Оглавление

[Основные шаги реализации и состав проекта 2](#_Toc468806273)

[Вид сверху 3](#_Toc468806274)

[Запуск приложения 4](#_Toc468806275)

[Главная форма 4](#_Toc468806276)

[Модуль доступа к данным 5](#_Toc468806277)

[Класс менеджера 5](#_Toc468806278)

[Класс контроллера 6](#_Toc468806279)

[Форма списка 7](#_Toc468806280)

[Вывод данных 7](#_Toc468806281)

[Вывод связанных данных 7](#_Toc468806282)

[Установка команды 8](#_Toc468806283)

[Вызов списка 9](#_Toc468806284)

[Установка фильтра 9](#_Toc468806285)

[Форма редактора 10](#_Toc468806286)

[Привязка данных к полям 10](#_Toc468806287)

[Установка команды 11](#_Toc468806288)

[Поведение формы 11](#_Toc468806289)

[Дополнительно 12](#_Toc468806290)

[Поля и свойства форм 12](#_Toc468806291)

[События форм 12](#_Toc468806292)

[Методы форм 13](#_Toc468806293)

[Свойства списков 14](#_Toc468806294)

[События списков 14](#_Toc468806295)

[Свойства и события поля выбора 15](#_Toc468806296)

[Свойства контроллера 16](#_Toc468806297)

[Методы контроллера 16](#_Toc468806298)

# Основные шаги реализации и состав проекта

Для реализации необходимо

1. Спроектировать модель данных Linq2Sql
2. Реализовать контроллер – наследник DataObject (реализующий IDataController) - его абстрактные/виртуальные методы, которые описывают логику взаимодействия с объектами модели: получение данных, изменение/удаление данных, выполнение других действий.
3. Реализовать формы (как наследники FormEdit и FormList) для отображения наборов данных и редактирования объекта модели:
   1. Дизайн формы (компоненты DataList, SelectBox, DataTimeBox, NumberBox и любые другие)
   2. Связь контролов с полями объектов, которые будут получены от контроллеров
   3. Логика поведения формы, при необходимости – создание дополнительных команд и их обработчиков.
   4. Описание взаимодействия с другими формами (свойства DataList или SelectBox или вызов в обработчиках через FormManager или методы FormBase)
4. Установить связь форм и контроллеров через объект, реализующий IdataManager
5. Главная форма – запуск форм через FormManager. Запуск приложения.

Состав:

1. Библиотечные модули

* Common.dll – общие методы, контейнер контроллеров, интерфейсы, выдача сообщений
* Ctrls.dll – контролы – DataList (с фильтром и поиском) и Box-ы для выбора, даты и чисел
* Forms.dll – FormBase, FormEdit, FormList, класс команды формы и контейнера команд, класс FormManager
* Manager.dll – классы DataObject и Master

1. Модули, которые необходимо реализовать:
   * Модуль модели Linq2Sql
   * Модуль форм (главная и все наследники от FormEdit и FormList)
   * Модуль контроллеров данных (все наследники DataObject и реализация IDataManager)

# Вид сверху

Запуск формы:

IDataManager – создание контроллеров для формы

FormManager – инициализация формы

Обработка:

через DataObject:IDataController

Возврат данных и результатов.

Первоначальное получение данных объектов. Получение изменений данных объектов через стандартный механизм биндинга.

Объекты модели данных

Компоненты

Команды

Объекты данных

Форма

# Запуск приложения

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

try

{

// загрузить настройки из имя\_приложения.xml

AppConfig.Load();

// создать главную форму

Form form = new FormMain();

// создать менеджер форм, связяв его с главной формой

FormManager.Io = new FormManager(form);

// создать контроллер приложения, связав его с менеджерами

Master master = new Master(new DataManager(), FormManager.Io);

Application.Run(form);

}

catch(Exception e)

{

Loger.SendMess(e, "Ошибка запуска приложения!");

}

}

}

## Главная форма

Это обычная форма от Form

Необходимо в обработчиках меню прописать вызовы соответствующих форм через FormManager, можно сразу проверить подключение, можно обработать F9 (в FormBase для немодальных форм по ней активизируется ToolStrip главного окна)

private void toolStripButton2\_Click(object sender, EventArgs e)

{

FormManager.Io.ExecForm(FormManager.Io.GetForm("имяФормы"), this);

}

private void FMain\_Shown(object sender, EventArgs e)

{

Refresh();

FormManager.Io.CheckConnection(this);

}

private void FMain\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.F9)

tools.Items[0].Select();

}

# Модуль доступа к данным

## Класс менеджера

Должен реализовать **IDataManager** – методы :

public bool CheckConnection() - тут необходимо проверить подключение

public List<IDataController> GetDataObjects(string viewType) - тут необходимо создать, заполнить (в зависимости от viewType) и вернуть List<IDataController> - это список объектов IDataController - как правило наследников DataObject

В конструкторе можно получить Connection – например через AppConfig.Prop("ConnectionString")

Пример:

public class DataManager : IDataManager

{

string conn;

public DataManager()

{

conn = AppConfig.Prop("ConnectionString");

}

//-------------------------------------------------------------------------

public bool CheckConnection()

{

conn = AppConfig.Prop("ConnectionString");

try

{

DB db = new DB(conn);

db.Connection.Open();

db.Connection.Close();

return true;

}

catch (Exception e)

{

Loger.SendMess(e, "Ошибка подключения!");

return false;

}

}

//-------------------------------------------------------------------------

public List<IDataController> GetDataObjects(string viewType)

{

List<IDataController> res = new List<IDataController>();

if (viewType == "имя\_формы1")

{

res.Add(new DataКлассКонтроллера1(new DB(conn)));

}

if (viewType == "имя\_формы2")

{

var d = new DataКлассКонтроллера(db);

res.Add(new DataКлассКонтроллера (d));

res.Add(new DataКлассКонтроллераДет(new DB(conn),"Имя",d));

}

. . .

return res;

}

Здесь объект DB – это объект контекстного модуля, который представляет из себя автоматически сгенерированную реализацию System.Data.Linq.DataContext и его членов.

Если два DataObject создаются в одном контексте и один из них дочерний, то в 3-м параметре конструктора необходимо указать родителя.

Объект DataManager создается при запуске приложения.

## Класс контроллера

В общем случае - должен реализовывать IDataController. Необходимо наследовать от DataObject , который уже реализует этот интерфейс и предоставляет абстрактные и виртуальные методы для реализации логики работы с конкретными объектами контекста. См.snippet DataController

Существуют следующие привязки/способы вызова методов контроллера (DataObject или наследника), делегатов интерфеса IDataController и обработчиков (делегатов грида/формы):

|  |  |  |  |
| --- | --- | --- | --- |
| **Делегат грида/формы или способ вызова** | **IDataController** | **DataObject** | **Реализация в контроллере** |
| из FormBase.InitData | OnGetDataBinds | GetDataBinds | override |
| DataList.DoGetList | OnGetList | GetList | override |
| FormEdit.DoGetEditData | OnGetEditData | GetEditData | override + вызов DataObject.GetEntityEditData |
| DataList.DoSetDefaults | OnSetDefaults | SetDefaults | override |
| DataList.DoDelete | OnDelete | Delete | override + вызов DataObject.DeleteEntities, возможна предобработка |
| DataList.DoSave  FormEdit.DoSave | OnSave | Save | override + вызов DataObject.SaveEntity, возможна предобработка |
| DataList.DoClone | OnCloneEntity | CloneEntity | override |
| DataList.DoCheck  FormEdit.DoCheck | OnCheck | Check | override (не обязательно)  по умолчанию вызывает SetDefaults и CheckEntity |
| DataList.DoExecCommand  FormEdit.DoExecCommand | OnExecCommand | ExecCommand | override |
| из  FormEdit.SetMenuThruController  DataList.SetMenuThruController | OnSetCommands | SetCommands | override |

Подробнее – см. в Дополнительно – свойства и методы контроллера.

# Форма списка

Наследуем от FormList.

Заполняем:

**DataControllerName** – имя контроллера, объект которого будет в поле **data** и из которого (через OnGetDataBinds) мы хотим получить объекты в **bss** (например для использования в фильтре), вызвать команды или еще что-то…

**DefaultControlName** – на него встает фокус

**MainList** – этот DataList будет главным

**Text** – заголовок формы

## Вывод данных

1. Кидаем грид (**DataList**), определяем столбцы, для них указываем имя поля в источнике, для дат указываем **DefaultCellStyle**-**Format** = d
2. Для **DataList** задаем:

**DataControllerName** – имя контроллера, объект которого вернет данные (OnGetList), вызовет удаление (OnDelete), вызов команды по-умолчанию (OnExecCommand), а для редактируемого грида – и сохранение (OnSave, OnCheck, OnSetDefaults, OnCloneEntity)

**DefaultSort** – имя столбца для сортировки

**EditFormName** – имя формы для редактирования

**KeyNames** – ключевые поля источника через ;

**ReadOnly** = true если грид редактируемый

1. Делаем контроллер с GetList.
2. Прописываем вызов этой формы в главной форме или еще где-то (SelectBox или команда другой формы).
3. Прописываем в менеджере данных ветку для этой формы, в которой определяем контроллеры для нее.

## Вывод связанных данных

1. Для гридов дочерних списков задаем:

Свой **DataControllerName** - вызовет те же делегаты, кроме OnGetList, т.к. данные дочерних списков поступают при начитке главного списка.

**FilterKeyNames** = «поле объекта источника этого грида = ключевое поле источника главного грида;...» - нужно, чтобы получать id родительской записи при добавлении/редактировании дочерней.

**MainList** = false

1. Переопределяем Link()

protected override void Link()

{

listChild.SetMaster(listParent, "имяКоллекцииИсточника");

base.Link();

}

имяКоллекцииИсточника – поле в наборе, который возвращает GetList родительского контроллера, в котором сидят дочерние записи. Причем, если дочерняя запись должна содержать поля из каких-то родительских объектов, их нужно включить в класс дочернего объекта (сделать его partial с соответствующими get-свойствами).

## Установка команды

1. Переопределяем

protected override void SetCommands()

{

base.SetCommands();

cmds.Items.Add(new Command("КодКоманды", "Надпись", image, "Текст кнопки", "Текст подсказки", new[] { tools, menus }));

. . . .

menus.Items[menus.Items.Add(new ToolStripSeparator())].Visible = false;

}

При необходимости в параметрах конструктора устанавливаем обработчик (onExec: Action<string>) и признак постоянной активности (activeOnDefault) – иначе обработчики и поведение определяем в событиях формы (напр. OnControlChanged) или грида (OnSetMenu)

1. Делаем обработчик грида - OnSetMenu

private void grid\_OnSetMenu(object obj)

{

if (grid.CellClicked)

((Dictionary<string, Action<string>>)obj).Add("КодКоманды", grid.ExecCommand);

grid.SetMenuThruController(data, obj);

}

Тут мы изменяем словарь обработчиков команд, куда по умолчанию входят Добавить, Изменить, Удалить, Добавить копию... и др.

CellClicked учитываем, если команда должна быть доступна только при активной ячейке

SetMenuThruController вызываем, если словарь обработчиков команд изменяется в контроллере (data или какой-то dc[“имя”]) – метод вызовет SetCommands указанного контроллера

1. В общем случае можно использвать коллекцию команд из свойства cmds формы.
2. Для обработки выполнения команды вместо штатного вызова ExecCommand контроллера грида – можно обработать событие грида OnExecCommand
3. Для обработки результата выполнения команды можно обработать событие грида OnCommandResult

## Вызов списка

Нужно установить команду (в SetCommands) и в обработчике события грида OnExecCommand обработать ее, не забыв для остальных команд вызвать стандартную обработку (см. snippet OnExecCommand)

private object dataList2\_OnExecCommand(string cmd, object key, object filter, object data, object[] keys)

{

if (cmd == "Команда")

{

FormManager.Io.ExecForm("ФормаСписка", null, FormModes.Default, null, null, key, true);

return null;

}

else if (dataList2.DoExecCommand != null)

{

return dataList2.DoExecCommand(cmd, key, filter, data, keys);

}

else

return null;

}

Здесь в список мы передаем ключ текущей записи грида в качестве фильтра, поэтому этот ключ попадет как фильтр в GetDataBinds контроллера вызываемой формы. Последний параметр (true) необходим для обновления уже открытой формы.

## Установка фильтра

1. Связать поля фильтра с источниками данных – переопределить BindFilter. См. о привязке данных к полям в редакторе.

protected override void BindFilter()

{

base.BindFilter();

sbПолеВыбора.BindParentFromForm(. . . .);

. . .

}

1. Обработать событие формы OnControlChanged - если необходимо что-то сделать при изменении полей фильтра, например обновить данные:

private void форма\_OnControlChanged(Control c, bool entered)

{

if (!entered && c.Name == "sbПолеВыбора")

LoadData(null, null);

}

1. Обработать событие грида OnSetFilter – оно срабатывает, когда нужны данные фильтра. Их нужно начитать из объекта источника данных фильтра и занести в словарь «имя поля - значение»

private object грид\_OnSetFilter()

{

Dictionary<string, object> f = new Dictionary<string, object>();

f.Add("имя поля", CommonLib.GetValueFromObject(bss["источник данных фильтра"].Current, "имя поля"));

. . .

return f;

}

1. Если данные фильтра приходят извне, то, чтобы обновлять их в уже открытой форме, необходимо переопределить SetExternalFilter.

# Форма редактора

Наследуем от FormEdit.

## Привязка данных к полям

Необходимо переопределить метод Bind:

protected override void Bind()

{

base.Bind();

DataSourceUpdateMode m = DataSourceUpdateMode.OnValidation;

//----------------

BindingSource bs = bss[EntityName];

tПоле.DataBindings.Add("Text", bs, "имя поля источника", true, m);

sbПолеВыбора.Bind...From...(...);

}

Вместо источника bss[EntityName] может использоваться другой объект контроллера (из его словаря DataBinds) – он сидит в словаре bss с таким же ключом. Объекты контроллера должны быть заполнены в GetDataBinds или в GetEditData.

Варианты настройки полей для выбора:

1. Выбор только значений выбранного объекта из формы со списком

BindTextFromForm(BindingSource extSource, string extSourceFields,

string form, Func<object, object> getFunc, string propName,

bool editable, bool nullable, Binding bind = null)

Вызывает форму с именем form. После выбора из списка присваивает полям extSource значения полей «выбранного объекта». Соответствие полей указывается в extSourceFields как «поле\_extSource = поле\_источника\_списка\_выбора». Функция, которая возвращает «выбранный объект» по ключу выбранной строки списка передается в getFunc.

В качестве getFunc используется, как правило, элемент словаря делегатов контроллера (GetParentFuncs), получаемы через функцию GetParentFunc контейнера контроллеров формы.

Компонент отобразит поле propName из extSource.

Editable позволяет редактировать поле, nullable – очищать, даже если оно не редактируемое.

Вместо propName можно задать bind.

1. Выбор объекта из формы со списком (например - родительский объект для редактируемого объекта)

BindParentFromForm(BindingSource thisSource, string keyNames,

BindingSource extSource, string extSourceParent, string extSourceFields,

string form, Func<object, object> getFunc, string propName,

bool nullable, Binding bind = null)

Здесь поведение то же, но дополнительно есть возможность указать источник (thisSource), в DataSource которого будет помещен выбранный объект, а также имя поля extSourceParent для extSource, которому будет присвоен выбранный объект.

В keyNames указываются имена ключевых полей (через ;) для thisSource. Это необходимо для правильной обработке объекта.

В отличие от BindParentFromForm, компонент отобразит поле propName из thisSource.

1. Выбор объекта из выпадающего списка (например - родительский объект для редактируемого объекта)

BindParentFromCombo(object ds, string member,

BindingSource thisSource, string keyNames,

BindingSource extSource, string extSourceParent, string extSourceFields,

Func<object, object> getFunc, bool nullable)

Получаем выпадающий список с DataSource = ds и DataMember = member. Здесь та же логика занесения выбранного в списке объекта в thisSource.DataSource и в поле extSourceParent в extSource, и его полей в поля extSource через соответствие в extSourceFields.

Редактировать выбранное значение нельзя. Nullable = true дает возможность очистить поле.

1. Выбор значения из выпадающего списка

BindTextFromCombo(object ds, string member, BindingSource bs, string propName,

bool editable, bool nullable, Binding bind = null)

Получаем выпадающий список с DataSource = ds и DataMember = member. Можно указать источник (bs) и его поле (propName) с которым будет связано выбранное значение.

## Установка команды

Обычно переопределяем SetCommands (см.для формы списка), указывая в конструкторе параметр onExec: Action<string>, в котором указываем или ExecCommand – тогда вызов пойдет в контроллер формы, в его метод ExecCommand, либо пишем метод, в котором что-то делаем, например обновляем источник перед вызовом команды (SetControlsData) и только затем вызываем ExecCommand

protected override void SetCommands()

{

base.SetCommands();

cmds.Items.Add(new Command("Код", "Текст", null, null, null, new[] { tools }, onExec: SomeProc));

}

private void SomeProc(string cmd)

{

SetControlsData();

ExecCommand(cmd);

}

Также можно вызвать SetMenuThruController в обработчике события формы OnSetMenu если команды рулятся в контроллере

Для обработки вместо штатного вызова ExecCommand контроллера – можно обработать событие формы OnExecCommand

Для обработки результата выполнения команды можно обработать событие формы OnCommandResult

## Поведение формы

* Установить минимальный размер – нужно в конструкторе после InitializeComponent() вызвать SetFormEditSize(контрол, меньше правой границы которого не будет)
* Что-то сделать перед сохранением – нужно обработать событие BeforeSave
* Описать логику работы формы – в событии OnControlChanged(Control текущий\_контрол, bool вошли\_в\_контрол). Событие отрабатывает как вход (событие Enter), так и изменение (CheckedChanged для CheckBox и RadioButton, AfterGetSelected для SelectBox и SelectedIndexChanged для ComboBox)

# Дополнительно

## Поля и свойства форм

|  |  |  |
| --- | --- | --- |
| FormBase | protected Dictionary<string, BindingSource> bss; | словарь источников |
| FormBase | protected ErrorProvider error; |  |
| FormBase | protected DataControllers dc; | контейнер контроллеров данных |
| FormBase | protected IDataController data; | основной контроллер данных |
| FormBase | protected Commands cmds; | контейнер команд |
| FormBase | protected object inKey; | ключ внешнего объекта |
| FormBase | protected object inFilter; | ключи внешнего фильтра |
| FormBase | protected bool getResult; | форма вернет результат |
| FormBase | protected bool newEntity; | форма задает новый объект |
| FormBase | protected bool isReady; | форма инициализирована |
| FormBase | protected Action<object> CallBack; | делегат вызывающей формы, получает ключ объекта |
| FormBase | public object InKey | ключ внешнего объекта: return inKey |
| FormBase | public object InFilter | ключи внешнего фильтра: return inFilter |
| FormBase Browsable | public string DataControllerName | имя основного контроллера |
| FormBase Browsable | public string DefaultControlName | имя основного контроллера |
| FormEdit Browsable | public string EntityName | имя редактируемого объекта (для выбора из словаря источников) |
| FormEdit Browsable | public string KeyNames | имена ключевых полей через ; |
| FormEdit | protected Func<object, bool, bool> DoSave | делегат для сохранения - из контроллера (IDataController.OnSave) |
| FormEdit | protected Func<object, object> DoCheck { get; set; } | делегат для проверки - из контроллера (IDataController.OnCheck) |
| FormEdit | protected Action<object, bool, object> DoGetEditData | делегат для начитки данных объекта - из контроллера (IDataController.OnGetEditData) |
| FormEdit | protected Func<string, object, object, object, object[], object> DoExecCommand | делегат для выполнения произвольной команды - из контроллера (IDataController.OnExecCommand) |
| FormList | protected List<DataList> dataLists; | список гридов формы |
| FormList | protected DataList activeList; | активный грид |
| FormList Browsable | public DataList MainList | главный грид |

## События форм

|  |  |  |
| --- | --- | --- |
| FormBase | public event Action<Control, bool> OnControlChanged; | При входе в контрол или изменении данных в нем: Action(контрол, true если вошли в контрол)  Используется для реализации логики взаимодействия контролов на форме |
| FormEdit | public event Action BeforeSave; | перед сохранением |
| FormEdit | public event Func<object, bool, bool> OnSave; | при сохранении: Func(редактируемый объект, признак добавления) - вернет true если нет ошибок |
| FormEdit | public event Func<object, object> OnCheck; | при проверке: Func(редактируемый объект) - вернет словарь ошибок (имя поля, текст ошибки) |
| FormEdit | public event Action<object, bool, object> OnGetEditData; | при начитке: Action(ключ, признак нового объекта, ключ родителя для нового объекта) |
| FormEdit | public event Func<string, object, object, object, object[], object> OnExecCommand; | при выполнении произвольной команды: Func(код команды, ключ, ключи фильтра, обрабатываемый объект) - вернет объект результата |
| FormEdit | public event Action<string, object> OnCommandResult; | после выполнения команды: Action(код команды, объект результата) |
| FormEdit | public event Action<object> OnSetMenu; | после установки команд (для установки дополнительных команд): Action(словарь команд "имя - делегат вызова") |

## Методы форм

|  |  |  |
| --- | --- | --- |
| FormBase | public void Init(DataControllers d, FormModes modes = FormModes.Default, Action<object> callback = null, object key = null, object filter = null) | Инициализация свойств формы. Вызов InitControls, SetControllers, SetCommands |
| FormBase | protected override void OnLoad(EventArgs e) | Вызов InitData и Bind, завершение инициализации, начатой в Init. |
| FormBase | protected virtual void InitControls() | Начальные установки для контролов |
| FormBase | protected virtual void SetControllers() | Привязка контроллеров к обработчикам |
| FormBase | protected virtual void SetCommands() | Установка команд |
| FormBase | protected virtual void InitData() | Начитка объектов данных в контроллерах, наполнение словаря источников из объектов основного контроллера. |
| FormBase | protected virtual void Bind() | Привязка контролов к данным. |
| FormBase | public virtual void LoadData(Control sender, object key) | Загрузка данных из контроллера. |
| FormBase | protected virtual void ExecSelectForm(Control sender, string formName, object key, object filter, Action<object> callback) | Запуск формы списка для выбора |
| FormBase | protected void SetControlsData() | Записать данные из контролов в биндинги всех источников из словаря источников |
| FormBase | protected void SetControlsData(object src) | Записать данные из контролов в биндинги указанного источника |
| FormBase | protected virtual void SetBindingSources(IDataController d) | Заполнение словаря источников из объектов данных контроллера |
| FormEdit | public void SetMenuThruController(IDataController c, object cmds, string code = "") | Обработка набора команд через контроллер (вызывается например из обработчика OnSetMenu) |
| FormEdit | protected virtual object GetEntityBind() | Редактируемый объект |
| FormEdit | protected virtual object GetEntityKey() | Ключ редактируемого объекта |
| FormEdit | protected void ExecCommand(string cmd) | Обработчик команды |
| FormEdit | protected virtual void SaveData() | Сохранение данных редактируемого объекта |
| FormEdit | protected virtual bool CheckData(object o) | Проверка данных объекта и установка ошибок |
| FormEdit | protected void SetFormEditSize(Control control) | Установка размеров формы по контролу на табличной панели |
| FormList | protected virtual void BindFilter() { } | Установка фильтров (привязка данных к полям фильтра) |
| FormList | public virtual void SetExternalFilter(object filter) | Установка внешнего фильтра |
| FormList | protected virtual void Link() { } | Установка подчиненности гридов |
| FormList | protected virtual void ExecRefresh(string cmd) | Команда обновления списков |
| FormList | protected virtual void ExecSelect(string cmd) | Команда выбора значения списка |

## Свойства списков

|  |  |  |
| --- | --- | --- |
| Browsable | public BindingSource ThisSource | источник данных (по умолчанию - из GetList контроллера) |
| Browsable | public bool MainList | признак главного грида |
| Browsable | public string DataControllerName | имя контроллера, для получения и обработки данных источника |
| Browsable | public string DeleteMessage | предупреждение перед удалением |
| Browsable | public string KeyNames | имена ключевых полей объекта источника данных через ; |
| Browsable | public string FilterKeyNames | пары "поле объекта источника данных = поле родительского объекта" через ; |
| Browsable | public string EditFormName | имя формы для редактирования |
| Browsable | public string DefaultSort | имя столбца для начальной сортировки с указанием направления: asc (по умолчанию) или desc |
| Browsable | public bool CanFilter | возможна фильтрация |
| Browsable | public bool CanSearch | возможен поиск |
| Browsable | public bool CanExcelExport | возможна выгрузка в Excel |
| Browsable | public int MaxRowsToExcel | максимально допустимое количество строк для выгрузки в Excel |
|  | public bool CellClicked { get; private set; } | признак нахождения в ячейке |
|  | public Func<object, object, object> DoGetList | получить коллекцию данных: Func(ключ, ключи фильтра) |
|  | public Action<object, object> DoSetDefaults | установка умолчаний: Action(объект, ключ внешнего(родительского) объекта) |
|  | public Action<object[]> DoDelete | удаление объектов: Action(массив ключей) |
|  | public Func<object, bool, bool> DoSave | сохранение объекта: Func(объект, признак нового) - вернуть true если нет ошибок |
|  | public Action<object, object> DoClone | передача данных из одного объекта в другой: Action(объект-источник, объект-приемник) |
|  | public Func<object, object> DoCheck | проверка данных объекта: Func(объект) - вернуть словарь "поле-ошибка" |
|  | public Func<string, object, object, object, object[], object> DoExecCommand | обработать команду: Func(код команды, ключ, ключи фильтра, обрабатываемый объект) - вернуть объект результата |
|  | public Action<Control, string, bool, object, object, Action<object>> DoExecEdit | запуск редактора: Action(this, имя форма, признак нового, ключ, ключи фильтра, метод обработки завершения редактирования) |

## События списков

|  |  |
| --- | --- |
| [Browsable(false)]  public event Action<Dictionary<string, Action<string>>> OnCheckMenu; | переустановить меню (стандартно подписывается контейнер команд формы): Action(словарь "имя команды - делегат запуска"), цель - перенаправить вызов команд формы на свои делегаты |
| [Browsable(false)] public event Action<Control, object> OnReload; | необходимо перечитать данные (стандартно подписывается форма, чтобы синхронизировать свое поведение, связанные гриды и т.п.): Action(грид, ключ) |
| public event Action<object> OnSetMenu; | после установки команд (для установки дополнительных команд): Action(словарь команд "имя - делегат вызова") |
| public event Func<DataList, int, string, object> OnGetRowKey; | запрос ключа записи: Func(грид, индекс строки, имена ключевых полей) |
| public event Action<string, object> OnCommandResult; | постобработка команды: Action(код команды, объект результата) |
| public event Func<object> OnSetFilter; | формирование ключей фильтра: Func(существующие ключи фильтра) - вернуть новые ключи фильтра |
| public event Func<object, object, object> OnGetList; | получить коллекцию данных: Func(ключ, ключи фильтра) |
| public event Action<object, object> OnSetDefaults; | установка умолчаний: Action(объект, ключ внешнего(родительского) объекта) |
| public event Action<object[]> OnDelete; | удаление объектов: Action(массив ключей) |
| public event Func<object, bool, bool> OnSave; | сохранение объекта: Func(объект, признак нового) - вернуть true если нет ошибок |
| public event Action<object, object> OnClone; | передача данных из одного объекта в другой: Action(объект-источник, объект-приемник) |
| public event Func<object, object> OnCheck; | проверка данных объекта: Func(объект) - вернуть словарь "поле-ошибка" |
| public event Func<string, object, object, object, object[], object> OnExecCommand; | обработать команду: Func(код команды, ключ, ключи фильтра, обрабатываемый объект) - вернуть объект результата |
| public event Action<bool, object, object> OnExecEdit; | запуск редактора: Action(признак нового, ключ, ключи фильтра) |

## Свойства и события поля выбора

( про методы настройки – см. Варианты настройки полей для выбора в описании привязок для редактора )

|  |  |
| --- | --- |
| public BindingSource ThisSource | источник данных, которому присваивается выбранный объект |
| public string KeyNames | имена ключевых полей объекта источника данных через ; |
| public BindingSource ExtSource | другой источник данных (например - редактируемый объект), полям которого может быть присвоен выбранный объект или значения его полей |
| public string ExtSourceParent | имя поля в ExtSource, которому будет присвоен выбранный объект (например - родительский объект для редактируемого объекта) |
| public string ExtSourceFields | пары "поле ExtSource = поле выбранного объекта" через ; |
| public string SelectFormName | имя формы для выбора |
| public bool Nullable | допустимо пустое значение |
| public bool Editable | допустимо редактирование текста |
| public bool Locked | заблокирован |
| public SelectBehaviorType SelectBehavior | поведение - выбор из выпадающего списка или из формы |
| public event Action<object, object, Action<object>> OnExecSelect; | Вызов формы выбора: Action(ключ, ключи фильтра, метод обработки завершения выбора) |
| public event Func<object, object> OnGetParent; | Получить объект по выбранному ключу: Func(ключ) - должен вернуть объект или typeof. Возврат null вызывает ошибку. Вызывается из метода обработки завершения выбора, если не задано OnGetSelected. |
| public event Func<object, string, object> OnGetKey; | Получить ключ объекта по именам ключевых полей (вместо стандартного способа) |
| public event Func<object> OnSetFilter; | Получить ключи фильтра для передачи на форму для выбора |
| public event Action<object> OnGetSelected; | Обработка выбранного ключа. Перекрывает OnGetParent Вызывается из метода обработки завершения выбора |
| public event EventHandler AfterGetSelected; | После обработки завершения выбора |
| [Browsable(false)] public Action<Control, string, object, object, Action<object>> DoExecSelect | Вызов формы выбора: Action(this, имя формы, ключ, ключи фильтра, метод обработки завершения выбора). В базовой форме устанавливается делегат базовой формы. Вызывается если не задано OnExecSelect. |
| [Browsable(false)] public Func<object, object> DoGetParent | Получить объект по выбранному ключу: Func(ключ) - должен вернуть объект или typeof. Возврат null вызывает ошибку. При инициализации устанавливается делегат контроллера данных.  Вызывается если не задано OnGetSelected или OnGetParent |

## Свойства контроллера

Свойства DataObject и IDataController

|  |  |
| --- | --- |
| protected DataContext db; | модель |
| protected DbConnection conn | объект подключения |
| protected DataObject parentDataObject; | связанный контроллер (родительские данные) |
| protected List<DataObject> childDataObjects | связанные контроллеры (дочерние данные) |
| protected DateTime minSqlSmallDate | DateTime.ParseExact("19000101", "yyyyMMdd", null); |
| protected DateTime minSqlDate | DateTime.ParseExact("17530101", "yyyyMMdd", null); |
| protected string msgNoValue | "Не задано значение" |
| protected string msgNoUnique | "Значение не уникально" |
| **свойства IDataController** |  |
| public string Name | имя контроллера |
| public Dictionary<string, object> DataBinds | словарь объектов данных (имя-объект) |
| public Dictionary<string, Func<object, object>> GetParentFuncs | словарь функций (имя-делегат) для возврата объекта по ключу, используется например при выборе через SelectBox |
| public Action<object, object> OnGetDataBinds | заполнить словарь объектов данных и словарь функций |
| public Func<object, object, object> OnGetList | получить коллекцию данных (ключ, ключи фильтра) |
| public Action<object, bool, object> OnGetEditData | заполнить в словаре объект для редактирования (ключ, признак нового, ключ внешнего(родительского) объекта) |
| public Action<object, object> OnSetDefaults | установка умолчаний (объект, ключ внешнего(родительского) объекта) |
| public Action<object[]> OnDelete | удаление объектов (массив ключей) |
| public Func<object, bool, bool> OnSave | сохранение объекта (объект, признак нового) - вернуть true если нет ошибок |
| public Action<object, object> OnCloneEntity | передача данных из одного объекта в другой (объект-источник, объект-приемник) |
| public Func<object, Dictionary<string, string>> OnCheck | проверка данных объекта (объект) - вернуть словарь поле-ошибка |
| public Func<string, object, object, object, object[], object> OnExecCommand | обработать команду (код команды, ключ, ключи фильтра, обрабатываемый объект) - вернуть объект результата |
| public Action<object, object, object, object[], string> OnSetCommands | установить набор команд из интерфейса (набор команд (слоарь код-действие), ключ, обрабатываемый объект, массив ключей выделенных записей) |

## Методы контроллера

Методы DataObject

|  |  |
| --- | --- |
| public DataObject(DataContext context, string name = "Main") | Конструктор. Базовая инициализация контроллера данных, установка всех делегатов для IDataController. При необходимости отключить некоторый функционал нужные делегаты IDataController обнуляются в конструкторах наследников |
| public void CheckKey<T>(object o, string name) | Проверка наличия ключа по имени и типу |
| public void ChangeContext(DataContext context) | Переустановка модели контекста в этом и связанных контроллерах |
| public T GetKey<T>(object o, string name) | Выдать значение ключа. Сообщить если не найден по имени и типу |
| public bool KeyExists<T>(object o, string name) | Проверка существования ключа по имени и типу |
| public T KeyValue<T>(object o, string name) | Значение ключа по имени и типу |
| public abstract object GetObject(object key); | вернуть текущий объект по ключу, ключ обязателен |
| public abstract object GetEntity(object key); | вернуть актуальный объект сущности по ключу (использовать для получения родительского объекта), необходимо обновить коллекцию, в случае отсутствия ключа вернуть тип или новый объект |
| public abstract void GetDataBinds(object key, object filter); | заполнить словарь объектов данных и словарь функций |
| public abstract object GetList(object key, object filter); | получить коллекцию данных |
| public abstract void GetEditData(object key, bool add, object addKey); | подготовить данные для редактирования |
| public abstract void SetDefaults(object data, object addKey); | установка умолчаний  для случая добавления из под родителя, нужно его установить – см.snippet ParentDefault |
| public abstract void Delete(object[] keys); | удаление объектов по ключам - нужно вызывать DeleteEntities<T> или явно реализовывать |
| public abstract bool Save(object data, bool add); | сохранение - нужно вызывать SaveEntity или явно реализовывать |
| public abstract void CloneEntity(object src, object dst); | передача данных из одного объекта в другой |
| public virtual Dictionary<string, string> Check(object data) | проверка данных объекта, с предварительной установкой умолчаний, переопределяем при необходимости изменить стандартный подход к проверке, вызывает SetDefaults и CheckEntity |
| protected abstract void CheckEntity(object data, Dictionary<string, string> errs); | проверка данных объекта, заполнение словаря ошибок  для распространенных проверок есть snippet-ы: CheckDate, CheckParent, CheckString |
| public abstract object ExecCommand(string command, object key, object filter, object data, object[] keys); | обработать команду |
| public abstract void SetCommands(object cmds, object key, object data, object[] keys, string code); | установить/скорректировать набор команд из интерфейса |
| protected T GetEntityEditData<T>(object key, bool add, object addKey) where T : class, new() | получить данные объекта для редактирования |
| protected void DeleteEntities<T>(object[] keys, Table<T> table, Action<object> beforeDelete) where T : class | удаление объектов по ключам - стандартный способ, с предобработкой по каждому объекту |
| protected bool SaveEntity<T>(Table<T> table, object data, bool add) where T : class | сохранение объекта - стандартный способ, с учетом вставки |
| protected bool SaveEntity(object data, bool add, Action<object, bool> beforeSave) | сохранение объекта - стандартный способ, с предобработкой |
| protected void Reject() | Сбросить помеченные к удалению или вставке. Перечитать измененные |