Отчеты в системе Pilot-ICE: глубокое погружение

Александр Стремнев

Подготовка отчетов в условиях электронного документооборота — немаловажная часть рабочего процесса. Модуль создания диаграмм DevExpress Charts предоставляет мощные средства для формирования отчетов в системе Pilot-ICE посредством интерфейса прикладного программирования и богатой палитры настраиваемых элементов визуализации данных.

Александр Стремнев,
к.т.н., доцент кафедры информационных технологий Белгородского государственного технологического университета им. В.Г. Шухова

В системе электронного документооборота Pilot-ICE есть всё необходимое для организации эффективной работы, включая сервер администрирования и клиентский модуль. Первый компонент позволяет определить структуру организации, систему циркулирующих в ней объектов-документов и разграничение прав доступа, клиентская же часть служит для непосредственной работы с документами и реализует такие бизнес-процессы, как выдача и контроль заданий, а также формирование отчетов. Именно подготовке отчетов в системе Pilot-ICE посвящен данный материал. Здесь необходимо сразу же сделать следующую оговорку. Если интерфейс Pilot-ICE в целом довольно дружелюбен (это касается даже такого «основательного» компонента, как модуль администрирования), то подготовка отчетов на определенном этапе предполагает погружение на сравнительно глубокий уровень работы с системой. Конечно же, в поставку Pilot-ICE входит базовый набор шаблонов для создания отчетов [1], использование которых сводится главным образом к нажатию «большой зеленой кнопки» *Построить отчет* (рис. 1). Но когда в шаблон необходимо внести изменения (а может быть, создать собственный отчет!), то наступает пора использовать следующую команду: *Редактировать отчет*.

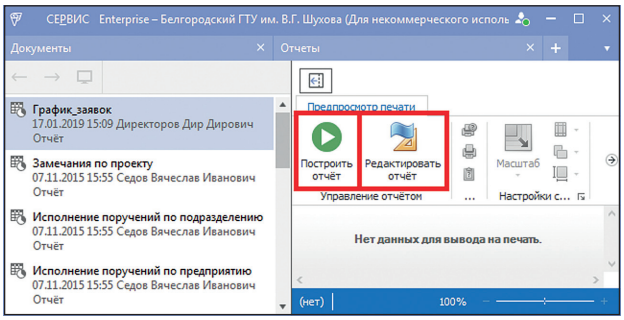

Рис. 1

Окно *Дизайнер отчета* представляет собой «конструктор», знакомый большинству разработчиков: в центре — поле макета, а вокруг — всё необходимое (палитра элементов управления, навигатор полей базы данных и окно свойств используемых объектов) — рис. 2. Если вид этого окна вас не испугал, будем разбираться дальше. Начнем с того, что в справочной системе Pilot-ICE имеется ряд пошаговых инструкций для подготовки как простых отчетов, так и шаблонов, содержащих запрос на ввод пользовательских пара-

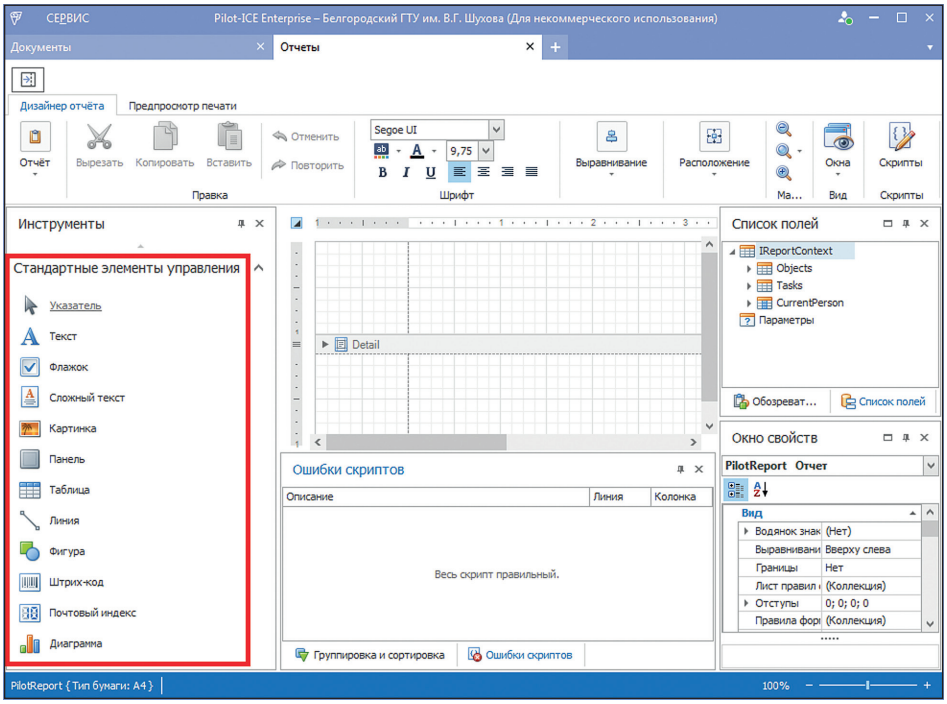

Рис. 2

метров [2]. В этих руководствах в числе прочих рассматриваются вопросы использования различных элементов управления (текстовых полей и таблиц). Мы попытаемся расширить наши возможности по визуальному представлению данных в отчетах посредством такого функционально насыщенного элемента, как *Диаграмма*.

Инструментарий для добавления диаграмм в отчеты Pilot-ICE реализован на основе компонента Chart Wizard (*Мастер Диаграмм*), разработанного компанией DevExpress [3].

Предположим, необходимо подготовить отчет в формате круговой диаграммы, на которой для выбранного исполнителя будет отображаться процентное соотношение просроченных и выполненных вовремя заданий.

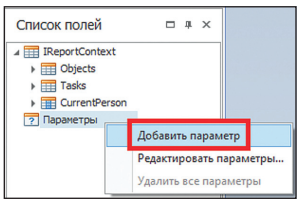

Рис. 3

Сперва в новом пустом отчете определим параметр для указания исполнителя. Для этого в окне *Список полей* секции *Параметры* вызовем команду *Добавить параметр* (рис. 3).

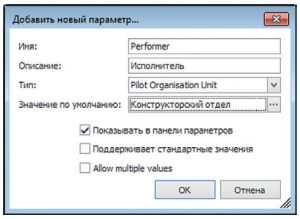

Рис. 4

Введем данные об имени (*Performer*), типе и значении параметра (рис. 4). Тип указывается, как *Pilot Organisation Unit* — элемент организационной структуры (сотрудник или подразделение). В качестве *Значения по умолчанию* можно выбрать любой объект структуры организации.

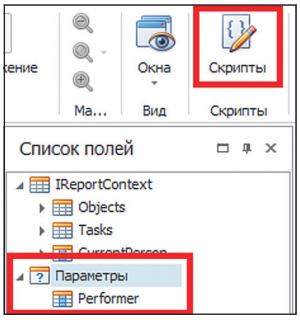

Рис. 5

Теперь параметр доступен в *Списке полей* окна *Дизайнера отчета* (рис. 5). Запрос на ввод значений параметра будет производиться при генерации отчета, а имя параметра понадобится при написании скрипта, формирующего выборку из базы документооборота. Для работы над скриптом щелкнем по соответствующей кнопке (*Скрипты*) в *Дизайнере отчета* (рис. 5).

В окне скрипта набираем код, приведенный в листинге.

```
//Ссылки на необходимые пространства имен
using System.Linq;
using System.Collections.Generic;
using Ascon.Pilot.Core;
using Ascon.Pilot.Client.Search;
using Ascon.Pilot.Core.WorkFlow;
using Ascon.Pilot.Pilot.Reports;
using DevExpress.XtraReports.Parameters;
//Для версий Pilot_ICE, начиная с 18.37.0.30607
//using Ascon.Pilot.DataClasses;

//Создание переменной для хранения выборки
//из базы документооборота Pilot-ICE
ReportContext context = new ReportContext();

//Процедура для подготовки выборки по заданному
//параметру (исполнителю), а также для заполнения
//и настройки диаграммы
private void PilotReport_DataSourceDemanded(object sender,
System.EventArgs e)
{LongRunning.Start(this, () => {

//Чтение значения параметра (выбранного исполнителя(ей))
var unit=(ROrganisationUnit)Parameters[«Performer»].
Value;
//Получение идентификаторов всех исполнителей
//в выбранном подразделении
var ids=GetPositions(unit);

//Создание построителя запросов
var tasksBuilder = QueryBuilder.CreateTaskQueryBuilder();
//Выполнение запроса по идентификатору(ам) исполнителя(ей)
tasksBuilder.Must(TaskFields.ExecutorPositionId.BeAnyOf(ids.
ToArray()));
//Создание выборки с информацией о заданиях исполнителя
context.Tasks = context.GetTasks(tasksBuilder);

//Подсчет общего количества заданий исполнителя (task_count)
//и количества просроченнных (task_exp)
int task_count = 0;
int task_exp = 0;
foreach (Ascon.Pilot.Pilot.Reports.RTask cur_task in
context.Tasks) {
if (cur_task.IsDeadlineExpired){
task_exp++;}
task_count++;
}

//Завершение программы при отсутствии заданий у исполнителя
if (task_count==0){
return;}

//Удаление существующего ряда в диаграмме
chart1.Series.Remove(chart1.Series.GetElementByIndex(0));

//Создание нового ряда значений для круговой диаграммы
Series series1 = new Series(«заявки», ViewType.Pie);

//Подготовка значений для отображения на диаграмме
var p_in_time=(task_count-task_exp)*100/task_count;
var p_exp=task_exp*100/task_count;

//Заполнение ряда значениями (точками данных)
series1.Points.Add(new SeriesPoint(«Выполнено вовремя»,
p_in_time));
series1.Points.Add(new SeriesPoint(«Просрочено», p_exp));
//Размещение ряда на диаграмме
chart1.Series.Add(series1);

//Настройка параметров границ между секторами диаграммы
PieSeriesView myView = (PieSeriesView)series1.View;
myView.Border.Visible = true;
```

```
    myView.Border.Color = Color.White;
    myView.Border.Thickness = 2;

    //Заполнение легенды названиями точек диаграммы
    series1.LegendTextPattern = «{A}»;
    //Вертикальное и горизонтальное выравнивание легенды
    chart1.Legend.AlignmentVertical = LegendAlignmentVertical.
Center;
    chart1.Legend.AlignmentHorizontal = LegendAlignmentHorizontal.
Left;

    });}

    //Функция, возвращающая массив идентификаторов всех
    //исполнителей из выбранного для отчета подразделения
    private IEnumerable<int>GetPositions(ROrganisationUnit unit)
    {return unit.IsPosition ?
    new[] {unit.Id}:
    unit.Children.SelectMany(x => GetPositions(x));}
```

Более подробную информацию об используемых в скрипте специальных классах можно получить в разделе справочной системы Pilot-ICE, касающейся API для построения отчетов [4], а также в структуре классов диаграмм DevExpress [5].

Теперь выборка данных и настройки будущей круговой диаграммы для отчета подготовлены. Повторным щелчком по кнопке *Скрипты* вернемся в «визуальное» окно дизайнера и добавим в секцию *Detail* вложенный отчет (рис. 6).

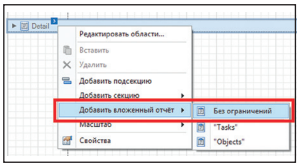

Рис. 6

Затем «перетащим» экземпляр объекта типа «диаграмма» из палитры стандартных элементов управления в добавленную область вложенного отчета. При этом автоматически запустится *Мастер диаграмм*, на первом шаге которого выбирается тип диаграммы. В нашем примере это *Круговая диаграмма* (рис. 7).

Следующие шаги *Мастера* позволяют детально определить внешний вид диаграммы и настроить точки данных. В нашей задаче все необходимые действия для отрисовки диаграммы мы уже приготовили в скрипте, поэтому в *Мастере* после выбора типа можно сразу щелкнуть по кнопке *Готово* (рис. 8).

Аннотируем отчет, добавив в него информацию об исполнителе заявок-заданий. Для этого поместим экземпляр элемента *Текст* перед графиком-диаграммой в *Дизайнере отчета* (рис. 9).

Изменим текст в добавленной надписи (рис. 10).

Для отображения конкретного исполнителя, выбираемого при формировании отчета, создадим еще один объект-надпись. Для ее заполнения будем использовать данные о параметре из *Списка полей* в окне *Дизайнера отчета* (рис. 11).

Перетащим параметр *Performer* в подготовленное текстовое поле (рис. 12).

Сохраним шаблон отчета и перейдем к его формированию на вкладке *Предпросмотр печати* (рис. 13).

При построении отчета становится доступной панель для ввода значений параметров. Завершив ввод нажатием кнопки *OK*, мы получим искомые результаты в виде диаграммы, иллюстрирующей процентное соотношение заявок определенному исполнителю, выполненных вовремя и просроченных (рис. 14).

При выборе исполнителя заявки становится доступной организационная структура, позволяющая выбрать как подразделение, так и отдельного сотрудника (рис. 15).

Итак, поставленная задача решена. Отчет в компактной и наглядной форме представляет информацию о заданиях, выполняемых указанным сотрудником или подразделением. Стоило ли ради этого углубляться в «джунгли» API Pilot-ICE и DevExpress? Думается, что в этой области разработчикам

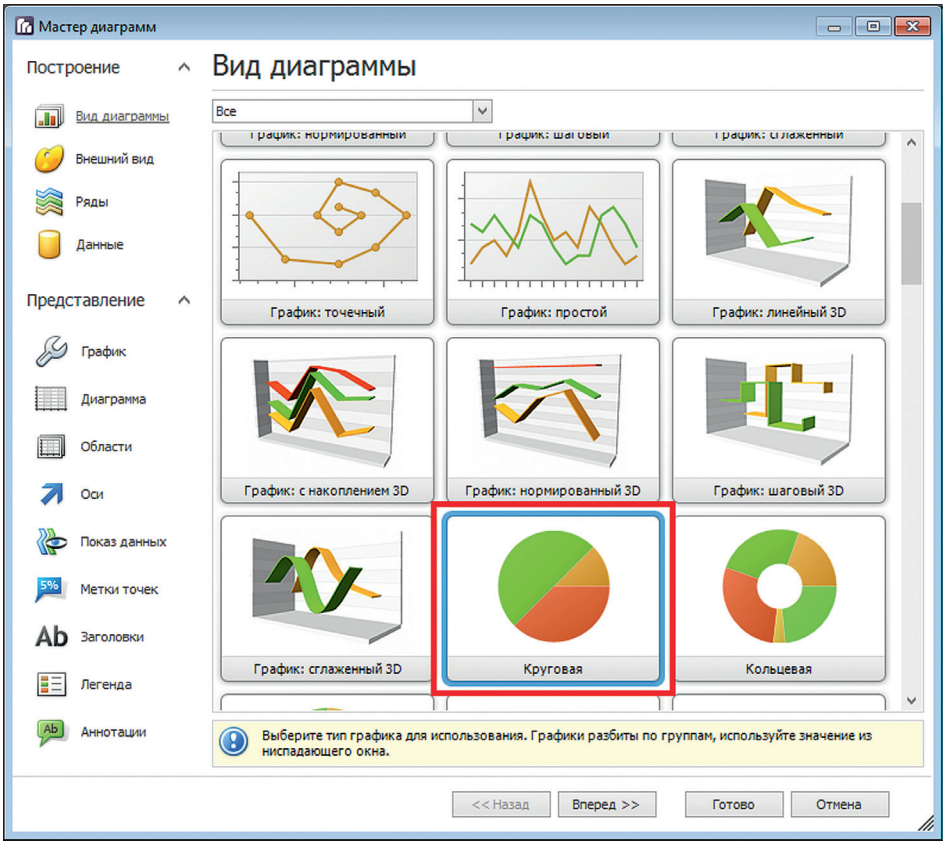

Рис. 7

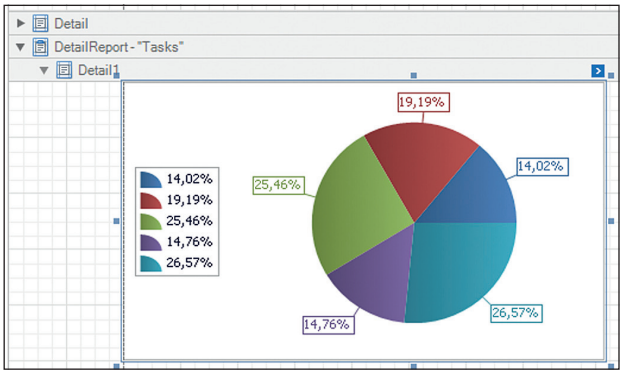

Рис. 8

Pilot-ICE имеет смысл стремиться к ими же поднятой (довольно высоко) планке дружелюбия интерфейса системы к рядовым пользователям.

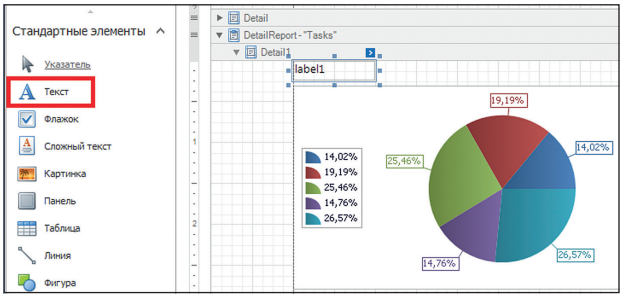

Рис. 9

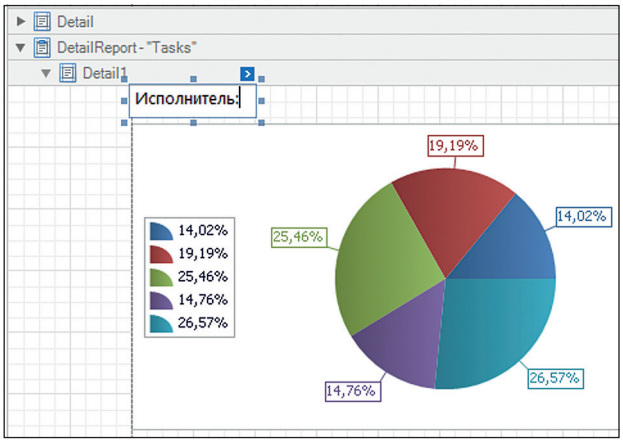

Рис. 10

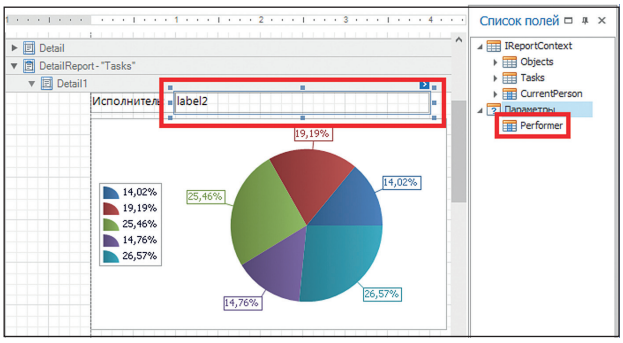

Рис. 11

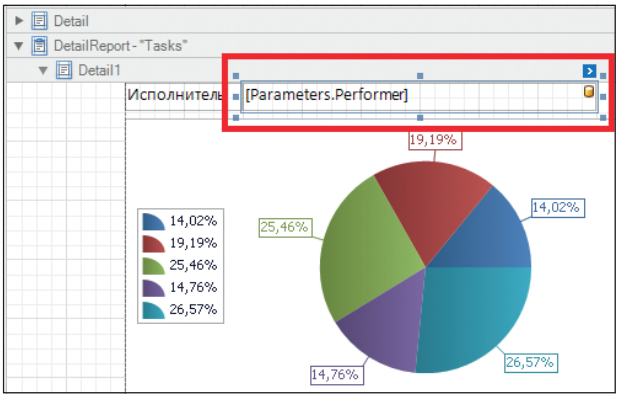

Рис. 12

Шаблон рассмотренного нами отчета можно скачать [6], добавить в систему Pilot-ICE и при необходимости изменить в нем те или иные настройки.

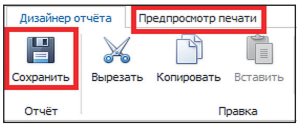

Рис. 13

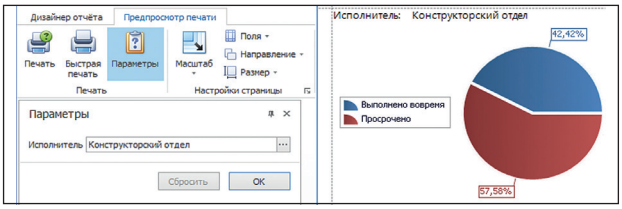

Рис. 14

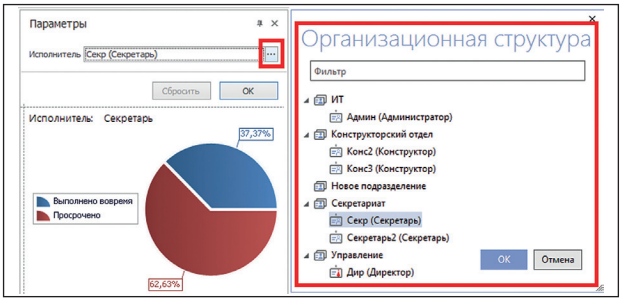

Рис. 15

Полезные ссылки:

1. Базовый комплект отчетов для системы Pilot-ICE // *https://pilot.ascon.ru/release/Report-Samples_ru.zip*
2. Online-справка по созданию шаблонов отчетов в системе Pilot-ICE // *http://help.pilotems.com/ru/Content/report_template.htm?tocpath=Pilot-ICE|Отчеты|_____2*
3. Документация по Chart Wizard // *https://documentation.devexpress.com/WindowsForms/7039/Controls-and-Libraries/Chart-Control/Design-Time-Features/Chart-Wizard*
4. Online-справка по API для построения отчетов в системе Pilot-ICE // *http://help.pilotems.com/ru/Content/report_api.htm?tocpath=Pilot-ICE|Отчеты|_____3*
5. Структура классов для диаграмм DevExpress // *https://documentation.devexpress.com/CoreLibraries/DevExpress.XtraCharts.namespace*
6. Файл рассматриваемого в статье примера отчета // *https://yadi.sk/d/D4Hdmkiw7K2J_w*