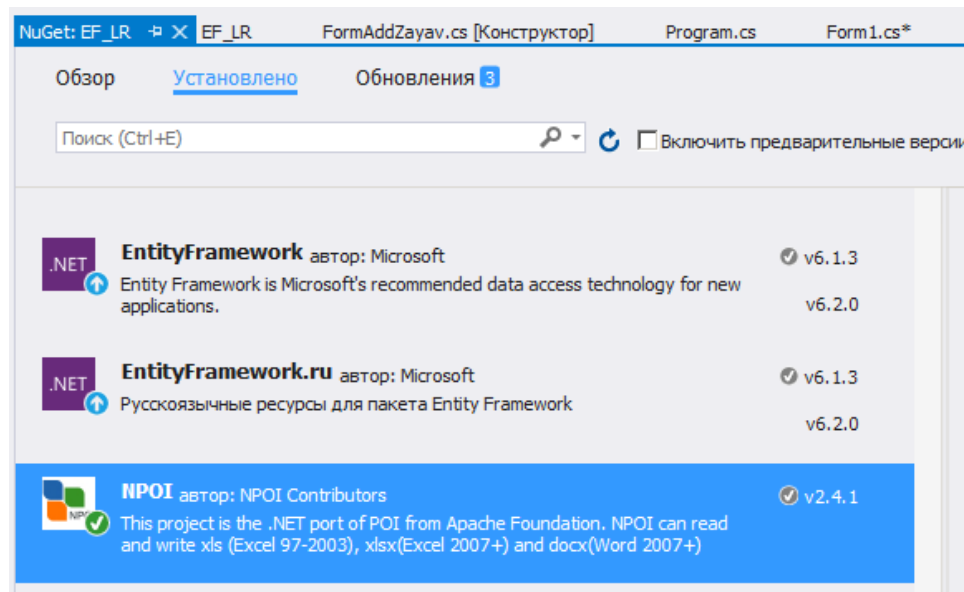


Рисунок 3 – Установка пакета NPOI

Apache POI содержит следующие компоненты API (таблица 1) для работы с MS Office-документами (цветом выделены компоненты, доступные в NPOI):

Таблица 1 – Состав NPOI

| Тип документа | Компоненты API |              |
|---------------|----------------|--------------|
|               | Office 97-2003 | Office 2007+ |
| Excel         | HSSF           | XSSF         |
| Word          | HWPf           | XWPF         |
| PowerPoint    | HSLF           | XSLF         |
| Outlook       | HSMF           |              |
| Visio         | HDGF           |              |
| Publisher     | HPBF           |              |

Помимо них и в Apache POI, и в NPOI включены компоненты для работы с OpenXml и OLE2.

Стоит оговорить, что .docx и .xlsx – это zip-архивы с содержимым документа в формате XML, графическими файлами (если есть в документе) и XML-описанием стилей, разметки и отношениями между элементами контейнера. Поэтому альтернативой NPOI будет работа именно с XML-содержимым. Для организации чтения и записи xls-файлов, помимо Systm.IO необходимо подключить пространство имен, соответствующее компоненту API XSSF:

```
using System.IO;
using NPOI.XSSF.UserModel;
```

Далее будем генерировать документ на основе xlsx-шаблона (в терминах 1С – макета печатной формы). Для этого создадим новую таблицу Excel и заполним ее, как показано на рисунке 4:


|    |   |  |            |                 |                   |                         |   |  |  |  |
|----|---|--|------------|-----------------|-------------------|-------------------------|---|--|--|--|
|    | A   | B  | C          | D               | E                 | F                       | G |  |  |  |
| 1  |  | <b>МЦК-ЧЭМК<br/>МИНОБРАЗОВАНИЯ ЧУВАШИИ</b> |            |                 | Отчет № _____     |                         |   |  |  |  |
| 2  |   |  |            |                 | от _____          |                         |   |  |  |  |
| 3  |   |  |            |                 |                   |                         |   |  |  |  |
| 4  |   |  |            |                 |                   |                         |   |  |  |  |
| 5  | <b>Список абитуриентов</b>  |  |            |                 |                   |                         |   |  |  |  |
| 6  |   |  |            |                 |                   |                         |   |  |  |  |
| 7  | Всего в базе данных   |  |            |                 |                   | записей об абитуриентах |   |  |  |  |
| 8  |   |  |            |                 |                   |                         |   |  |  |  |
| 9  |   |  |            |                 |                   |                         |   |  |  |  |
| 10 | <b>Код</b>  | <b>Фамилия</b>                             | <b>Имя</b> | <b>Отчество</b> | <b>Статус</b>     | <b>Город</b>            |   |  |  |  |
| 11 |   |  |            |                 |                   |                         |   |  |  |  |
| 12 |   |  |            |                 | Среди которых     |                         |   |  |  |  |
| 13 |   |  |            |                 | зачислено         |                         |   |  |  |  |
| 14 |   |  |            |                 | не зачислено      |                         |   |  |  |  |
| 15 |   |  |            |                 | забрали документы |                         |   |  |  |  |
| 16 |   |  |            |                 |                   |                         |   |  |  |  |

Рисунок 4 – Шаблон XLSX-файла

Здесь шрифт выделенных ячеек – полужирный, а выравнивание текста в них – по центру. В эти ячейки будут вноситься значения, генерируемые кодом приложения. Строки 1-10 -header (1С – «шапка») шаблона, 12-15 – footer («подвал» в 1С), а между строками 10 и 11 будут вставляться новые, содержащие информацию об абитуриентах. Упоминание «1С: Предприятие» здесь не случайно, структура решения здесь абсолютно идентична, поэтому при острой необходимости и небольших неоправданных усилиях возможна в некотором роде интеграция клиентского приложения с этой системой.

Чтобы уберечь файл шаблона от пользователя, необходимо спрятать его в ресурсах приложения (его размер в 63 КБ не сильно увеличит размер исполняемого файла): «Проект \ Свойства... \ Ресурсы \ Добавить ресурс \ Добавить существующий файл».

Дополните приведенный выше код кнопки следующими строками:

```
var template = new MemoryStream(Properties.Resources.template, true);
var workbook = new XSSFWorkbook(template);
var sheet = workbook.GetSheetAt(0);
//header
sheet.GetRow(1).GetCell(5)
    .SetCellValue(DateTime.Today.ToShortDateString());
sheet.GetRow(6).GetCell(4).
    SetCellValue(sheet.abiturients.Count());
//footer
sheet.GetRow(12).GetCell(5).SetCellValue(sheet.abiturients.
    Count(w => w.Статус == "зачислен"));
sheet.GetRow(13).GetCell(5).SetCellValue(sheet.abiturients.
    Count(w => w.Статус == "не зачислен"));
sheet.GetRow(14).GetCell(5).SetCellValue(sheet.abiturients.
    Count(w => w.Статус == "забрал документы"));
```

Здесь создается новая рабочая книга XSSFWorkbook workbook на основе шаблона (если бы работа шла с xls-файлами, тогда тип был бы HSSFWorkbook). Далее получаем первый (нумерация с нуля) лист таблицы. Затем заполняем ячейки с датой составления отчета и количеством абитуриентов на основе LINQ-запросов. Здесь важно помнить, что индекс ячейки – это пара чисел, а нумерация начинается с нуля. Получаем строку листа GetRow(), затем ячейку GetCell() и устанавливаем в ней нужное значение методом SetCellValue(). Стиль ячейки при этом не меняется, а формат зависит от содержимого: при внесении значения типа string – общий, а при int – числовой. Является ли абитуриент зачисленным в колледж, можно проверить с помощью несложной формулы и LINQ-запроса. Даты перед записью на лист необходимо привести к

короткому формату с помощью ToString().

Теперь сместим уже заполненный footer шаблона на необходимое (строки с 10 до конца, на sheetclients.Count(...) строк) количество строк вниз и организуем перебор всех нужных записей:

```
sheet.ShiftRows(10, sheet.LastRowNum, sheetabiturients.Count(), true, true);
int row = 10;
foreach (var item in sheetabiturients.OrderBy(o => o.Фамилия))
{
    ...
}
```

Где каждая запись вносится в таблицу файла:

```
var rowInsert = sheet.CreateRow(row);
rowInsert.CreateCell(0).SetCellValue(row - 9);
rowInsert.CreateCell(1).SetCellValue(item.Фамилия);
rowInsert.CreateCell(2).SetCellValue(item.Имя);
rowInsert.CreateCell(3).SetCellValue(item.Отчество);
rowInsert.CreateCell(4).SetCellValue(item.Статус);
rowInsert.CreateCell(5).SetCellValue(item.Город);
row++;
```

Как можно заметить, последовательность такова:

- 1) вставляется новая строка на лист;
- 2) заполняются соответствующие ячейки.

Очевидно, что для заполнения таблицы удобнее хранить данные в двумерном массиве, это позволило бы организовать цикл для обхода столбцов и соответствующих им полей записей. Однако для работы с БД выгоднее хранить записи как объекты, что сводит на нет применение второго индекса. Приведение List<abiturients> sheetabiturients к двумерному массиву вряд ли добавит удобства. Единственное, что можно здесь посоветовать, – организация отдельного метода: если необходимо формировать отчеты из нескольких таблиц, универсальный метод будет очень полезен. После всех манипуляций с таблицей нужно сохранить файл, в конструкцию if добавив

```
workbook.Write(file);
```

и посмотреть результат в Excel (см. Рисунок 5):

МЦК-ЧЭМК  
МИНОБРАЗОВАНИЯ ЧУВАШИИ

Отчет № \_\_\_\_\_  
от 02.08.2019

## Список абитуриентов

Всего в базе данных

11

записей об абитуриентах

| Код | Фамилия   | Имя        | Отчество     | Статус           | Город          |
|-----|-----------|------------|--------------|------------------|----------------|
| 1   | Абрамов   | Артём      | Сергеевич    | зачислен         | Москва         |
| 2   | Иванов    | Петр       | Сергеевич    | не зачислен      | Чебоксары      |
| 3   | Иванова   | Александра | Петровна     | зачислен         | Казань         |
| 4   | Иванова   | Анастасия  | Сергеевна    | зачислен         | Чебоксары      |
| 5   | Каренина  | Анна       | Васильевна   | забрал документы | Ханты-Мансийск |
| 6   | Попова    | Людмила    | Валентиновна | не зачислен      | Оурлут         |
| 7   | Романова  | Анастасия  | Юрьевна      | зачислен         | Ханты-Мансийск |
| 8   | Сайкова   | Альбина    | Сергеевна    | зачислен         | Чебоксары      |
| 9   | Сыроежкин | Сергей     | Павлович     | зачислен         | Ханты-Мансийск |
| 10  | Трубецкой | Петр       | Фомин        | не зачислен      | Оурлут         |
| 11  | Шариков   | Егор       | Андреевич    | не зачислен      | Казань         |

Среди которых

зачислено

6

не зачислено

4

забрали документы

1

Рисунок 5 - Сгенерированный отчет

В результате проделанной работы получаем программный модуль, позволяющий не только генерировать отчеты по записям в БД, но и гибко настраивать их содержимое и внешний вид. Использование платформы .NET и тесно взаимодействующих с ней EF и NPOI позволяет легко реализовать обработку информации на всех без исключения этапах: при получении ее из БД, при выдаче пользователю, при экспорте в офисный документ.

## **Задание 2 для самостоятельного выполнения по вариантам:**

**Вариант 1:** Сгенерировать отчет по данным из таблицы Заявления: ФИО абитуриента, Специальность, Статус.

**Вариант 2:** Сгенерировать отчет по данным из таблицы Экзамены: Специальность, Дисциплина, Дата экзамена, Положительный балл.

**Вариант 3:** Сгенерировать отчет по данным из таблицы Оценки: ФИО абитуриента, Специальность, Дисциплина, Оценка.

- 1) Разработайте xlsx-шаблон отчета в удобном вам табличном процессоре, поддерживающем xlsx-формат. Добавьте файл шаблона в ресурсы приложения.
- 2) Реализуйте заполнение отчета согласно варианту. Помните, что отчет должен содержать не просто выборку из БД, но и некоторые вычисляемые значения: количество записей, среднее значение, минимальный или максимальный показатель по столбцу и т. д.
- 3) Реализуйте сохранение стиля ячеек при вставке строк на лист.
- 4) Реализуйте сохранение файла отчета и выбор директории для сохранения через диалоговое окно.

