

## **Обзор шаблонов**

### **Что такое шаблон**

Шаблоны Clariion - это гибко настраиваемые, с диалоговым интерфейсом, интерпретируемые скрипты для генерации кода. Темплейты обычно предлагают Вам ответить на некоторые вопросы, на основании ответов на которые строится определенный набор модулей с исходным кодом. В добавление к этим вопросам, многие шаблоны предоставляют точки вставки исходного кода в Ваше приложение - точки, в которых Вы можете определить исходный код, который попадает в исходный код, сгенерированный шаблоном. Вы можете воспринимать вопросы шаблонов как способ определения статических (на момент компиляции) характеристик программы или процедуры, а встроенный исходный код - как способ определить изменяющиеся (на момент исполнения) характеристики программы или процедуры.

### **Вопросы, запрашиваемые шаблонами**

Обычно шаблон предлагает Вам вводить информацию во время разработки. Application Generator интерпретирует темплейт и представляет диалог со всеми приглашениями темплейтов. Вы заполняете приглашения, используя встроенную помощь, для того, чтобы определить статические (на момент компиляции) характеристики Вашей программы или процедуры. Например, заполняете приглашение Record Filter для обеспечения фильтра для шаблона BrowseBox.

### **Точки вставки кода в шаблонах**

В дополнение к этим приглашениям, многие шаблоны предлагают также точки встраивания исходного кода в Ваше приложение или процедуру - точки, в которых Вы можете обеспечить определенный код, который попадает в код, сгенерированный по темплейту. Вы также можете использовать эти точки вставки для определения изменяющихся (на момент исполнения) характеристик программы или процедуры. Например, Вы можете вставить исходный код для того, чтобы “спрятать” связанный список, когда нет связанных записей для отображения в списке. Для получения более подробной информации об использовании точек вставки кода, смотрите Application Generator—Embedded Source Code в User’s Guide.

## **Преимущества шаблонов**

Темплейты обеспечивают многократное использование и централизованное обслуживание кода. Они обладают теми же преимуществами, что и объектно-ориентированное программирование, особенно возможностью многократного использования. Вдобавок, темплейты могут дополнять и усиливать использование объектно-ориентированного кода, обеспечивая легко используемые “обертки” для сложных объектов. Темплейты ABC и библиотека ABC являются превосходным примером такой связи между шаблонами и объектами.

## **Гибкость шаблонов**

Вы можете изменять шаблоны по Вашему усмотрению и хранить Ваши модификации в Template Registry. Смотрите более подробную информацию в User's Guide—Maintaining Your Templates. Вы можете также добавлять и использовать темплейты третьих фирм вместе с темплейтами Clarion. Вы можете создавать собственные темплейты. Template Language документирован в Programmer's Guide и в оперативной помощи.

## **Шаблоны Clarion и шаблоны Application Builder Class (ABC)**

Clarion поставляется с несколькими классами (наборами) темплейтов или цепочками(chains) темплейтов. По умолчанию, темплейты устанавливаются в директорию \CLARION4\TEMPLATE. Вдобавок, темплейты ABC становятся зарегистрированными во время установки Clarion. Смотрите более подробную информацию в Registering Templates ( User's Guide ).

## **Шаблоны ABC**

Темплейты ABC становятся зарегистрированными во время установки Clarion. Они являются самыми новыми темплейтами и используют наиболее продвинутые возможности генерации кода, включая генерацию объектно-ориентированного кода - кода, который основывается на библиотеке Application Builder Class (ABC) Library.

Шаблоны ABC включают:

ABCChain.tpl	Class ABC - Application Builder Class Templates
ABWizard.tpl	Class ABC Wizards - Clarion Wizard Templates

## **Шаблоны Clarion (Совместимость)**

Вместе с шаблонами ABC, Clarion 4 включает последние темплейты Clarion for Windows 2.00x Templates. Эти темплейты не регистрируются по умолчанию. Они включаются только для обеспечения совместимости. Мы не рекомендуем использование этих темплейтов для разработки новых приложений, так как темплейты ABC находятся в центре дальнейших усилий Topspeed. Темплейты Clarion включают:

CW.tpl            Class Clarion - Clarion Release Templates  
Wizard.tpl        Class Wizards - Clarion Wizard Templates

## **Шаблоны ABC и генерация кода**

Шаблоны ABC генерируют исходный код несколькими способами. Различные темплейты из этой поставки генерируют все - от одного предложения до целых процедур и приложений:

### **Class ABC Wizards**

#### Quick Start Wizard

Генерирует словарь данных с одним файлом и целое приложение для просмотра, поиска, обновления и печати данных.

#### Application Wizard

Генерирует целое приложение, на основе существующего словаря данных с одним или большим количеством связанных или несвязанных файлов и включающее главное меню с подчиненными процедурами просмотра, поиска, редактирования и печати данных.

#### Procedure Wizards

Генерирует ориентированные на данные процедуры (просмотр данных, ввод данных и отчеты), основанные на особых описаниях в словаре данных. Сгенерированный код согласуется с определенными между файлами связями, включая множество процедур для поддержки обновления и проверки первичных и связанных файлов.

### **Class ABC**

#### Procedure Templates

Генерирует ориентированную на одну задачу или ориентированную на данные процедуру (меню, сплеш-экраны, ввод данных, отчеты и др.)

#### Control Templates

Генерирует исходный код для определения одного или более оконного элементами и для управления этими элементами, загружая данные в элементы управления и из них, скроллируя и выбирая данные и т.д.

## Control Templates

Генерирует множество предложений задаче-ориентированного исходного кода в том месте, которое Вы укажете.

## Extension

Templates Генерирует множество предложений задачно-ориентированного исходного кода в одной или большем количестве определенных точек для выполнения задачи расширения. Расширения могут применяться как для одной процедуры, так и для целого приложения.

Часть 1 Application Handbook (эта часть) описывает все темплейты Clarion ABC и обеспечивает инструкции и предложения для заполнения их настроечных полей.

## **ABC Templates и ABC Library**

---

Шаблоны ABC плотно связаны с библиотекой ABC Library. Однако, темплейты гибко настраиваемы и разработаны так, чтобы позволить Вам подставлять Ваши собственные определения классов. За дополнительной информацией по настройке глобального уровня пересечения между темплейтами ABC и библиотекой ABC обращайтесь к разделу Classes Tab Options (Global). Для настройки локального пересечения (на уровне модуля) между темплейтами ABC и библиотекой ABC обращайтесь к разделу Classes Tab Options (Local).

## **Классы и их Объекты, Генерируемые Темплейтами (Template Generated Objects)**

---

Темплейты ABC создают экземпляры объектов из библиотеки ABC. Имена объектов, генерируемые темплейтами, обычно связаны с соответствующими именами классов, но не совпадают полностью. Код Вашего приложения, сгенерированный по темплейтам ABC, может содержать объявления данных и исполняемые предложения, похожие на эти:

GlobalErrors	ErrorClass
Hide:Access:Customer	CLASS(FileManager)
INIMgr	INIClass
ThisWindow	CLASS(ReportManager)
ThisWindow	CLASS(WindowManager)
ThisReport	CLASS(ProcessClass)
ThisProcess	CLASS(ProcessClass)
BRW1	CLASS(BrowseClass)
EditInPlace::CUS:NAME	EditClass

Resizer	WindowResizeClass
Toolbar	ToolbarClass

CODE

```
GlobalResponse = ThisWindow.Run()
BRW1.AddSortOrder(BRW1::Sort0:StepClass,ST:StKey)
BRW1.AddToolbarTarget(Toolbar)
GlobalErrors.Throw()
Resizer.AutoTransparent=True
Previewer.AllowUserZoom=True
```

Объявления данных создают экземпляры объектов из ABC Library, а выполняемые предложения ссылаются на созданные экземпляры. Различные классы Application Builder Classes и их экземпляры в темплейтах перечислены ниже таким образом, чтобы Вы смогли отследить объекты ABC в сгенерированном коде и найти соответствующую документацию по ABC библиотеке.

**Объект, сгенерированный темплейтом      Класс ABC(Application Builder Class)**

GlobalErrors	E r r o r C l a s s
INIMgr	I N I C l a s s
Access:file	F i l e M a n a g e r
Relate:file	R e l a t i o n M a n a g e r
ThisWindow	WindowManager, ReportManager
BRWn	B r o w s e C l a s s
BRWn::Sortn:Locator	L o c a t o r C l a s s
BRWn::Sortn:StepClass	S t e p C l a s s
EditInPlace::field	E d i t C l a s s
Popup	P o p u p C l a s s
Resizer	W i n d o w R e s i z e C l a s s
Toolbar	T o o l b a r C l a s s
ToolbarForm	T o o l b a r U p d a t e C l a s s
RELn::Toolbar	T o o l b a r R e l t r e e C l a s s
ThisReport	P r o c e s s C l a s s
Previewer	P r i n t P r e v i e w C l a s s
ThisProcess	P r o c e s s C l a s s
ProgressMgr	S t e p C l a s s
FDBn	F i l e D r o p C l a s s
FDCBn	F i l e D r o p C o m b o C l a s s
ViewerN	A S C I I V i e w e r C l a s s

FileLookupN  
Translator

S e l e c t F i l e C l a s s  
TranslatorClass

## **Парадигма приложения Browse-Form**

---

Существует много различных способов структурирования программ баз данных и их процедур. По умолчанию, мастера (Wizards) Clarion и процедурные темплейты (ABC и Clarion) используют много-поточную (multi-threaded) парадигму Browse-Form для программ баз данных и процедур, генерируемых ими.

### **Multi-threading**

Многопоточные программы являются стандартом Windows, так как возможность выполнения нескольких нитей позволяет конечным пользователям управлять их программами, выбирая необходимую программу или процесс тогда, когда это ему нужно.

### **Browse-Form**

Парадигма Clarion Browse-Form использует Browses (окна с сортируемыми, скроллируемыми, с возможностью поиска и выбора списками данных), Forms (окна с единственной обновляемой записью базы данных), и Reports для организации и представления информации из базы данных конечным пользователям. В парадигме Clarion Browse-Form, Form отображает не только запись из первичного (primary) файла, но и связанные записи из других файлов в виде дочерних Browses. Browse может быть расширен возможностью “редактирования на месте” для обеспечения более лаконичного интерфейса пользователя с пониженным уровнем сложности.

### **Browse-Form и нормализованные данные**

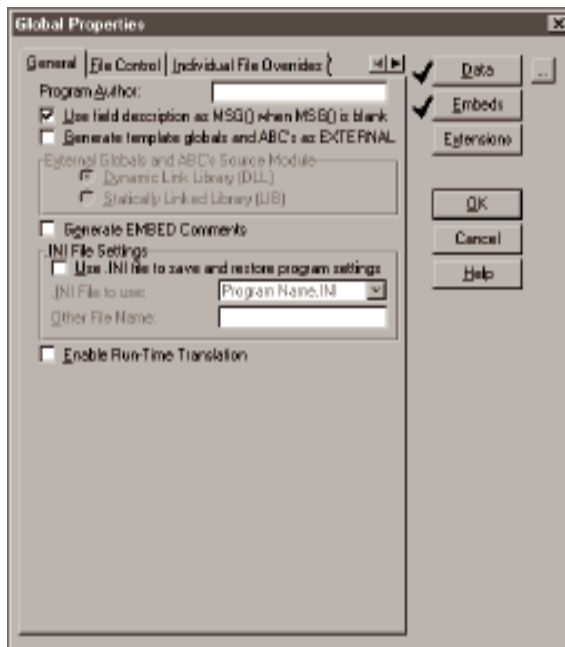
Прелесть этого подхода в том, что он хорошо работает с любыми нормализованными данными. Следовательно, он может уверенно применяться во многих ситуациях и будет давать согласующуюся, удобную, проверенную программу для бесконечного множества данных. Это делает подход Browse-Form идеальным для программирования общих задач баз данных.

Если у Вас есть необходимость использовать другую парадигму, Вы можете использовать для создания Вашего приложения индивидуальные процедурные, управляющие, кодовые темплейты и темплейты расширения. Например, большое количество Browses могут вызвать большой сетевой трафик в приложении клиент-сервер. В этом случае, парадигма Scrolling-Form может исключить Browsers и уменьшить сетевой трафик.

## Глобальные настройки ABC Template

Вы можете определить установки, которые будут влиять на все Ваше приложение, включая умолчания по обработке файлов, использования .INI файлов, глобальных переменных и встроенного исходного кода. Эти “глобальные” установки производятся через диалог Global Properties.

Эти глобальные поля диалога генерируются и обрабатываются темплейтом Class ABC Application. Каждое поле диалога имеет установку по умолчанию, применимую для большинства приложений. Во многих случаях Вы можете просто использовать установки по умолчанию.



Кнопки в диалоге Global Properties (Data, Embeds и Extensions) обеспечиваются Application Generator, а не ABC Application Template. Смотрите для получения более подробной информации по этим кнопкам разделы Global Embed Points в этой главе и Global Application Settings в User's Guide.

## Параметры закладки General

---

В закладке General в диалоге Global Properties Вы найдете следующие параметры:

### Program Author

Определите разработчика или организацию-разработчика.

### Use field description as MSG() when MSG() is blank

Установите этот флаг для использования описания поля в Data Dictionary в качестве сообщения по умолчанию в строке состояния для каждого поля приложения. Смотри MSG в Language Reference.

### Generate Template global data and ABC's as EXTERNAL

Добавляет атрибут EXTERNAL к описаниям глобальных переменных, сгенерированных темплейтами и атрибут DLL к каждому объявлению CLASS. Это означает, что Ваша программа обращается к внешней библиотеке для размещения памяти для этих переменных и объектов и для экспорта их таким образом, чтобы Ваша программа получила доступ к ним. Вы должны добавить атрибуты EXTERNAL и DLL для получения того же эффекта для любых глобальных переменных или классов, определяемых Вами. Для получения подробной информации об этих атрибутах смотрите Language Reference.

Замечание: Если Вы создаете программу, состоящую из более, чем одной DLL, созданной AppGen, Вы должны установить флаг Generate Global Data as EXTERNAL для всех приложений, кроме одного. Смотрите User's Guide—Development and Deployment Strategies.

### External Globals and ABC's Source Module

Определяет - подключается ли внешняя библиотека статически или динамически.

Это устанавливает параметр flag атрибута DLL для сгенерированных темплейтами объявлений классов. Для получения более подробной информации см. Language Reference по атрибуту DLL.

### Generate EMBED Comments

Установите этот признак для генерации опознавательных комментариев вокруг Вашего встроеного кода. Если Вы установите этот признак, то для оптимизации генерации комментариев необходимо установить также признак Enable embed commenting в диалоге Application Options (выберите Setup > Application Options в закладке Generation).

### Enable Run-Time Translation

Генерирует код для трансляции текста окна, базируясь на строке трансляции, определенной по умолчанию в файле UTILITY.TRN. Для получения более подробной информации см. Translator Class.



## **Поддержка .INI файла**

Темплейты Clarion поддерживают .INI файлы (стандартные файлы инициализации Windows). Это текстовые ASCII файлы, которые содержат информацию между сессиями.

Одно из использований .INI файлов - это сохранение для следующей сессии расположения окна на экране, предпочитаемого пользователем. Процедурные темплейты Clarion предлагают производить это автоматически, когда Вы включаете поддержку .INI файлов. Для получения более подробной информации см. Procedure Templates.

### **Для включения автоматической поддержки .INI файлов:**

1. Выберите закладку General в диалоге Global Properties.
2. Установите признак Use .INI file to save and restore program settings.
3. Определите имя .INI файла и путь к нему.

Чтобы определить имя, совпадающее с именем исполняемого модуля с расширением .INI, выберите Program Name.INI из выпадающего списка .INI file to use. Это размещает INI файл в системной директории Windows.

Для определения другого имени выберите Other из выпадающего списка, а затем заполните поле Other File Name. Вы можете определить полный путь, отсутствие пути, или путь относительно (.\) для генерации INI файла (как показано ниже).

Other File Name	Результирующее имя файла
c:\Programs\Payroll.ini	c : \ P r o g r a m s \ P a y r o l l . i n i
\Programs\Payroll.cfg	current drive:\Programs\Payroll\Payroll.cfg
Payroll.ini	windows system directory\Payroll.ini
.\Payroll.ini	current directory\Payroll.ini

4. Нажмите кнопку ОК для закрытия диалога Global Properties.

Для получения более подробной информации см. GETINI и PUTINI в Language Reference и в этой книге - INI Class and Procedure Templates.

Совет: Если Ваше приложение требует сохранения между сеансами дюжин или даже сотен переменных, не размещайте их в .INI файле, - используйте вместо него управляющий файл и нормальный ввод/вывод для файла. Возврат переменной из .INI файла производится относительно медленно. Также, если Вам необходимо прятать информацию от конечного пользователя, помните, что .INI файлы - это текстовые файлы и легко доступны.

## Параметры закладки File Control

---

Диалог Global Properties предлагает Вам переопределить некоторые из установок в Вашем словаре данных (см. раздел Dictionary Editor в User's Guide). Вы можете также определить, как Ваши процедуры осуществляют доступ к файлам. Вы можете определить эти атрибуты файлов для всех файлов или для каждого файла индивидуально. Для того, чтобы получить доступ к этим возможностям, выберите закладку File Control.

### Generate all file declarations

Установите этот признак для объявления всех файлов из Data Dictionary, производится или не производится на них ссылка из темплейтов приложения. Объявив все файлы, Вы можете обращаться к ним в любом написанном вручную коде в Вашем приложении.

### When done with a File

Определяет, будет ли приложение автоматически закрывать каждый файл, когда процедура завершается.

Совет: Единственно хороший способ разработки Вашего приложения для Windows - не захватывать системные ресурсы больше необходимого. Одним из лимитируемых ресурсов является количество открытых файлов(file handles). Вы можете "возвращать" file handles, не находящиеся в использовании, выбрав из выпадающего списка Close the File.

### Enclose RI code in transaction frame

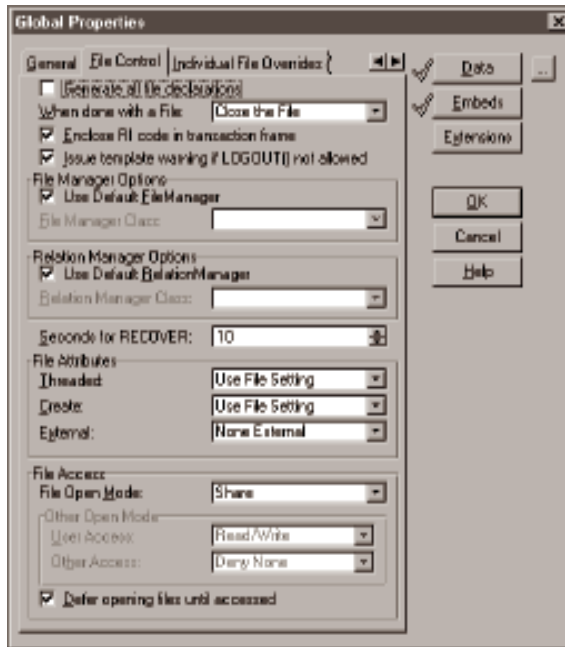
Установите этот признак для отката (ROLLBACK) изменений, если обновление завершится неуспешно во время выполнения операции поддержки ссылочной целостности (Referential Integrity)(transaction). Вы должны снять этот признак для файловых систем, которые не поддерживают транзакции, таких как Clipper, dBase, и FoxPro. Для получения более подробной информации см. LOGOUT, COMMIT, и ROLLBACK в Language Reference.

Совет: Если все файлы в цепочке связи используют одинаковую файловую систему и эта файловая система поддерживает транзакции, и Вы не хотите транзакционного обрамления вокруг RI - кода, Вы должны очистить признак для каждого файла в Individual File Overrides и в Global Settings.

### Issue Template warning if LOGOUT() not allowed

Когда Ваш словарь данных включает драйвер файла, не поддерживающий предложение LOGOUT (используемое в подпрограммах проверки

ссылочной целостности), установка этого параметра включает предупреждение в момент компиляции.



должны быть уверены, что этот параметр не установлен для драйверов типа dBaseIII. Для получения более подробной информации см. Database Drivers.

### Use default FileManager

Установите этот признак для генерации кода, использующего класс RelationManager, указанный на закладке Classes этого диалога. Снимите этот признак для выбора альтернативного класса из выпадающего списка RelationManager Class. Темплейты ABC создают экземпляр глобального RelationManager объекта, называемого Relate:file для каждого файла словаря данных. Объект Relate:file управляет связями файла, включая проверку связанных записей в других файлах и сохранение целостности связанных полей между связанными файлами. Для получения более подробной информации см. Relation Manager Class.

### Seconds for RECOVER

Определяет число секунд ожидания до инициализации процесса восстановления. Это применимо только для файлов Clarion. Для получения более подробной информации см. Database Drivers—Clarion

Files.

### Threaded

Определяет - добавляет ли или нет генератор приложений к структуре FILE атрибута THREAD. THREAD необходим для MDI-процедур browse и form для предотвращения конфликтов с буфером записи при изменении конечным пользователем фокуса от одной нити к другой.

### Use File Setting

Установите атрибут THREAD согласно установке в словаре данных. Для получения более подробной информации см. User's Guide—Dictionary Editor—File Properties.

### All Threaded

Добавляет атрибут THREAD к каждой структуре FILE.

### None Threaded

Пропускает атрибут THREAD для каждой структуры FILE.

### Create

Определяет, будет ли Ваше приложение создавать файл данных, если он не существует. Добавляет атрибут CREATE к каждой структуре FILE.

### Use File Setting

Устанавливает атрибут CREATE согласно установкам в словаре данных. См. User's Guide—The Dictionary Editor—File Properties.

### Create All

Добавляет атрибут CREATE к каждой структуре FILE.

### Create None

Определяет, добавляет ли или нет генератор приложений к структуре FILE атрибут EXTERNAL. EXTERNAL определяет, что память для буфера записи размещается внешней библиотекой. Для получения более подробной информации см. Language Reference.

Замечание: При использовании EXTERNAL для объявления структуры FILE, разделяемой несколькими библиотеками (.LIB или DLL и .EXE), структуру FILE без атрибута EXTERNAL должна определить только одна библиотека. Это обеспечит, что будет только один буфер записи, выделенный для структуры FILE и все библиотеки и .EXE будут обращаться к этой же памяти при доступе к элементам из этой структуры FILE.

### None External

Удаляет атрибут EXTERNAL из всех объявлений файлов и делает доступным поле ввода Export All File Declarations.

### Export All File Declarations

Установите этот признак для экспорта описаний файлов (см. Module Definition Files в Programmer's Guide). Это поле доступно, только если Вы определили Dynamic Link Library (.DLL) как Destination Type в диалоге Application Properties .

### All External

Добавляет атрибут EXTERNAL ко всем описаниям файлов и предлагает Вам определить Declaring Module и будет ли All files are declared in another .APP.

#### Declaring Module

Имя файла (без расширения) MEMBER - модуля, содержащее определение структуры FILE без атрибута EXTERNAL. Если структура FILE определена в модуле PROGRAM, оставьте это поле пустым.

#### All files are declared in another .APP

Установите этот признак чтобы гарантировать, что файлы будут открываться и закрываться в правильный момент времени, таким образом сохраняя целостность буферов файлов, в случае, когда файлы объявлены в другом приложении (предпочтительней, чем ручной код).

#### File Open Mode

Определяет, как Ваше приложение разделяет файлы между конкурирующими пользователями. Для получения более подробной информации см. Language Reference.

#### Open

Открывает файл как:  
Read/Write (первичный пользователь) +  
Deny Write (все другие пользователи).

#### Share

Открывает файл как :  
Read/Write (первичный пользователь) +  
Deny None (все другие пользователи).

#### Other

Определите необходимую комбинацию для первичного пользователя + доступ остальных пользователей.

User Access Выберите из Read Only, Write Only или Read and Write.

Other Access Выберите из Deny None, Deny All, Deny Read, Deny Write или Any Access (режим совместимости с FCB).

#### Defer opening files until accessed

Определяет, как Ваше приложение открывает связанные файлы. Установите этот признак для задержки открытия файла до тех пор, пока к нему не будет действительно осуществлен доступ. Задержка открытия может увеличить производительность при доступе только к одной из серий связанных файлов.

Очистите этот признак для открытия файла немедленно, как только открывается связанный файл.

Для получения более подробной информации см. File Manager Class— LazyOpen и UseFile.

## **Параметры закладки Индивидуальные Переопределения для Файла (Individual File Overrides)**

---

Выберите закладку Individual File Overrides для переопределения установок словаря данных или установок File Control для конкретных файлов. Выберите файл, атрибуты которого Вы хотите изменить, а затем нажмите кнопку Properties.

Поля ввода на этой закладке - отражение полей с закладки File Control, и ведут они себя точно таким же образом, но с двумя исключениями.

- ◆ Установки, приведенные здесь, применяются только к одному выбранному файлу.

- ◆ Каждый выпадающий список обеспечивает дополнительный выбор: Use Default. Use Default уставляет атрибут согласно закладке File Control.

## **Закладка External Module Options**

---

Выберите закладку External Module Options для установки параметров, связанных с внешними модулями Вашего приложения. Эта закладка доступна только когда Ваше приложение содержит внешний модуль (LIB или DLL). Выберите внешний модуль, атрибуты которого Вы хотите изменить, затем нажмите кнопку Properties.

### Standard Clarion 4 LIB/DLL

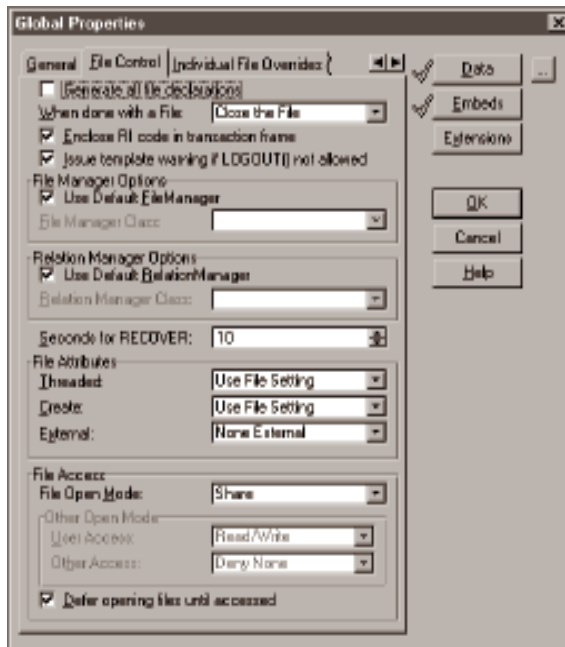
Установите этот признак, если LIB или DLL создан темплейтами ABC или похожей схемой. При установленном признаке генерируется код инициализации и завершения глобальных объектов, используемых в LIB или DLL. Если это вручную написанный LIB или DLL, вероятно, Вы должны будете очистить этот признак.

## **Параметры закладки Classes—Global**

По умолчанию, темплейты ABC плотно связаны с Application Builder Classes. Однако темплейты гибко конфигурируются и разработаны с возможностью (при желании) замещать классы Вашими собственными или классами третьих фирм. Закладка Classes называет и конфигурирует классы ABC, используемые в приложении. Эти глобальные установки могут быть переопределены для каждого темплейта (Procedure, Control или Extension). Для получения более подробной информации об определении классов для каждого темплейта см. Classes Tab Options—Local. Мы настоятельно рекомендуем использование с ABC- темплейтами только ABC-совместимых классов. Для получения более подробной информации см. ABC Compliant Classes.

## Закладка Global Properties Classes

Эта закладка позволяет Вам определить классы, используемые по умолчанию темплейтами ABC для выполнения различных задач. Эта закладка предлагает использовать Вам столько, сколько Вы хотите классов из ABC Library, Ваших собственных или третьих фирм. Вы можете переопределить эти классы (используемые по умолчанию) в закладках Classes для каждого темплейта.



### Refresh Application Builder Class Information

Нажмите эту кнопку, если Вы меняете содержимое или добавляете .INC - файл в директорию \LIBSRC. Обычно это бывает необходимо при установке продуктов третьих фирм, использующих ABC Compliant Classes, хотя Вы и сами тоже можете создавать свои собственные ABC Compliant Classes. Для получения более подробной информации см. ABC Compliant Classes. Темплейты ABC используют информацию, собранную из файлов с заголовками для генерации встроенных точек, загрузки Application Builder Class Viewer, конвертации приложений, и т.д.

### Application Builder Class Viewer

Нажмите эту кнопку, чтобы отобразить классы, свойства и методы, используемые темплейтами ABC и связи между родительскими и

производными (дочерними) классами. Эта утилита может помочь Вам анализировать и понимать классы, которые использует темплейт ABC.

## task grouping buttons

Каждая кнопка группировки задач идентифицирует задачи или типы задач, которые выполняют темплейты ABC. Каждая кнопка предлагает Вам определить класс или классы, используемые темплейтами ABC для выполнения задачи, указанной в тексте на кнопке. Ниже приведены задачи темплейтов ABC и ассоциированные с ними классы по умолчанию.

General	WindowManager ErrorClass PopupClass SelectFileClass WindowResizeClass INIClass TranslatorClass
File Management	FileManager ViewManager RelationManager
Browser	BrowseClass StepLongClass StepRealClass StepStringClass StepCustomClass StepLocator EntryLocator IncrementalLocator FilterLocator FileDropClass FileDropComboClass
Process & Reports	ProcessClass PrintPreviewClass ReportManager
Ascii Viewer	AsciiViewerClass AsciiSearchClass AsciiPrintClass AsciiFileClass



## Toolbar Managers

ToolbarClass  
ToolbarListboxClass  
ToolbarReltreeClass  
ToolbarUpdateClass

Вы можете определить альтернативные классы, набрав имя класса в соответствующем поле ввода. Класс, который Вы укажете, должен быть ABC Compliant Class ( для получения более подробной информации см. ABC Compliant Classes).

## Конфигурирование Default Classes

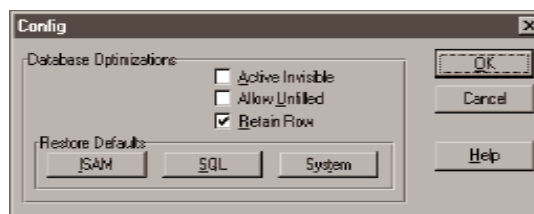
Некоторые классы из задачи General предлагают параметры конфигурации. Это можно определить по присутствию кнопки Configure(сразу под меткой класса). Нажмите кнопку Configure для установки поведения по умолчанию для всех объектов класса во всем приложении. Для переопределения этих глобальных настроек Вы можете встроить кодовый темплейт SetABCProperty для установки соответствующих свойств внутри каждой процедуры.

## Конфигурирование BrowseClass

ABC Application Template обеспечивает следующие поля ввода для настройки конфигурации для BrowseClass:

### Active Invisible

Установите этот признак для заполнения очереди просмотра, даже когда список просмотра LIST “невидим”, так как находится не на выбранной закладке или по другим причинам скрытан. Очистите этот признак для подавления перезаполнения, когда список просмотра невидим. Очистка данного признака увеличит производительность для процедур с невидимыми списками просмотра, однако Вы не должны полагаться на содержимое буфера для невидимого списка просмотра. См. BrowseClass Properties—ActiveInvisible.



### Allow Unfilled

Установите этот признак для разрешения частично-заполненных списков LIST, когда результирующий набор “заканчивается” в середине списка. Это увеличивает (SQL) производительность подавлением добавочных чтений, необходимых для заполнения списка. Очистите признак для того, чтобы всегда отображать “полный” список. См. BrowseClass Properties—AllowUnfilled.

### Retain Row

Установите этот признак для удержания полосы подсветки на том же ряду списка после изменения порядка сортировки, обновления или другого действия, обновляющего список. Это может вызвать провал в производительности при использовании драйвера TopSpeed’s pre-Accelerator ODBC driver. Очистите этот признак, чтобы разрешить перемещение полосы подсветки. См. BrowseClass Properties—RetainRow.

### ISAM

Нажмите эту кнопку для оптимизации конфигурационных полей для файловых систем ISAM.

### SQL

Нажмите эту кнопку для оптимизации конфигурационных полей для систем баз данных SQL.

### System

Нажмите эту кнопку для установки конфигурационных полей таким образом, чтобы дать возможность объекту BrowseClass выбрать наилучшее действие.

## **Конфигурирование WindowManager**

---

ABC Application Template обеспечивает следующие поля конфигурации для класса WindowManager:

### Reset on gain focus

Установите этот признак для безусловной переустановки WindowManager всякий раз, как окно получает фокус.

Очистите этот признак, чтобы разрешить условную переустановку (переустановку, только если этого требуют обстоятельства, например, когда пользователь запрашивает новую сортировку в BrowseBox или обращается к локатору BrowseBox). См. WindowManagerClass Properties—ResetOnGainFocus.

### Auto Tool Bar

Установите этот признак, чтобы WindowManager постарался установить подходящий ToolbarTarget всякий раз, когда пользователь выбирает новую закладку (TAB). Очистите этот признак для ручной установки ToolbarTarget или использования текущего ToolbarTarget. См. WindowManagerClass Properties—AutoToolbar, Toolbar Classes и FrameBrowseControl для получения

более подробной информации.

## **Конфигурирование WindowResizeClass**

---

ABC Application Template обеспечивает следующие поля конфигурации для класса WindowResizeClass:

### Automatically find parent controls

Установите этот признак, чтобы каждый объект Resizer устанавливал parent/child(родительско / дочерние) связи между управляющими элементами окна. Очистка данного признака делает WINDOW “родителем” для всех его управляющих элементов. Установка отношений parent/child предлагает любое особое каскадирование от “родителей” к “детям”. См. WindowResizeClass Methods—SetParentDefaults для получения более подробной информации.

### Optimize Moves

Установите этот признак, чтобы переместить все управляющие элементы во время операции изменения размера окна, провести быстрое изменение размера окна и избежать ошибок в некоторых окнах”. См. WindowResizeClass Properties—DeferMoves для получения более подробной информации.

### Optimize Redraws

Установите этот признак, чтобы сделать управляющие элементы “прозрачными” (с атрибутом TRN) в момент операции изменения размера окна, и чтобы обеспечить более гладкую перерисовку окна и избежать ошибок в некоторых окнах. См. WindowResizeClass Properties—AutoTransparent для получения более подробной информации.

## **Конфигурирование TranslatorClass**

---

ABC Application Template обеспечивает следующие поля конфигурации для класса TranslatorClass:

### Extract Filename

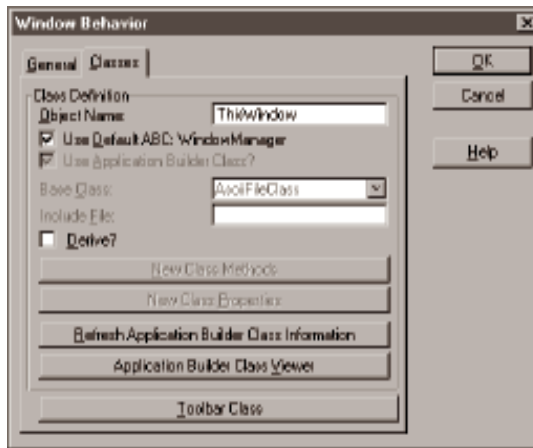
Определите имя файла для получения списка всех текстов, которые могут потребовать перевода для много-языкового приложения. См. See TranslatorClass Properties—ExtractText для получения более подробной информации.

#### Замечание:

Для разрешения этого параметра Вы должны установить признак Enable Run-Time Translation в закладке General.

## Параметры закладки *Classes—Local*

Многие процедуры, управляющие элементами, и расширения темплейтов ABC предлагают диалог или закладку *Classes*. Эти локальные закладки *Classes* предлагают Вам управлять классами (и объектами), которые Ваши процедуры используют для решения задач темплейта. Они переопределяют глобальные установки для классов, определенные в диалоге *Global Properties* (см. *Classes Tab Options—Global*).



Вы можете принять *Application Builder Class* по умолчанию, определенный в диалоге *Global Properties* (рекомендуется), или указать Ваш собственный класс или класс третьих фирм для переопределения установок по умолчанию. Создание Ваших собственных производных классов может дать Вам очень хорошее управление над процедурой, когда стандартный *Application Builder Class* не удовлетворяет в точности Вашим потребностям. Мы строго рекомендуем использование с темплейтами ABC только ABC Compliant классов.

См. *ABC Compliant Classes* для получения более подробной информации.

**Object Name**

Устанавливает метку объекта для сгенерированного темплейтом кода. С помощью тонкой настройки имен объектов Вы можете сделать Ваш код более читаемым.

**Use Default Application Builder Class?**

Установите этот признак для использования *Application Builder Class*, определенный по умолчанию в диалоге *Global Properties* (см. *Template Overview—Classes Tab Options* для получения более подробной информации).

Очистите этот признак, чтобы использовать класс отличный от класса, заданного по умолчанию и чтобы сделать доступными следующие пункты.

### Use Application Builder Class?

Установите этот признак для выбора класса из выпадающего списка Base Class. Этот список включает все ABC Compliant Classes (см. ABC Compliant Classes для получения более подробной информации). Очистите этот признак для использования несовместимого класса (не рекомендуется).  
Base Class

Если Вы установили признак Use Application Builder Class?, выберите класс из выпадающего списка. Если Вы очистили признак Use Application Builder Class?, то наберите здесь имя класса, а в поле Include File введите имя файла, содержащего описание класса.

### Include File

Если Вы очистили признак Use Application Builder Class?, то наберите имя класса в поле Base Class, а здесь введите имя файла, содержащего описание класса.

### Derive?

Установите этот признак для определения производного класса, базирующегося на родительском классе, указанном выше и для включения кнопок New Class Methods и New Class Properties для назначения производному классу новых свойств и методов.

Это поле необходимо, чтобы позволить Вам определить в производном классе новые свойства и методы. Для переопределения существующих методов просто встройте код в соответствующие точки вставки кода для методов.

Использование признаков Derive?, New Class Methods и New Class Properties делает код, генерируемый темплейтом, похожим на следующий:

MyProcessCLASS(Process)	!порождает класс из родительского класса
NewMethod PROCEDURE	!прототип нового метода класса
NewProperty BYTE	!объявляет новое свойство класса
END	

Замечание: Темплейт автоматически определяет производный от родительского класса, если Вы встраиваете Ваш исходный код в любую точку вставки производного метода, вне зависимости от состояния этого признака.

См. See Using ABC Templates to Derive Classes для получения более подробной информации.

### New Class Methods

Нажмите эту кнопку для определения новых прототипов метода для генерации в производную структуру CLASS. Откроется диалог New Class Methods (см. New Class Methods).

## **New Class Properties**

Нажмите эту кнопку для определения новых объявлений свойств для генерации в производную структуру CLASS. Откроется диалог New Class Properties (см. New Class Properties ).

Application Builder Class Viewer

Нажмите эту кнопку для отображения классов, свойств и методов, используемых темплейтами ABC и связей между родительскими и производными классами. Эта утилита может помочь Вам анализировать и понимать классы, используемые темплейтами ABC.

Refresh Application Builder Class Information

Нажмите эту кнопку, если Вы изменили содержимое .INC файла или добавили include - файл в директорию \LIBSRC. Обычно это необходимо, когда Вы устанавливаете продукты третьих фирм, которые используют ABC-совместимые классы, хотя Вы и сами тоже можете создавать свои собственные ABC Compliant Classes. Для получения более подробной информации см. ABC Compliant Classes. Темплейты ABC используют информацию, собранную из файлов с заголовками для генерации встроенных точек, загрузки Application Builder Class Viewer, конвертации приложений, и т.д.

composite Class

Нажмите эти кнопки для открытия диалога Classes для каждого класса, используемого родительским классом, указанным выше. Например, WindowManager использует класс Toolbar, поэтому диалог Classes для WindowManager содержит кнопку Toolbar Class для открытия диалога Classes для его класса Toolbar.

## **New Class Methods**

---

New Class Methods

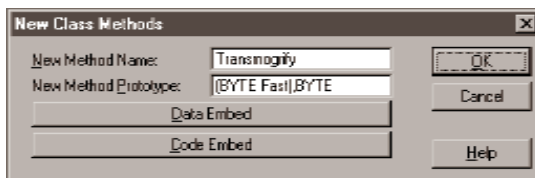
Нажмите эту кнопку для определения прототипов нового метода, для генерации в производную структуру CLASS. Это откроет диалог New Class Methods. Нажмите кнопку Insert для добавления нового прототипа метода и точек вставки, связанных с этим методом.

New Method Name

Введите название (метку) метода

New Method Prototype

Введите список параметров метода и тип возвращаемых данных. Если метод не имеет параметров, но возвращает значение, введите скобки и запятую перед возвращаемым типом данных. Не вводите "PROCEDURE" или "FUNCTION", так как темплейт генерирует для Вас предложение PROCEDURE.



### Data Embed

Нажмите эту кнопку для использования Text Editor для написания раздела данных для метода.

### Code Embed

Нажмите эту кнопку для использования Text Editor для написания раздела кода для метода.

### New Class Properties

Нажмите эту кнопку для определения объявления нового свойства для генерации в структуру CLASS. Это откроет диалог New Class Properties. Нажмите кнопку Insert для добавления нового объявления свойства.

## New Class Properties

---

### Property Name

Введите название (метку) свойства.

### Property Type

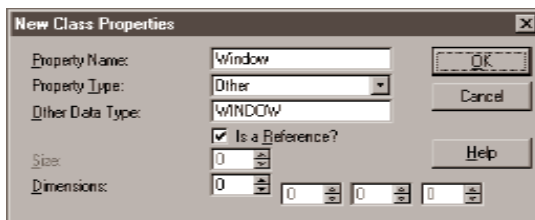
Выберите простой тип данных из списка или выберите Other для включения поля Other Data Type.

### Other Data Type

Введите название(метку) определенного пользователем сложного типа данных (такую, как метки а GROUP, QUEUE или CLASS), или введите разрешенный объектный тип данных (такой как FILE, VIEW или WINDOW).

### Is a Reference

Установите этот признак для объявления переменной-ссылки. Вы должны использовать переменную-ссылку для объектных типов данных и для любого другого сложного типа данных, не разрешенных внутри GROUP. Вы можете использовать переменные-ссылки для любых других типов данных. См. GROUP and Reference Variables в Language Reference.



Size	Определите длину поля в байтах.
Dimensions	Для объявления поля как массива и для определения размеров массива определите размер до четырех размерностей. Общий размер массива не должен превышать 65,520 байт. См. Language Reference для получения более подробной информации о размерных переменных и массивах.

## **ABC Compliant классы**

Классы, используемые с темплейтами ABC должны быть ABC Compliant. То есть классы должны соответствовать спецификациям ABC Library, документированным в Part II—Application Builder Class Library.

Код, генерируемый темплейтами ABC, который ссылается на свойства, методы и на параметры методов документирован во 2-й части этой книги. Если эти свойства, методы и параметры не определены в определенных Вами классах, код, сгенерированный темплейтом не откомпилируется. Далее, если классы не выполняются, как это документировано, код, сгенерированный темплейтом, возможно, не будет работать. Самый легкий путь создания ABC Compliant классов - это создание производных от классов из ABC Library - это как раз то, что делают темплейты ABC. См. CLASS в Language Reference Reference для получения более подробной информации о создании производных классов.

## **Требования для ABC Compliant классов**

- Классы должны соответствовать спецификациям ABC Library, документированным в Part II—Application
  - Builder Class Library
  - Головной файл, содержащий объявления CLASS должен иметь расширение .INC
    - Головной файл (.INC), содержащий объявления CLASS должен находиться в Clarion-директории \LIBSRC
    - Головной файл (.INC), содержащий объявления CLASS должен содержать следующие комментарии перед началом компилируемого кода:  
!ABCIncludeFile
  - Объявления CLASS должны иметь атрибут LINK, называющий соответствующие файлы реализации (.CLW).

Удовлетворение этих требований гарантирует, что Ваши ABC Compliant классы



появятся в Application Builder Class Viewer, Embeditor, диалоге Embedded Source, и что среда разработки имеет полную информацию о Ваших классах. С этой информацией среда разработки может корректно управлять точками вставки кода и генерацией кода для совместимых (compliant) классов.

## **Глобальные ABC точки вставки**

ABC Application Template обеспечивает глобальные точки вставки, чтобы позволить настраивать

- глобальный MAP приложения
- глобальные данные приложения
- инициализацию и завершение программы
- методы обработки файлов (открытие, закрытие, проверку значений, вводимых в поля, заполнение записей, и т.д.) для всех файлов словаря данных
  - файл экспорта (.EXP- известный также как Module Definition file (файл определения модулей) -см. Programmers Guide для получения более подробной информации)
  - ship list приложения

Чтобы получить доступ к этим точкам вставки кода, нажмите кнопку Embeds в диалоге Global Properties, или в Application Tree (дереве приложения) выберите закладку Module, нажмите правую кнопку мыши (RIGHT-CLICK) над модулем Default Program, затем выберите Embeds из всплывающего меню.

В каждой точке вставки Вы можете писать Ваш собственный код, вызывать процедуру, или использовать кодовый темплейт. Генератор приложений при генерации кода разместит Ваш код или вызовы Ваших процедур в строке кода, следующей сразу за точкой вставки, которую Вы выбрали в диалоге Embedded Source. См. Application Generator—Embedded Source Code в User's Guide для получения более подробной информации о добавлении исходного кода в Ваше приложение.

## **Точки вставки кода, относящиеся к файлам**

Глобальные точки вставки кода темплейтов ABC включают точки для FileManager и RelationManager для каждого файла из словаря данных. Вставка кода в эти точки генерирует код для производных методов FileManager и RelationManager, переопределяющих методов родительского класса для индивидуальных файлов словаря данных. См. Manager Methods и Relation Manager Methods для получения более подробной информации об этих методах.

ABC Application темплейт генерирует производные методы FileManager и RelationManager в модули от аррнаBC0.CLW до аррнаBC9.CLW (где аррна - первые пять символов имени файла приложения ). Реальное число сгенерированных модулей зависит от числа файлов в Вашем словаре данных. По умолчанию, темплейты ABC, генерируют код для 20 файлов словаря данных в каждый модуль аррнаBCn.CLW. Вы можете изменить это, изменив значение символа темплейта %FilesPerBCModule.

## **Точки вставки кода, связанные с полями - заполнение(Priming) и проверка (Validation) полей**

Темплейты ABC включают глобальные точки вставки для заполнения каждого поля и для проверки ввода в каждое поле(ошибки до и после ввода).

Вставка кода в эти точки генерирует код для производных методов FileManager и RelationManager для каждого файла словаря данных. См. Manager Methods и Relation Manager Methods для получения более подробной информации об этих методах.

Эти точки вставки обеспечивают одно место в приложении для заполнения и проверки полей. Код, генерируемый темплейтами ABC автоматически вызывает заполнение полей и проверку полей всякий раз, когда Ваше приложение добавляет или изменяет файлы словаря данных.

Замечание:

Эти глобальные точки вставки темплейтов ABC обеспечивают одно место, где Вы можете управлять настройками NULL для SQL-приложений.

## **Использование темплейтов ABC для создания производных классов**

В целях данного обсуждения, упомянем, что создание производных классов означает генерацию структуры CLASS, содержащей объявления данных(свойств) и прототипы методов, плюс генерация соответствующего кода реализации методов. Например:

Whatever PROCEDURE

BRW1	CLASS(BrowseClass)	!производный BRW1 CLASS из BrowseClass
MySwitch	BYTE	!объявление нового свойства BRW1
ResetSort	PROCEDURE,VIRTUAL	!прототип переопределяющего метода
BRW1		
MyMethod	PROCEDURE	!прототип нового метода BRW1

END CODE		!код процедуры
BRW1.ResetSort PROCEDURE CODE PARENT.ResetSort		!определение/реализация метода !некоторый встроенный код !сохранить документированную !функциональность
	!некоторый встроенный код	
BRW1.MyMethod реализация метода CODE	PROCEDURE	! определение/
	!некоторый встроенный код	

### Почему производные классы темплейтов

Производные классы темплейтов ABC производят классы таким образом, что они могут использовать виртуальные методы для настройки поведения объекта порожденного класса по особым требованиям. Виртуальные методы предлагают Вам вставлять код в существующий класс без копирования или дублирования существующего кода. Кроме того, существующий класс вызывает виртуальные методы (содержащие настраивающий код) как часть его нормального функционирования, поэтому Вы не должны явно вызывать их. Затем, когда Topseed обновит существующий класс, обновления автоматически интегрируются в Ваше приложение посредством перекомпиляции. Существующий класс продолжает вызывать виртуальные методы, содержащие настраивающий код, как часть его нормальной работы. Этот подход дает Вам преимущества при настройке Ваших ABC - приложений, в то же время минимизирует затраты на сопровождение.

Темплейты ABC обеспечивают два разных механизма для генерации производных классов.

- Точки вставки кода
- Закладки классов

### Наследование(deriving) с точками вставки кода

Встраивание кода в точке ABC “Method” автоматически генерирует, если необходимо, предложение CLASS, плюс прототип производного класса, плюс код реализации производного класса. Генерируемый код реализации (implementation) включает Ваши точки вставки и вызов соответствующего метода родительского класса. Это гарантирует, что производный метод сохраняет документированную

функциональность родительского метода и обеспечивает дополнительную функциональность встроенного кода. Вы можете удалить функциональность родительского метода, вставляя RETURN перед вызовом методом родительского класса. Если производный метод является VIRTUAL, код, сгенерированный темплейтом, не должен явно вызывать метод, так как объект родительского класса вызывает метод порожденного класса. Однако, если производный метод не является виртуальным, код, генерируемый темплейтом, должен вызывать производный метод или он не будет выполнен. Объект родительского класса вызывает VIRTUAL методы в производном классе. Он не вызывает неvirtуальные методы в производном классе.

Замечание:

Чтобы увидеть, какие методы являются виртуальными, а какие - нет, нажмите правую кнопку мыши на процедуре в дереве приложения, затем выберите Source из выпадающего меню. В Embeditor ищите "VIRTUAL".

См. Application Generator—Embedded Source Code в User's Guide для получения более подробной информации.

### **Наследование(deriving) с закладкой Classes**

Каждый темплейт ABC, который генерирует код для создания экземпляра ABC-объекта, обеспечивает закладку Classes или диалог для того, чтобы помочь Вам создавать производные методы и свойства для его объекта. Установка признака Derive? генерирует, если необходимо, предложение CLASS. Нажатие кнопки New Class Methods предлагает Вам определить новые прототипы методов и код реализации. Нажатие кнопки New Class Methods предлагает Вам определить новые объявления данных (свойства) для генерации внутри структуры CLASS. См. Classes Tab Options—Local для получения более подробной информации.

Для переопределения существующих методов используйте их соответствующие точки вставки кода. См. Deriving with Embed Points для получения более подробной информации.

## ***Мастера(Wizards) и сервисные шаблоны***

Мастера (Wizards) для генерации кода Clarion предоставляет WIZARDS - мощные сервисные темплейты, позволяющие Вам создавать процедуры Browse, Form или Report, отвечая на несколько быстрых вопросов. Вы даже можете использовать wizard, чтобы создавать целостные приложения из существующего словаря базы данных!

### **Парадигма Browse-Form**

Мастера (wizards) генерации кода Clarion следуют парадигме Browse-Form. Парадигма Clarion Browse-Form использует Browsers (окна с сортируемыми, скроллируемыми, с возможностями поиска и выбора списками данных, Формы (окна с единственной, обновляемой базой данных записью), и Отчеты для организации и представления конечным пользователям информации из базы данных. Для справок смотрите Template Overview—Browse-Form Application Paradigm

### **Тонкая настройка Wizards**

Опции, задаваемые Вами заблаговременно в словаре базы данных Clarion, обеспечивают дополнительное управление процессом создания процедур с помощью wizards. Для справок смотрите Optimizing the Wizards.

### **Запуск Wizards**

Чтобы запустить wizard для генерации кода, выберите Application > Template Utility, затем выберите wizard из списка. Иначе: создайте новую процедуру (выберите Procedure > New) и отметьте признак Use Procedure Wizard в диалоге Select Procedure Type.

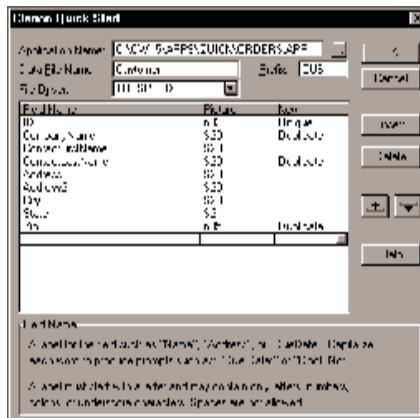
## ***Мастера(Wizard) приложений***

Clarion Application Wizards порождают целостную прикладную программу. Программа включает главное меню и подчиненные процессы поиска, обновления и печати данных из одного или более файлов. Quick Start Wizard порождает словарь данных с одним файлом и программу. Application Wizard генерирует полную программу, базирующуюся на существующем словаре базы данных с одним или более связанных или несвязанных файлов.

## Quick Start Wizard (Быстрый Старт)

Используя Quick Start Wizard, вы сможете создать словарь базы данных и рабочее приложение без необходимости какого-либо кодирования. Обратитесь к Getting started для изучения пошагового примера использования Quick Start Wizard.

Просто определите файл данных, затем Quick Start Wizard создаст завершенное Windows-приложение. Весь процесс займет менее пяти минут! Ваше приложение имеет процедуру form для обновления файла, многоключевую процедуру просмотра browse и столько процедур отчетов report, сколько ключей в файле.



Просто определите поля для единственного файла. Для каждого поля определите имя, формат изображения и информацию о ключе. В результате создастся словарь базы данных. Quick Start Wizard создаст приложение, базирующееся на этом словаре. После определения всех этих опций, кнопка OK генерирует .APP файл и загружает процедуры в диалог Application Tree.

Чтобы использовать Quick Start Wizard:

1. В Clarion выберите File >New >Application .

Открывается диалог New file.

2. Наберите имя .APP файла в поле Filename. Clarion автоматически добавляет расширение .APP.

3. Установите признак Use Quick Start Wizard под списком файлов, затем нажмите кнопку Save. Это иницирует Quick Start Wizard. Этот диалог позволит Вам описать файл, на котором базируется приложение и словарь базы данных. Заполните поля как указано ниже.

### **Application Name**

Имя APP файла. Quick Start Wizard использует то же имя файла (с расширением .DCT) для файла словаря базы данных.

При желании, нажмите кнопку эллипсиса (...), чтобы изменить директорию, затем наберите имя файла в диалоговом окне Open File. Рабочий каталог, в который будут сгенерированы все файлы с исходными текстами зависит от того, где находится .APP файл.

### **Data File Name**

Введите имя файла ( не требуется никакого расширения) для файла данных.

### **Prefix**

Это поле автоматически заполняется первыми тремя буквами имени файла данных, когда Вы переключаетесь с помощью TAB из ячейки Data File Name. При желании задайте в этой области до 14 букв по Вашему выбору.

Префикс позволяет Вашему приложению различать идентичные имена переменных, появляющиеся в различных файлах. Поле с названием Invoice может существовать в одном файле данных с названием Orders и другом (с названием Sales). Устанавливая уникальный префикс для Orders (ORD) и Sales (SAL), приложение может различать два поля как ORD: Invoice и SAL: Invoice.

### **File Driver (драйвер файла)**

Определите тип файла данных. При использовании генератора приложений (Application Generator), Clarion автоматически подключится к правильной библиотеке драйвера базы данных. См. в разделе Database Drivers обсуждение преимуществ каждого драйвера.

Отдельные драйверы файлов могут отличаться в поддержке некоторых параметров, которые Вы добавляете к структуре FILE в этом диалоговом окне.

### **Field Name(Название поля)**

Чтобы назвать каждое поле, введите правильную метку Clarion в поле Name. Правильные имена полей могут немного изменяться в соответствии с драйвером файла.

### **Picture**

Определите формат изображения по умолчанию, набрав его в поле Picture field. Формат изображения вместе с выбранным драйвером файла определяют тип данных,

который Quick Start Wizard использует для поля. Когда Application Generator создает окно и элементы управления report для поля, это используется также как формат изображения по умолчанию для управляющего элемента.

### Key

Это определяет, будет ли создан ключ с использованием данного поля в качестве компоненты, и, если это так, то определение типа ключа. Задав Unique, Вы определяете что Ваше приложение будет проверять уникальность значения каждой записи. Duplicate определяет ключ, который разрешает существование более одной записи с одинаковым значением данной компоненты ключа. Autonumber определяет уникальный числовой ключ, который Ваше приложение автоматически увеличивает на 1 при каждом добавлении новой записи.

Quick Start Wizard создает Browse-список, сортируемый по всем ключам, определенным Вами. Он также создает Report для каждого ключа.

4. Нажмите клавишу TAB для определения следующего поля в Вашем файле. Вы можете использовать командные кнопки для определения полей.

Insert                    Эта кнопка добавляет новое поле перед выбранным полем.

Delete                    Эта кнопка удаляет выбранное поле.



Эта кнопка перемещает выбранное поле на одну позицию вверх в списке полей.



Эта кнопка перемещает выбранное поле на одну позицию вниз в списке полей.

5. Когда Вы определите все поля, нажмите кнопку ОК.

Quick Start Wizard создаст Ваш словарь данных и связанное с ним приложение, затем отобразит Application Tree(дерево приложения).

## **Мастер приложения (Application Wizard)**

---

Application Wizard создает полное приложение из существующего словаря данных. Он создает главную процедуру, содержащую меню с параметрами, вызывающее все созданные подчиненные процессы. Он также создает процедуры Browse, Report и Form для каждого определенного файла.



### Два типа приложений для совместимости

Application Wizard может создавать два различных типа приложений “Полные” приложения для совместимости с Clarion 1.5 - 2.001, и “Простые” приложения, начиная с Clarion Standard Edition 2.002. По умолчанию, в версии Clarion 2.002 и старше Application Wizard создает простые приложения.

Простые приложения меньше, более просты и обеспечивают всю функциональность полных приложений. Простые приложения устраняют Range Limited Browsers, а взамен осуществляют доступ к порожденным файлам через вспомогательные процедуры Browse, Form.

Browsers “Полного” приложения имеют кнопки для доступа ко всем порожденным файлам, равно как и кнопки для доступа к родительским файлам. Эти кнопки связи вызывают Range Limited Browsers.

Вы можете изменить тип приложения по умолчанию, отредактировав ..\TEMPLATE\ABWIZARD.TPL и определив другое значение для %ProgramType. По умолчанию, %ProgramType установлен в ‘Simple’. Чтобы сгенерировать “Полные” (Full) приложения, совместимые с более ранними версиями Clarion, установите %ProgramType в ‘Full’ следующим образом (закомментируйте ‘Simple’ и раскомментируйте объявление ‘Full’):

```
#DECLARE(%ProgramType)
#!SET(%ProgramType,'Simple')    #! Взаимно исключающие опции
#SET(%ProgramType,'Full')
```

### Специальные возможности приложения для Файловых Систем SQL

Для файловых систем, базирующихся на SQL, Application Wizard также генерирует код для сбора информации о начале сеанса пользователя, чтобы затем многократно использовать эту информацию при каждом доступе к файлу.

### Создание приложений - Запуск Application Wizard

Чтобы использовать Application Wizard:

1. В Clarion выбирают File > New > Application.

Открывается диалог New file.

2. Введите имя .APP файла в поле Filename, затем нажмите кнопку Save.

Clarion автоматически добавляет расширение .APP. Не используйте Quick Start Wizard - очистите поле под списком файлов. Откроется диалог Application Proper-

ties, позволяющий Вам определить основные файлы и свойства приложения.

3. Введите в поле Dictionary File название . DCT файла или нажмите кнопку эллипсиса (...) для выбора файла в диалоге Select Dictionary .

4. При желании, переименуйте First Procedure или примите значение по умолчанию - Main.

Это имя процедуры Frame приложения. При желании, Вы можете переименовать его позднее.

5. Выберите из выпадающего списка Destination Type.

Это определяет целевой файл для Вашего приложения.

Выберите Executable (.EXE), Library (.LIB) или Dynamic Link Library (.DLL).

6. При желании, в поле Help File печатают имя файла справок Windows или используют кнопку эллипсиса (...) для выбора файла из диалога Open File.

Если Вы называете файл, то он должен существовать, однако он не обязательно должен представлять собой истинный файл справок Windows. Application Generator позволяет Вам указать темы помощи в Вашем приложении, даже если эти темы не существуют в указанном файле справок. Вы отвечаете за создание файла справок, содержащего строки контекста и ключевые слова, которые Вы при желании вводите как HLP атрибуты для различных элементов управления и диалогов.

7. Подтвердите темплейты ABC template, определенные по умолчанию в поле Application Template. Выбранный шаблон управляет созданием исходного текста. Вы можете выбрать другие шаблоны Clarion или шаблоны третьих фирм, которые Вы зарегистрировали.

8. Подтвердите темплейт по умолчанию ToDo(ABC), определенный в поле ToDo Template.

Выбранный шаблон управляет созданием исходного текста. Вы можете выбрать другие шаблоны Clarion или шаблоны третьих фирм, которые Вы зарегистрировали.

9. Установите признак the Use Application Wizard для использования мастера для создания полного приложения, основанного на выбранном словаре и нескольких ответах, полученных от Вас.

10. Нажмите кнопку ОК. Это создает файл приложения, затем запускает Application Wizard.

Замечание: Для записи поверх существующего приложения, откройте его, затем выберите Application >Template Utility для запуска Application Wizard.

### Использование Application Wizard

1. Ответьте на вопросы в каждом диалоге, затем нажмите кнопку Next.

Application Wizard предлагает ответить на следующие вопросы:

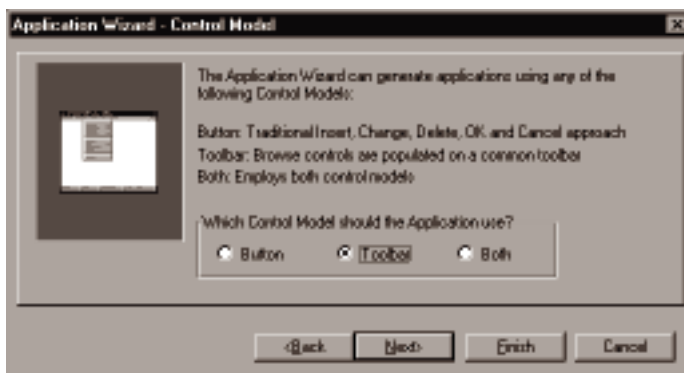
**Generate Procedures for all files in my dictionary** (Генерировать процедуры для всех файлов в моем словаре)

Установите этот признак для всех файлов или очистите его для выбора определенного файла.

**Which control model should the Application use?** (какую управляющую модель должно использовать Приложение?)

Button Wizard строит приложение с традиционными командными кнопками Insert, Change, Delete, OK и Cancel, которые появляются в каждом диалоге.

Toolbar Wizard строит приложение с глобальными toolbar (расположенными на панели инструментов) командными кнопками, которые появляются во фрейме приложения (application frame). Toolbar - кнопки управляют каждым диалогом. См. для справок See Control Templates—FrameBrowseControl.



Both Wizard строит приложение с кнопками обоих видов традиционными диалоговыми и глобальными toolbar-кнопками.

**Overwrite existing procedures** (переписать существующие процедуры)  
Установите этот флаг для замены существующих процедур новыми процедурами с такими же именами.

Очистите этот флаг для сохранения существующих процедур.

**Generate Reports for each file** (генерировать отчеты для каждого файла)  
Установите этот флаг для автоматической генерации процедур report. Очистите этот флаг для пропуска процедур report.

2. В последнем диалоге находится кнопка Finish. Если Вас устраивают данные Вами ответы, нажмите кнопку Finish.

Вы можете нажать кнопку Back для изменения предыдущего выбора или Cancel для отказа от приложения.

Application Wizard создает файл .APP, основанный на словаре и на Ваших ответах, затем отображает диалог Application Tree для Вашего нового приложения.

### **Тонкая настройка Wizard (мастера)**

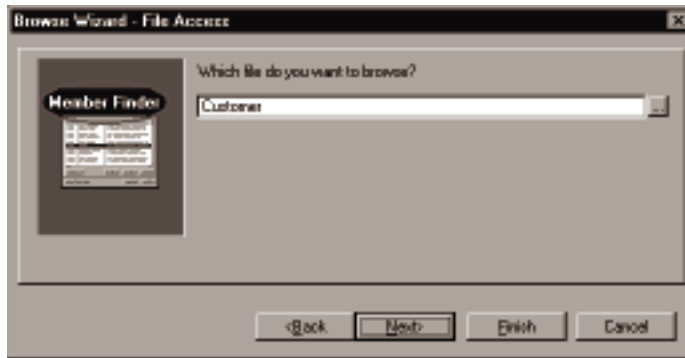
Вы можете управлять тем, как wizard строит Ваше приложение, определяя параметры для файлов, полей, ключей и связей в словаре данных (Data Dictionary) (см Optimizing the Wizards).

## ***Процедурные Мастера (wizards)***

Процедурные Мастера Clarion генерируют одну или несколько процедур, ориентированных на данные (просмотр данных, ввод данных, и отчеты), основываясь на определенных описаниях файлов из словаря данных. Генерируемый код настраивается на определенные файловые связи, включая множественные процедуры, необходимые для поддержки обновления связанных файлов и проверки вводимых данных.

### **Browse Wizard (мастер списков просмотра)**

Этот wizard создает многоключевую процедуру Browse из существующего определения файла в словаре данных. BrowseBox сортируется по каждому определенному Вами ключу. Порядок сортировки управляется с помощью закладок (TAB). Он также создает присоединенную процедуру Form, если Вы определили, что разрешается обновление информации.



Для использования Browse Procedure Wizard:

1. Выберите Application >Template Utility, затем выберите BrowseWizard и перейдите к п.4.

Или:

1. В диалоге Select Procedure установите признак Use Procedure Wizard.

Вы можете открыть диалог Select Procedure, выбрав процедуру ToDo в Application Tree, затем нажав ENTER, или просто нажав клавишу INSERT, затем набрав имя процедуры в диалоге New Procedure.

2. В диалоге Select Procedure, выберите Browse из списка процедурных темплейтов.

3. Нажмите кнопку Select.  
Запустится Browse Wizard.

4. Ответьте на вопросы в каждом диалоге, затем нажмите кнопку Next.  
Browse Wizard предлагает ответить на следующие вопросы:

**What name should be used as the label of the procedure?**

Наберите имя процедуры browse

**Which file do you want to browse?**

Нажмите кнопку эллипсиса (...) для выбора файла из словаря.

**Browse using all record keys**

Установите этот признак, чтобы создать список, сортируемый по всем ключам. Очистите признак, чтобы определить один ключ сортировки.

**Allow the user to update records**

Установите этот признак, чтобы сгенерировать подчиненную процедуру для обновления файла. При желании, обеспечьте имя для процедуры обновления. Очистите данный признак, чтобы сделать список доступным только для чтения.

### Call update using popup menu

Установите этот признак, чтобы обеспечить всплывающие меню(по right-click) для Browse-списка в добавление к всем toolbar и другим командным кнопкам.

### Parent Record Selection

Это приглашение появляется, только если Вы определили один ключ сортировки, являющийся ключом для связи в отношении один ко многим (Many:One). Browse Wizard из этого делает вывод, что Вы желаете просматривать только одну дочернюю запись для определенной родительской записи. Выберите один из следующих пунктов для подтверждения или запрещения этого заключения.

#### Do not select by parent record

Не ограничивать просмотр - другими словами, просматривать все записи.

#### Select parent record via button

Просматривать только дочерние записи для определенной родительской записи. Обеспечьте кнопку для выбора родительской записи.

#### Assume that the parent record is active

Просматривать только дочерние записи для определенной родительской записи. Предполагается, что родительская запись в данный момент активна.

### Provide buttons for child files

Установите этот признак, чтобы обеспечить кнопки в Browse-окне для доступа к связанным дочерним файлам. Иначе: доступ к связанным файлам может осуществляться из сгенерированной update-процедуры.

### Provide a “Select” button

Установите этот признак, чтобы обеспечить кнопку “Select”, которая отображается, когда процедура Browse вызывается для выбора записи, но прячется, когда Browse вызывается для обновления записей.

### Which control model should the Application use?

Какую модель управления использует приложение?

- |                |   |
|----------------|---|
| <b>Button</b>  | Wizard строит browse с традиционными командными кнопками Insert, Change, Delete, OK и Cancel, которые появляются в каждом диалоге.  |
| <b>Toolbar</b> | Wizard строит browse с глобальными toolbar (расположенными на панели инструментов) командными кнопками, которые появляются во фрейме приложения(application frame). Toolbar - кнопки управляют каждым диалогом. См. для справок See Control Templates—FrameBrowseControl. |
| <b>Both</b>    | Wizard строит browse с кнопками обоих видов - традиционными диалоговыми и глобальными toolbar-нопками.  |

**Overwrite existing procedures** (переписать существующие процедуры)  
Установите этот флаг для замены существующих процедур новыми процедурами с такими же именами. Очистите этот флаг для сохранения существующих процедур.

5. В последнем диалоге находится кнопка Finish. Если Вас устраивают данные Вами ответы, нажмите кнопку Finish.

Application Wizard создает процедуру (процедуры), основанную на словаре и на Ваших ответах, затем отображает диалог Procedure Properties для Вашей новой процедуры.

### **Тонкая настройка Wizard (мастера)**

Вы можете управлять тем как wizard строит Вашу процедуру, определяя параметры для файлов, полей, ключей и связей в словаре данных (Data Dictionary) (см Optimizing the Wizards).

### **Form Wizard (мастер форм)**

Этот wizard создает процедуру обновления Form из существующего в словаре описания файла.



Для использования Form Procedure Wizard:

1. Выберите Application >Template Utility, затем выберите FormWizard и перейдите к п.4.

Или:

1. В диалоге Select Procedure установите признак Use Procedure Wizard.

Вы можете открыть диалог Select Procedure, выбрав процедуру ToDo в Application Tree, затем нажав ENTER, или просто нажав клавишу INSERT, затем набрав имя процедуры в диалоге New Procedure.

2. В диалоге Select Procedure, выберите Form из списка процедурных темплейтов.

3. Нажмите кнопку Select.  
Запустится Form Wizard.

4. Ответьте на вопросы в каждом диалоге, затем нажмите кнопку Next. Form Wizard предлагает ответить на следующие вопросы:

**What name should be used as the label of the form procedure?**

Наберите имя процедуры.

**Which file do you want the form to update?**

Нажмите кнопку эллипсис (...) для выбора файла из словаря.

**Allow Records To Be Added**

Установите этот признак, чтобы разрешить добавление новых записей.

**Allow Records To Be Modified**

Установите этот признак, чтобы разрешить изменение записей.

**Allow Records To Be Deleted**

Установите этот признак, чтобы разрешить удаление записей.

**Insert Message**

Введите текст для отображения при добавлении записи.

**Change Message**

Введите текст для отображения при изменении записи.

**Delete Message**

Введите текст для отображения при удалении записи.

**Where do you want this message to be displayed?**

(Где бы Вы хотели отображать сообщение)

Выберите строку заголовка или строку статуса.

**A field can be displayed that identifies the active record.**

(Поле может отображаться, что отображает активную запись)

Нажмите кнопку эллипсис для выбора поля из словаря данных для отображения в строке заголовка окна.

**Validate field values whenever field value changes?**

Установите этот признак, чтобы осуществлять немедленную проверку, когда конечный пользователь подтверждает поле.

**Validate field values when the OK button is pressed?**

Установите этот признак, чтобы включать проверку значения поля по кнопке ОК.

**Browsing child files**

Выберите один из следующих вариантов.

**Place children on tabs**

Размещать дочерние файлы на закладках

**Access children with push buttons**

Осуществлять доступ к дочерним файлам через кнопки

**Do not provide child access**

Не обеспечивать доступ к дочерним файлам

**Which control model should the Application use?**

Какую модель управления использует приложение?

**Button**

Wizard строит диалог с традиционными командными кнопками Insert, Change, Delete, OK и Cancel.



<b>Toolbar</b>	Wizard строит form с глобальными toolbar (расположенными на панели инструментов) командными кнопками, которые появляются во фрейме приложения(application frame). См. для справок See Control Templates—FrameBrowseControl.
<b>Both</b>	Wizard строит form с кнопками обоих видов традиционными диалоговыми и глобальными toolbar-кнопками.

### **Overwrite existing procedures** (переписать существующие процедуры)

Установите этот флаг для замены существующих процедур новыми процедурами с такими же именами. Очистите этот флаг для сохранения существующих процедур.

5. В последнем диалоге находится кнопка Finish. Если Вас устраивают данные Вами ответы, нажмите кнопку Finish.

Form Procedure Wizard создает процедуру (процедуры), основываясь на словаре данных и на Ваших ответах, затем отображает диалог Procedure Properties для Вашей новой процедуры.

### **Тонкая настройка Wizard (мастера)**

Вы можете управлять тем, как wizard строит Вашу процедуру, определяя параметры для файлов, полей, ключей и связей в словаре данных (Data Dictionary) (см Optimizing the Wizards).

## ***Мастер отчетов (Report Wizard)***

Этот wizard создает Report Procedure из существующего в словаре описания файла.

Для использования Report Procedure Wizard:

1. Выберите Application >Template Utility, затем выберите Report Wizard и перейдите к п.4.

Или:

1. В диалоге Select Procedure установите признак Use Procedure Wizard.

Вы можете открыть диалог Select Procedure, выбрав процедуру ToDo в Application Tree, затем нажав ENTER, или просто нажав клавишу INSERT, затем набрав имя процедуры в диалоге New Procedure.

2. В диалоге Select Procedure, выберите Report из списка процедурных темплейтов.

3. Нажмите кнопку Select.  
Запустится Report Wizard.

4. Ответьте на вопросы в каждом диалоге, затем нажмите кнопку Next.  
Report Wizard предлагает ответить на следующие вопросы:

**What name should be used as the label of the report procedure?**

Наберите имя процедуры.

**Which file do you want to report?**

Нажмите кнопку эллипсиса (...) для выбора файла из словаря данных.

**Enter a key below, or leave the field blank to run in record order.**

Нажмите кнопку эллипсиса (...) для выбора ключа сортировки. Оставьте поле пустым для определения отсутствия ключа сортировки.

**How many columns do you want the report to use?**

Введите число колонок в Вашем отчете. Report Wizard размещает поля отчета равномерно по колонкам.

**Overwrite existing procedures** (переписать существующие процедуры)  
Установите этот флаг для замены существующих процедур новыми процедурами с такими же именами. Очистите этот флаг для сохранения существующих процедур.

5. В последнем диалоге находится кнопка Finish. Если Вас устраивают данные Вами ответы, нажмите кнопку Finish.

Report Procedure Wizard создает процедуру (процедуры), основываясь на словаре данных и на Ваших ответах, затем отображает диалог Procedure Properties для Вашей новой процедуры.

### **Тонкая настройка Wizard (мастера)**

Вы можете управлять тем, как wizard строит Вашу процедуру, определяя параметры для файлов, полей, ключей и связей в словаре данных (Data Dictionary) (см Optimizing the Wizards).

### ***Мастер печати словаря данных (Dictionary Print Wizard)***

Этот wizard печатает описания файлов из словаря данных (dictionary) с различными уровнями детализации для файлов, полей, ключей и связей. Вы можете печатать на принтер или в файл.

Для использования Dictionary Print Wizard:

1. Откройте приложение, которое использует словарь данных.
2. Выберите из меню Application > Template Utility

Откроется диалог Select Utility.

3. Подсветите DictionaryPrint, затем нажмите кнопку Select.

Запустится Dictionary Print Wizard.

4. Ответьте на вопросы в каждом диалоге, затем нажмите кнопку Next.

После первого диалога станет доступной кнопка Finish. Теперь нажмите кнопку Finish для печати всей доступной информации обо всех файлах, полях, ключах и связях.

Или пройдите через диалоги wizard для выбора определенных файлов, плюс уровни детализации для печати (All(все), Some(некоторые) или None(ничего)) для различных компонент словаря.

## **Оптимизация мастера (wizard)**

Wizard Options(параметры мастера) в Data Dictionary Editor обеспечивают большее управление функциональностью мастера (wizard). Мастера при создании процедур используют параметры, определяемые для файла, поля, ключа или алиаса. Вдобавок, wizard использует имена файлов, полей, ключей и алиасов и описания для текстов в меню, строках заголовков, закладках и т.д.

### **File Options (Параметры файлов)**

---

Do Not Auto-Populate This File

Указывает Мастеру, что при создании процедур Browse или Report необходимо пропустить этот файл.

User Options (Параметры пользователя)

Параметры пользователя обеспечивают информацию для сервисных темплейтов(utility templates). Параметры пользователя разделяются запятыми. Выберите один из следующих вариантов:

EDITINPLACE

Browse Wizard обеспечивает обновление “по месту” для просматриваемых файлов вместо отдельной процедуры обновления (form). Мы рекомендуем этот параметр для файлов с односторонними lookup связями, такими как файл State Code. Файлы со сложными связями лучше управляются через отдельную процедуру обновления.

## Параметры для алиаса (Alias Options)

---

### Do Not Auto-Populate This Alias File

Указывает Мастеру, что при создании процедур Browse или Report необходимо пропустить этот Alias файл.

### User Options (Параметры пользователя)

Параметры пользователя обеспечивают информацию для сервисных темплейтов (utility templates). . User Options разделяются запятыми.

## Параметры поля (Field Options)

---

### Do Not Auto-Populate This Alias File

Указывает Мастеру, что при создании процедур Browse или Report необходимо пропустить это поле.

### Population Order

Определяет порядок, в котором поля размещаются Мастером (wizard). Выберите Normal, First или Last из выпадающего списка. Мастер размещает в следующем порядке: все поля, определенные как First, затем все определенные как Normal и, наконец, поля, определенные как Last.

### Form Tab

Определяет закладку (TAB), на которой мастер размещает поля. Введите заголовок (Caption) для TAB или выберите из выпадающего списка один из приготовленных Вами ранее. Это позволит Вам указать Мастеру группировать поля тем способом, которым Вы хотите.

### Add Extra Vertical Space Before Field Controls on Forms

Установите этот признак, чтобы указать Мастеру (wizard) добавлять вертикальный отступ между управляющим элементом для этого поля и размещенного перед ним.

### User Options (Параметры пользователя)

Параметры пользователя обеспечивают информацию для сервисных темплейтов (utility templates). User Options разделяются запятыми.

## Параметры ключа (Key Options)

---

### Do Not Auto-Populate This Alias File

Указывает Мастеру, что при создании процедур Browse или Report необходимо пропустить этот ключ.

## Population Order

Определяет порядок, в котором ключи размещаются Мастером (wizard). Выберите Normal, First или Last из выпадающего списка. Мастер размещает в следующем порядке: все ключи, определенные как First, затем все определенные как Normal и, наконец, ключи, определенные как Last.

## User Options (Параметры пользователя)

Параметры пользователя обеспечивают информацию для сервисных темплейтов(utility templates). User Options разделяются запятыми.

## **Relation Options (Параметры связи)**

---

## User Options (Параметры пользователя)

Параметры пользователя обеспечивают информацию для сервисных темплейтов(utility templates). User Options разделяются запятыми.

## **Соглашения по названиям**

---

При создании процедур Мастера(wizards) извлекают информацию из Вашего словаря данных (Data Dictionary) и применяют ее для генерации процедур. Понимание того, как Мастера используют информацию из словаря данных, может помочь Вам настроить Ваш словарь для достижения наилучших результатов.

### **Naming Fields and Keys (Именованние полей и ключей)**

Мастера (wizards) используют имя поля из словаря данных для формирования приглашений в окнах и отчетах. Если Вы используете имена со смешанными регистрами (например, FirstName), Мастера добавляют пробелы перед заглавными буквами для создания многословных приглашений и заголовков - в нашем случае - First Name.

Мастер Browse использует описание ключа в качестве текста закладки в browse-процедурах с несколькими ключами. Если описание отсутствует, Мастер использует название ключа.

### **Field Descriptions (Описание полей)**

По умолчанию, описания полей присваиваются атрибуту MSG в словаре. Мастера автоматически применяют атрибут MSG к каждому управляющему элементу в Вашем приложении, так что описание отображается в строке состояния приложения. Обеспечение описаний полей в словаре (один раз) устраняет необходимость определять (потенциально) а Вашем приложении несколько атрибутов MSG.

## **Использование управляющих элементов окна, определенных по умолчанию**

---

Редактор Словаря Данных (Dictionary Editor) создает управляющий элемент по умолчанию, основываясь на его типе данных. См. закладки Window и Report в диалоге Field Properties. Мастера используют эти управляющие элементы по умолчанию при создании процедур.

Например, LONG становится элементом управления ENTRY. Иногда Вы можете захотеть установить другой тип управляющего элемента. Например, если номер покупателя имеет тип LONG и автоматически увеличивается, то Вы никогда не захотите, чтобы пользователь модифицировал его. В этом случае, Вы можете установить, чтобы тип управляющего элемента по умолчанию был STRING.

Другим примером является поле, которое имеет ограниченный набор возможных значений. В этом случае, Вы можете указать Drop List как тип по умолчанию и определить правильные возможные значения в Validity Checks.

## Процедурные шаблоны

### Краткий обзор

В этой главе описываются Процедурные шаблоны Clarion. Упоминаются здесь также несколько шаблонов Элементов управления, которые рассматриваются в следующей главе.

### Процедуры и Процедурные шаблоны

---

*Процедура* представляет собой последовательность операторов языка Clarion (исходный код), которые выполняют поставленную задачу. *Процедурный Шаблон* - это *диалоговый инструмент*, который (с помощью среды разработки Clarion'a) запрашивает информацию у Вас как у разработчика, а затем генерирует программную процедуру именно для той задачи, которую Вы задали. Процедура, сохраненная в файле приложения Clarion (.app), как раз и является той информацией (иначе говоря, Спецификацией процедуры), которую среда разработки использует для генерации исходного кода этой процедуры. Спецификация включает в себя Процедурный Шаблон и Ваши ответы на его пункты диалога, описание Окна, Отчета и прочее (объявления локальных данных, внедренный исходный текст и т.д.).

Clarion предлагает богатый ассортимент задачно-ориентированных Процедурных шаблонов, с помощью которых Вы можете быстро разрабатывать приложения для работы с базами данных. В *Первоначальном Знакомстве*, *Обучающая программа Быстрого Старта* представляет несколько процедурных шаблонов; еще больше возможностей демонстрирует *Обучающая программа Генератора Приложения в Изучении Clarion*. В этой главе описываются все процедурные шаблоны а также предлагаемые ими пункты диалога.

### Использование Процедурных шаблонов

Использование процедурных шаблонов происходит путем выбора того из них, который наиболее подходит для решения требуемой общей задачи, такой как просмотр или поиск данных (шаблон просмотра (Browse)), изменение данных (шаблон Формы (Form)) или отчет по данным (шаблон Отчета (Report)). Шаблон выбирается при создании процедуры (см. *Генератор Приложения в Руководстве Пользователя* для получения дополнительной информации). Уже после этого

генерируемый с помощью шаблонов код настраивается для соответствия определенной Вами задаче с помощью диалога **Свойств Процедуры (Procedure Properties)**. Здесь необходимо ответить на ряд пунктов диалога, а также отсюда возможен доступ к другим инструментам среды разработки, таким как Конструктор Окна или Конструктор Отчета.

### Диалог Свойства Процедуры

Диалог **Свойства Процедуры (Procedure Properties)** содержит стандартные командные кнопки и пункты диалога Генератора приложения, а также дополнительные пункты диалога (предложения), предоставленные конкретным Процедурным шаблоном. В этой главе описываются пункты диалога, генерируемые шаблоном. См. *Генератор приложения в Руководстве Пользователя* для получения дополнительной информации о командных кнопках и пунктах диалога Генератора приложения.

### Модель Просмотр-Редактирование

Процедурные шаблоны Clarion'а придерживаются модели приложения типа Форма-Просмотра/Форма-Редактирования. В этой модели применяются Окна просмотра (окна со списками, которые могут быть отсортированы, проскроллированы, по которым можно организовать поиск и из которых можно выбрать данные), Формы (окна с единственной редактируемой записью из базы данных) и Отчеты, предназначенные для организации и представления конечным пользователям информации из базы данных. См. *Обзор Шаблона – Модель Приложения Просмотр-Редактирование* для получения дополнительной информации.

### Процедуры как Контейнеры

Процедуры могут содержать такие структуры данных, как WINDOW, REPORT, а также в пределах этих структур - элементы управления. Кроме того, Процедурные шаблоны могут содержать другие шаблоны — шаблоны Элементов управления и шаблоны Расширения, которые предоставляют дополнительные возможности по настройке процедуры.

### Процедуры содержат Элементы управления

Процедурные шаблоны обеспечивают стандартные пункты диалога для любого элемента управления типа BUTTON, ENTRY или CHECK, которые были размещены на окне процедуры. Доступ к пунктам диалога для каждого из этих элементов управления Вы получаете через его собственный диалог Свойств. Например, окно Свойств каждого элемента управления ENTRY процедурный



шаблон обеспечивает пунктами диалога, которые позволяют использовать его как поле поиска. В то же время, окно Свойств для элемента управления СНЕСК с помощью процедурного шаблона снабжается пунктами диалога, позволяющими обновлять переменные, а также скрывать или отображать тот или иной элемент управления в зависимости от состояния поля СНЕСК.

Процедурные шаблоны предлагают стандартные точки вставки для всех помещаемых на окно процедуры элементов управления. В общем случае, для каждого события, которое может произойти на этом элементе управления, существует своя точка вставки. Код, вставленный в эти точки вставки, выполняется тогда, когда на элементе управления произошло соответствующее событие. См. *Точки Вставки ABC Шаблона* для получения дополнительной информации.

### **Процедуры содержат Другие Шаблоны**

Наконец, Процедурные шаблоны могут содержать и другие шаблоны — шаблоны Элемента управления и шаблоны Расширений, которые предлагают свои собственные пункты диалога со средой разработки, а также свои точки вставки и свои функциональные возможности во время выполнения.

Таким образом, Процедура и ее шаблон действуют как контейнер, который автоматически обеспечивает поддержку большого количества слоев функциональных возможностей и настройки. Тогда и Приложение, сохраненное в среде разработки, действует как контейнер для Процедур и их шаблонов.

Многие из ABC Процедурных шаблонов уже содержат шаблоны Элемента управления. Шаблоны Элемента управления генерируют код, предназначенный для того, чтобы определить конкретный элемент управления и управлять им, включая загрузку и выгрузку данных в/из элемента управления. Фактически, уникальный набор шаблонов Элемента управления в пределах Процедурного шаблона и есть то, что определяет основную цель или задачу шаблона. Например, Шаблон Процедуры Просмотра является настраиваемым Процедурным шаблоном Окна, который содержит шаблоны Окна Просмотра (BrowseBox) и шаблоны Элементов управления (BrowseUpdateButtons).

### ***Процедурные шаблоны Окна***

Большинство ABC Процедурных шаблонов (Окно приложения (Frame), Окно просмотра (Browse), Форма редактирования (Form) и т.д.) генерируют процедуры, содержащие окна. Даже для таких шаблонов, как Процесс (Process) и шаблоны

Отчета (Report), определено окно отображения хода выполнения процесса. **Все шаблоны оконных процедур, описанные в этой секции, получены из Шаблона Окна и наследуют его пункты диалога, предназначенные для управления основным поведением процедуры.** Доступ к пунктам диалога, общим для оконной процедуры, можно получить через диалог **Свойств Процедуры (Procedure Properties)** для каждого процедурного шаблона.

## **Шаблон Окна (Window Template)**

---

Этот шаблон функционирует как чистая подложка, на которой можно создать свою собственную оконную процедуру любого вида. Большинство Процедурных АВС шаблонов получено из этого шаблона, и поэтому наследует его поведение и пункты диалога.

Для выбора типа окна нажмите кнопку **Окно (Window)** в диалоге **Свойства Процедуры (Procedure Properties)**. См. *Руководство Пользователя — Конструктор Окна — Выбор Типа Окна* для получения дополнительной информации.

Для элементов управления и шаблонов Элементов управления, которые Вы добавляете к окну, шаблон Окна добавляет точки вставки, предназначенные для обработки генерируемых ими событий. После того, как на окне будут размещены элементы управления, с помощью кнопок **Вставка (Embeds)** и **Исходный код (Source)** можно применять для обработки событий свой собственный программный код.

Только лишь “стандартные” элементы шаблона, доступ к которым Вы можете получить с помощью диалога **Свойства Процедуры (Procedure Properties)**, являются локальными переменными, используемыми для передачи данных в/из вызывающей процедуры, а также для управления окном и процедурой, отслеживая, является ли окно открытым, и должна ли процедура «ответить» на глобальное событие.

Код, сгенерированный этим шаблоном обрабатывает структуру WINDOW, созданную с помощью Конструктора Окна (Window Formatter). При этом генерируется код, предназначенный для обработки всех полей и событий на окне.

**Совет:** Чтобы дублировать окно, созданное для другого приложения или процедуры, не копируя всю процедуру целиком, необходимо скопировать описание структуры WINDOW из секции исходного кода, где описывается окно, а затем, нажав эллиптическую кнопку (...) рядом с кнопкой **Окно (Window)**, вставить содержимое буфера. Предостережение: Эта операция недопустима для окон, содержащих шаблоны Элемента управления!

## Пункты диалога Шаблона Окна



В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения* в *Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** Процедурного шаблона Окна содержатся следующие пункты диалога, унаследованные также всеми остальными Процедурными шаблонами Окна:

### **Параметры (Parameters)**

Задайте список параметров Вашей процедуры. См. *PROCEDURE* и *Прототипы Процедуры* в *Описании Языка* а также *Прототипирование и Пропуск Параметров* в *Руководстве Пользователя* для получения дополнительной информации. Список параметров представляет собой необязательный список типов данных и меток, которые впоследствии появятся в сгенерированном шаблонами операторе PROCEDURE. Весь список заключается в круглые скобки. Для каждого параметра, определенного в прототипе процедуры, должен присутствовать параметр в списке параметров. Рекомендуется определять тип данных *и* метку параметра, как в списке параметров, так *и* в прототипе процедуры.

Например: (SHORT Id,STRING Name)

Параметры должны быть обработаны в исходном коде процедуры.

### **Возвращаемое значение (Return Value)**

Задайте имя переменной, значение которой возвращается процедурой. К этому моменту уже должен быть определен тип возвращаемых данных в поле **Прототип (Prototype)**. См. *Прототипирование и Пропуск Параметров* в *Руководстве Пользователя* для получения дополнительной информации. Присвоение же этой переменной соответствующего значения должно содержаться в исходном коде процедуры.

## Поведение Окна (Window Behavior)

Кнопка **Поведение Окна (Window Behavior)** обеспечивает доступ к окну диалога с несколькими закладками, предоставляющими возможность задания опций как для Окна этой процедуры, так и для его WindowManager.

### **Режим Работы Окна (Window Operation Mode)**

Позволяет переопределять некоторые свойства окна, заданные в Конструкторе Окна. Этот пункт диалога предоставляет возможность быстрого изменения этих атрибутов без использования Конструктора Окна. Возможные варианты:

#### **Использовать Установки Окна (Use Window Setting)**

Использовать атрибуты, установленные в Конструкторе Окна.  
Нормальный (Normal)

Снимает с Окна атрибуты MDI и MODAL.  
**MDI** Добавляет Окну атрибут MDI.  
**Modal** Добавляет Окну атрибут MODAL.

#### **Сохранять и Восстанавливать Расположение Окна (Save and Restore Window Location)**

Если необходимо, чтобы размер окна процедуры и его местоположение сохранялись из предыдущей рабочей сессии, взведите этот флажок. Однако сначала Вы должны взвести флажок **Использовать INI-файл для сохранения и восстановления программных установок (Use INI file to save and restore program settings)** в окне диалога **Глобальные Свойства (Global Properties)**. См. *Обзор Шаблона — Пункты Диалога на Общей Закладке*.

#### **Классы (Classes)**

Закладка **Классы** позволяет управлять используемым в процедуре классом WindowManager (и его объектом). Можно либо использовать предлагаемый по умолчанию Класс Разработки Приложений (ABC) и его объект (рекомендуется), либо выбрать другой ABC-класс из выпадающего списка, либо определить свой собственный класс.

См. *Обзор Шаблона — Пункты Диалога Закладки Классов — Локальный* для получения полной информации об этих опциях.

## **Шаблон Окна просмотра (Browse Template)**

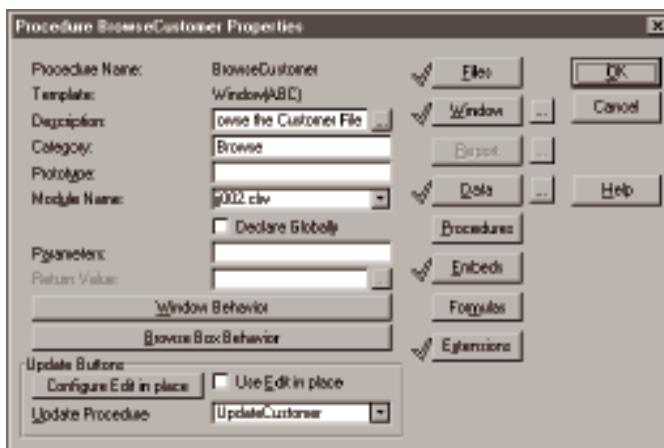
Шаблон Окна просмотра (Browse) получен из Шаблона Окна. Он генерирует процедуру, предназначенную для просмотра, скроллинга, поиска, и управления данными. Данные при этом могут содержаться в одном или нескольких связанных

файлах, а Процедура Окна просмотра может либо сама редактировать данные, либо вызывать для этого отдельную процедуру. Шаблон Окна просмотра в части, касающейся его функциональных возможностей, во многом зависит от шаблона Элемента управления BrowseBox (см. *Шаблоны Элемента управления* для получения дополнительной информации). Диалог **Определение Схемы Файлов (File Schematic Definition)** автоматически применяет составленную Вами схему файлов к шаблону Элемента управления BrowseBox. Сгенерированный код реализует поиск связанных записей.

### **Модель Просмотр-Редактирование (Browse-Form Paradigm)**

Шаблон Окна просмотра является неотъемлемой частью модели Просмотр-Редактирование Clarion'a, которая использует Окна просмотра (окна со списками, которые могут быть отсортированы, проскроллированы, по которым можно организовать поиск и из которых можно выбрать данные), Формы (окна с единственной редактируемой записью из базы данных) и Отчеты для организации и представления конечным пользователям информации из базы данных. См. *Обзор Шаблона – Модель Приложения Просмотр-Редактирование* для получения дополнительной информации.

### **Пункты диалога Шаблона Окна Просмотра (Browse Template Prompts)**



В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** шаблона Окна просмотра содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение и Поведение Окна** – смотри *Шаблоны Процедуры Окна*

— *Пункты Диалога Шаблона Окна*), плюс пункты диалога, необходимые для настройки таких элементов управления, как Окно просмотра (BrowseBox) и кнопки редактирования записей из Окна просмотра (BrowseUpdateButtons):

### **Поведение Окна Просмотра (Browse Box Behavior)**

Эта кнопка обеспечивает доступ к окну диалога с несколькими закладками, предоставляющими возможность задания опций для шаблона Элемента управления Окно просмотра. Отсюда можно управлять реализованным в Окне просмотра поиском записей, их скроллингом, выбором, подсчетом итоговых значений, цветами, иконками и т.д. См. *Шаблоны Элемента управления—Окно просмотра* для получения полного описания этих пунктов диалога.

### **Кнопки Модернизации**

Шаблон Элемента управления Кнопки Модернизации Окна просмотра (BrowseUpdateButtons) обеспечивает наличие дополнительных пунктов диалога. С их помощью определяется, редактируются ли записи прямо в Окне просмотра, или для этого вызывается отдельная процедура. См. *Шаблоны Элемента управления — BrowseUpdateButtons* для получения полного описания этих пунктов диалога.

### **Шаблон Формы (Form Template)**

Шаблон Формы получен из Шаблона Окна. Он генерирует код, предназначенный для отображения и модернизации единственной записи из файла. Реализовано также отображение и редактирование соответствующих записей в других связанных файлах.

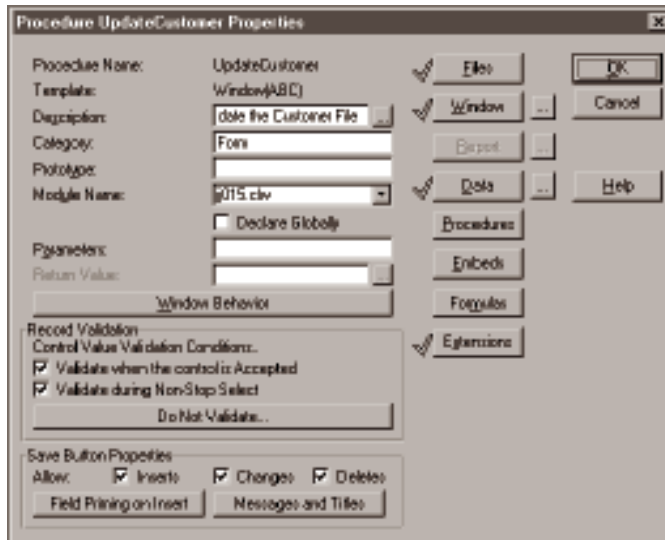
Шаблон Формы предлагает стандартное окно с шаблоном Элемента управления SaveButton и шаблоном Расширения ValidateRecord. Шаблон Элемента управления SaveButton обрабатывает ввод/вывод из файла, а шаблон ValidateRecord сверяет поступающие данные с их описаниями из словаря данных. Диалог **Определение Схемы Файлов (File Schematic Definition)** автоматически применяет составленную Вами схему файлов к шаблону Элемента управления SaveButton. Для доступа к связанным записям шаблон Формы опционально предусматривает использование шаблона Элемента управления BrowseBox. См. *Шаблоны Элемента управления и Шаблоны Кода и Расширения* для получения детального описания пунктов диалога и функциональных возможностей этого шаблона.

### **Модель Просмотр-Редактирование (Browse-Form Paradigm)**

Шаблон Формы является неотъемлемой частью модели Просмотр-Редактирование Clarion'a, которая использует Окна просмотра (окна со списками, которые могут быть отсортированы, проскроллинрованы, по которым можно

организовать поиск и из которых можно выбрать данные), Формы (окна с единственной редактируемой записью из базы данных) и Отчеты для организации и представления конечным пользователям информации из базы данных. См. *Обзор Шаблона – Модель Приложения Просмотр-Редактирование* для получения дополнительной информации.

### Пункты диалога Шаблона Формы (Form Template Prompts)



В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** шаблона Формы содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение и Поведение Окна** – смотри *Шаблоны Процедуры Окна – Пункты Диалога Шаблона Окна*), плюс пункты диалога, необходимые для настройки таких шаблонов, как шаблон Расширения Проверка пригодности записи (ValidateRecord) и шаблон Элемента управления SaveButton.

### Проверка Пригодности Записи (Record Validation)

Шаблон Расширения ValidateRecord добавляет дополнительные пункты диалога, с помощью которых можно управлять тем, как и когда происходит проверка пригодности записи. См. *Шаблоны Кода и Расширения – Проверка Пригодности Записи* для получения полного описания их пунктов диалога.

## Свойства Кнопки «Сохранить» (Save Button Properties)

Шаблон Элемента управления SaveButton добавляет дополнительные пункты диалога, с помощью которых можно управлять тем, как и когда происходит обновление записи. Также он предоставляет возможность выбирать тип разрешенного обновления, разрешать или нет множественные вставки, управлять показываемыми конечному пользователю сообщениями и т.д. См. *Шаблоны Элемента управления — SaveButton* для получения полного описания их пунктов диалога.

## Шаблон Окна приложения (Frame Template)

Этот шаблон обеспечивает родительскую структуру MDI-интерфейса (Интерфейс Множества Документов), которая содержит определенное стандартом Windows меню со стандартными командами управления файлом, редактированием, окнами, получением помощи.

## Модель Просмотр-Редактирование (Browse-Form Paradigm)

Шаблон Окна приложения является неотъемлемой частью модели Просмотр-Редактирование Clarion'a, которая использует Окна просмотра (окна со списками, которые могут быть отсортированы, проскроллированы, по которым можно организовать поиск и из которых можно выбрать данные), Формы (окна с единственной редактируемой записью из базы данных) и Отчеты для организации и представления конечным пользователям информации из базы данных. См. *Обзор Шаблона — Модель Приложения Просмотр-Редактирование* для получения дополнительной информации.

При создании MDI приложения Окно приложения должна быть главной процедурой, которая управляет всеми прочими процедурами Вашего приложения. Для каждого дочернего MDI окна, которое должно появляться внутри Окна приложения, необходимо начинать новый исполняемый процесс. Закладка **Действия (Actions)** для элемента управления или Пункта Меню предлагает поле флажка, взведя который, можно задать начало нового исполняемого процесса (с этой целью может быть также использован шаблон Кода «Начать Процесс»(InitiateThread)).

Стандартное окно содержит стандартное Windows-меню со следующими пунктами:

<b>Файл (File)</b> —	Настройка печати (Print Setup)и Выход (Exit);
<b>Редактирование (Edit)</b> —	Вырезать (Cut), Копировать (Copy) и Вставить (Paste);
<b>Окно (Window)</b> —	Выстроить (Tile), Расположить каскадом (Cascade) и Выровнять значки (Arrange Icons);
<b>Помощь (Help)</b> —	Содержание (Contents), Как Пользоваться Помощью (How

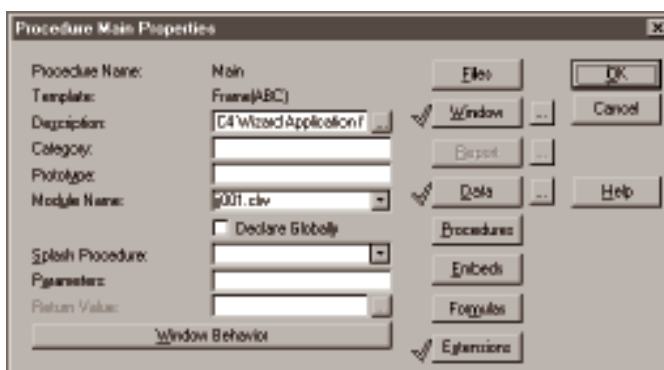


to Use Help) и Поиск (Search for Help on).

Каждая из стандартных команд меню реализует Стандартное Поведение Окон. Исполняемые Библиотеки Clarion'a автоматически обеспечивают это стандартное поведение, так что для этих команд меню писать дополнительный код не придется.

Шаблон Окна приложения включает в себя точки вставки: стандартные плюс дополнительные - для команд меню. Если же добавить Панель Управления, то точки вставки будут добавлены также для любого принадлежащего ей элемента управления.

### Пункты диалога Шаблона Окна приложения (Frame Template Prompts)



В дополнение к стандартным командным кнопкам Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** Процедурного шаблона Окна приложения содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение** и **Поведение Окна** — смотри *Шаблоны Процедуры Окна* — *Пункты Диалога Шаблона Окна*), а также:

#### **Процедура-заставка (Splash Procedure)**

Определяет процедуру, которая будет открыта сразу после открытия главного окна приложения, но прежде, чем произойдет любое инициированное пользователем событие. Имя процедуры либо выберите из выпадающего списка, либо напечатайте новое.

В соответствии с принятым соглашением, процедура-заставка сопровождает начало программы визуальными, звуковыми, или и теми, и другими эффектами. На экране заставки может быть размещен легко узнаваемый логотип или изображение, предназначенные как для поднятия уровня комфорта пользователя (вследствие узнавания привычного образа), так и в рекламных целях. Кроме того, это позволяет

отвлечь внимание пользователя от подчас скучной процедуры загрузки и инициализации программы.

См. *Шаблон Заставки* для получения дополнительной информации.

### **Отображение Даты и Времени (Date and Time Display)**

Эта кнопка обеспечивает доступ к окну диалога с несколькими закладками, предоставляющими возможность задания опций для шаблона Расширения Отображение Даты и Времени (DateTimeDisplay). Шаблон DateTimeDisplay делает возможным отображение Даты, Времени, или и того, и другого как на панели статуса окна, так и на элементе управления. См. *Шаблоны Кода и Расширения – Отображение Даты и Времени* для получения полного описания этих пунктов диалога.

### **Шаблон Меню (Menu Template)**

Этот шаблон предлагает окно интерфейса SDI (Интерфейс Одного Документа). Он действует подобно Шаблону Окна Приложения в части, касающейся генерации меню, являющегося исходной точкой SDI приложения. В дополнение к стандартным командным кнопкам Генератора приложения (см. *Генератор Приложения в Руководстве Пользователя*) диалог **Свойства Процедуры (Procedure Properties)** Процедурного шаблона Меню содержит только унаследованные от Шаблона Окна пункты диалога (см. *Процедурные шаблоны Окна – Пункты Диалога Шаблона Окна*).

### **Шаблон Процесса (Process Template)**

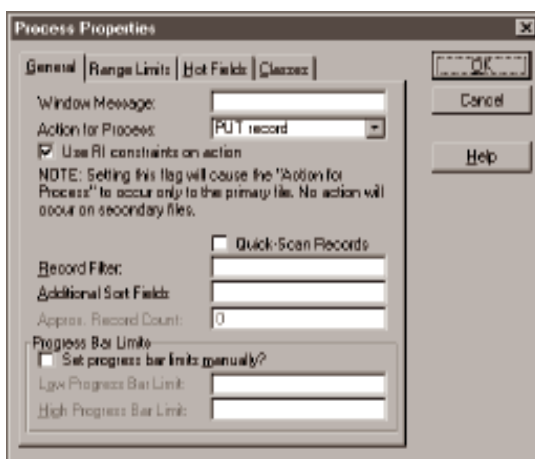
Процедурный шаблон Процесс генерирует код, предназначенный для просмотра всего файла с данными и произведения с каждой записью какой-либо операции. Здесь можно также задать фильтр или ограничивающий диапазон значений, подлежащих обработке. Стандартное окно содержит индикатор хода процесса, который показывает конечному пользователю процент выполнения операции. Использование шаблона элемента управления PauseButton предоставляет конечному пользователю возможность приостанавливать и возобновлять работу ProcessTemplate.

### **Пункты диалога Шаблона Процесса (Process Template Prompts)**

В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** Процедурного шаблона Процесса содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение и Поведение Окна** – смотри *Шаблоны Процедуры Окна – Пункты Диалога Шаблона Окна*), а также:

Свойства Процесса (Process Properties)

Эта кнопка обеспечивает доступ к окну диалога с несколькими закладками, предоставляющими возможность определения широкого спектра функциональных возможностей для этой процедуры Процесса. В этом разделе описывается окно диалога **Свойства Процесса (Process Properties)**.



**Совет:** По умолчанию, процедурный шаблон Процесса не производит ни создания новых записей, ни автонумерации существующих полей или записей.

## Общая (General)

### Сообщение в окне (Window Message)

Текст, который будет отображен в окне прогресса.

### Действия по обработке (Action for Process)

Этот пункт диалога позволяет задать вид действий, производимых при обработке записей: изменение (PUT) или удаление. Для выполнения любой другой необходимой обработки достаточно поместить в точку вставки **Действие над каждой Записью (Activity for each Record)** соответствующий программный код.

### Использовать RI ограничения на действия (Use RI constraints on action)

Взведите этот флажок для соблюдения заданных в словаре данных RI ограничений. Если этот флажок снять, то будет генерироваться простой PUT или DELETE в зависимости от выбранного **Действия по обработке (Action for Process)**.

### Быстрый просмотр записей (Quick-Scan Records)

Определяет буферизацию доступа для файловой системы, которая использует многострочные буфера (прежде всего ASCII, BASIC и DOS). См. *Драйвера Баз данных* для

получения дополнительной информации. Эти файловые драйвера одновременно считывают сразу несколько записей. Однако при работе в многопользовательской среде они не обеспечивают 100 % надежности, поскольку в промежутке между сеансами считывания записи могут быть изменены другим пользователем. Как мера безопасности при этом применяется повторное наполнение буфера перед обращением к каждой записи.

Быстрый просмотр является довольно удачным способом считывания записей при пакетной обработке. Однако в некоторых многопользовательских системах небольшое улучшение целостности данных с помощью повторного считывания содержимого буфера достигается за счет существенного замедления обработки.

### **Фильтр записей (Record Filter)**

Здесь может быть напечатано условие фильтра, призванное ограничить список обрабатываемых записей только теми, которые ему удовлетворяют. Необходимо также задать приблизительное количество записей (см. *Приблизительное Количество Записей*). Этим выражением фильтруются все отображаемые записи. Когда Фильтр Записей используется совместно с Ограничением Диапазона, сначала проверяется соответствие записи этому диапазону. Поскольку ограничение диапазона использует при работе ключи, то проверка попадания в диапазон выполняется намного быстрее проверки условия фильтра.

**Совет:** Поля, используемые в выражении фильтра должны быть связаны (BIND). См. ниже раздел Горячие Поля.

### **Дополнительные поля сортировки (Additional Sort Fields)**

Здесь можно напечатать разделенный запятыми список полей, по которым следует произвести сортировку. Эти поля, в дополнение к ключам для обработки, определяются в окне диалога **Определение Схемы Файлов (File Schematic Definition)**.

### **Приблизительное количество записей (Approx Record Count)**

При выполнении обработки записей в порядке их физического расположения (без ключей), это число используется для вычисления процента выполнения операции, предоставляемого конечному пользователю в качестве обратной связи. Если же количество записей не задано, то перед началом обработки производится его вычисление, что может происходить как быстрее, так и медленнее в зависимости от файловой системы и размера файла. Если же Вы используете Фильтр Записей (или Ограничение

Диапазона, используемое совместно с фильтром), то приблизительное количество записей должно быть задано Вами.

### **Устанавливать границы панели прогресса вручную? (Set progress bar limits manually?)**

Снятие этого флажка приведет к тому, что процедура считывает результат и устанавливать границы панели прогресса автоматически. Автоматическое задание границ может привести к снижению эффективности для некоторых SQL наборов данных, а также к беспорядочному или неточному индицированию хода выполнения процесса для неравномерно распределенных результирующих наборов данных. Для ручного задания границ панели прогресса этот флажок надо взвести. Установка границ вручную может обеспечить более быструю работу для SQL драйверов и более точному индицированию хода выполнения процесса для неравномерно распределенных результирующих наборов данных. Однако это будет иметь эффект только в том случае, когда у Файла задан Ключ в окне диалога **Определение**

### **Схемы Файлов (File Schematic Definition).**

#### **Нижняя граница панели прогресса (Low Progress Bar Limit)**

Задайте здесь самое маленькое значение ключевого элемента для результирующего набора данных. Задавать можно как само значение, так и метку переменной, его содержащей. Строки со значениями литерального типа помещаются в одиночных кавычках ('значение').

#### **Верхняя граница панели прогресса (High Progress Bar Limit)**

Задайте здесь самое большое значение ключевого элемента для результирующего набора данных. Задавать можно как само значение, так и метку переменной, его содержащей. Строки со значениями литерального типа помещаются в одиночных кавычках ('значение').

### **Ограничение Диапазона (Range Limits)**

Эта панель становится доступной для выбора, только если для Файла задан Ключ в окне диалога **Определение Схемы Файлов (File Schematic Definition)**. Поскольку ограничение диапазона использует при работе ключи, то проверка попадания в диапазон обычно выполняется намного быстрее проверки условия фильтра.

#### **Поле Ограничения Диапазона (Range Limit Field)**

Совместно с **Типом Ограничения Диапазона (Range Limit Type)** определяет запись или группу записей, подвергающихся обработке. Ключевое поле, по значению которого следует ограничивать отбираемые записи, выбирается путем нажатия эллиптической кнопки.

#### **Тип Ограничения Диапазона (Range Limit Type)**

Определяет используемый тип ограничения диапазона. Выбирается из выпадающего списка, представленного ниже.

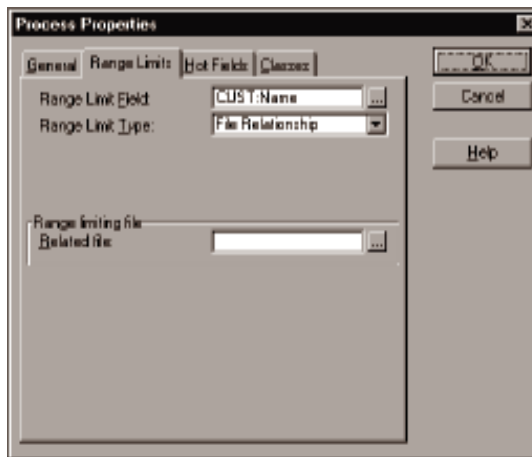
Текущее значение (Current Value)

Ограничивает ключевое поле его текущим значением.

**Единственное значение (Single Value)**

Позволяет ограничивать ключевое поле единственным значением.

Переменная, содержащая это значение, задается в поле **Ограничивающее диапазон значение (Range Limit Value)**.



**Ряд значений (Range of Values)**

Позволяет ограничивать ключевое поле некоторым рядом значений.

Переменные, содержащие верхнюю и нижнюю границы диапазона, задаются в полях **Нижняя граница диапазона (Low Limit Value)** и **Верхняя граница диапазона (High Limit Value)**.

**Связь между файлами (File Relationship)**

Позволяет ограничивать ключевое поле соответствующим текущим значением в связанном (родительском) файле. Для выбора файла, содержащего ограничивающие значения, нажмите эллиптическую кнопку **Связанный файл (Related file) (...)**. Это ограничивает процесс обработки включением только тех дочерних записей, которые соответствуют текущей записи родительского файла. Например, если Ваш отчет представляет собой список Заказов, то процесс обработки можно ограничить только теми заказами, которые относятся к текущему Клиенту.

## Горячие Поля

Закладка Горячие Поля позволяет выбирать дополнительные поля, добавляемые во VIEW-структуру. При просмотре файла сгенерированный исходный код считывает данные из VIEW-структуры, а не с диска, что позволяет оптимизировать скоростные характеристики. Элементы первичного и текущего ключа всегда включаются во VIEW, так что надобность в добавлении их к списку Горячих Полей отсутствует. Любое же другое поле, используемое при расчетах или в выражении фильтра, должно присутствовать во VIEW.

Кроме того, этот диалог можно использовать для связывания (BIND) полей. Любое поле, используемое в фильтре, должно быть связано.

## Классы

Закладка Классов позволяет управлять классом (и объектом), используемым в Вашей процедуре Процесса. Можно либо использовать предлагаемый по умолчанию Класс Разработки Приложений (ABC) и его объект (рекомендуется), либо выбрать другой ABC-класс из выпадающего списка, либо определить свой собственный класс. Использование Вашего собственного класса может дать возможность точно управлять работой процедуры, в то время как стандартный Класс разработки приложений (ABC) может не быть в точности тем, что нужно именно Вам.

См. *Обзор Шаблона — Пункты Диалога Закладки Классов — Локальный* для получения полной информации об этих опциях.

## Шаблон Отчета (Report Template)

Процедурный шаблон Отчета генерирует код, предназначенный для просмотра файла данных и приведения каждого элемента управления в разделе отчета ДЕТАЛЬНО (DETAIL) в соответствие каждой записи. Можно также задать фильгр записей или диапазон ограничения значений записей, подверженных обработке. Стандартное окно содержит индикатор хода процесса, который показывает конечному пользователю процент выполненной работы. Шаблон элемента управления PauseButton дает конечному пользователю возможность приостанавливать и возобновлять работу шаблона ReportTemplate.

Чтобы сформировать отчет, нажмите кнопку **Отчет (Report)**. Если только Вы не использовали при генерировании процедуры Мастера Отчета, то отчет не определен. См. *REPORT* в *Описании Языка* для получения дополнительной информации. Для определения итоговых значений используйте диалог **Свойства Строки (String Properties)**. См. *Создание Отчетов* и *Элементы управления и Их*

*Свойства в Руководстве Пользователя.*

**Совет:** Автоматический подсчет промежуточных итоговых значений уровней группы с помощью Процедурных шаблонов Отчета Clarion'a и элементов управления типа СТРОКА (STRING) не представляется возможным. Например, нельзя сложить вместе два итога по уровням группы с целью создания общего результата. Такой тип расчетов требует «ручного» отслеживания разрывов группы.

### Пункты диалога Шаблона Отчета (Report Template Prompts)

В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения* в *Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** Процедурного шаблона Отчета содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение** и **Поведение Окна** — смотри *Шаблоны Процедуры Окна* — *Пункты Диалога Шаблона Окна*), а также:

### Свойства Отчета (Report Properties)

Эта кнопка обеспечивает доступ к окну диалога с несколькими закладками, предоставляющими возможность определения широкого спектра функциональных возможностей для этой процедуры Отчета. В этом разделе описывается окно диалога **Свойства Отчета (Report Properties)**.

### Общая (General)

#### **Предварительный Просмотр Печати (Print Preview)**

Чтобы предоставить конечному пользователю возможность перед печатью просмотреть отчет на экране, взведите этот флажок. В этом случае конечный пользователь сможет напечатать отчет или отменить его. Установка этого флажка делает доступной закладку **Опции Предварительного Просмотра (Preview Options)**, позволяющую Вам управлять первоначальным отображением окна предварительного просмотра отчета.

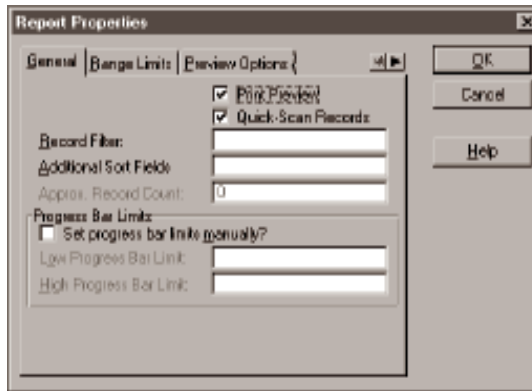
**Совет:** Класс ReportManager содержит свойство SkipPreview, которое управляет вызовом предварительного просмотра печати. Вы можете использовать SkipPreview для того, чтобы сделать доступным или нет предварительный просмотр печати на этапе выполнения: значение единица (1) делает предварительный просмотр печати доступным, значение же ноль (0) - нет.

### **Быстрый просмотр записей (Quick-Scan Records)**

Определяет буферизацию доступа для файловой системы, которая использует многострочные буфера (прежде всего ASCII, BASIC и DOS).



См. *Драйвера Баз данных* для получения дополнительной информации. Эти файловые драйвера одновременно считывают сразу несколько записей. Однако при работе в многопользовательской среде они не обеспечивают 100 % надежности, поскольку в промежутке между сеансами считывания записи могут быть изменены другим пользователем. Как мера безопасности при этом применяется повторное наполнение буфера перед обращением к каждой записи.



Быстрый просмотр является довольно удачным способом считывания записей при пакетной обработке. Однако в некоторых многопользовательских системах небольшое улучшение целостности данных с помощью повторного считывания содержимого буфера достигается за счет существенного замедления обработки.

### Фильтр записей (Record Filter)

Здесь может быть напечатано условие фильтра, призванное ограничить список обрабатываемых записей только теми, которые ему удовлетворяют. Необходимо также задать приблизительное количество записей (см. *Приблизительное Количество Записей*).

Этим выражением фильтруются все отображаемые записи. Когда Фильтр Записей используется совместно с Ограничением Диапазона, сначала проверяется соответствие записи этому диапазону. Поскольку ограничение диапазона использует при работе ключи, то проверка попадания в диапазон выполняется намного быстрее проверки условия фильтра.

**Совет:** Поля, используемые в выражении фильтра должны быть связаны (BIND). См. ниже раздел Горячие Поля.

### Дополнительные поля сортировки (Additional Sort Fields)

Здесь можно напечатать разделенный запятыми список полей, по которым следует произвести сортировку. Эти поля, в дополнение к ключам для обработки, определяются в окне диалога **Определение Схемы Файлов (File Schematic Definition)**.

### **Приблизительное количество записей (Approx Record Count)**

При выполнении обработки записей в порядке их физического расположения (без ключей), это число используется для вычисления процента выполнения операции, предоставляемого конечному пользователю в качестве обратной связи. Если же количество записей не задано, то перед началом обработки производится его вычисление, что может происходить как быстрее, так и медленнее в зависимости от файловой системы и размера файла. Если же Вы используете Фильтр Записей (или Ограничение Диапазона, используемое совместно с фильтром), то приблизительное количество записей должно быть задано Вами.

### **Устанавливать границы панели прогресса вручную? (Set progress bar limits manually?)**

Снятие этого флажка приведет к тому, что процедура считывает результат и устанавливает границы панели прогресса автоматически. Автоматическое задание границ может привести к снижению эффективности для некоторых SQL наборов данных, а также к беспорядочному или неточному индицированию хода выполнения процесса для неравномерно распределенных результирующих наборов данных. Для ручного задания границ панели прогресса этот флажок надо взвести. Установка границ вручную может обеспечить более быструю работу для SQL драйверов и более точному индицированию хода выполнения процесса для неравномерно распределенных результирующих наборов данных. Однако это будет иметь эффект только в том случае, когда у Файла задан Ключ в окне диалога **Определение Схемы Файлов (File Schematic Definition)**.

### **Нижняя граница панели прогресса (Low Progress Bar Limit)**

Задайте здесь самое маленькое значение ключевого элемента для результирующего набора данных. Задавать можно как само значение, так и метку переменной, его содержащей. Строки со значениями литерального типа помещаются в одиночных кавычках ('значение').

### **Верхняя граница панели прогресса (High Progress Bar Limit)**

Задайте здесь самое большое значение ключевого элемента для результирующего набора данных. Задавать можно как само значение, так и метку переменной, его содержащей. Строки со значениями литерального типа помещаются в одиночных кавычках ('значение').

## Ограничение Диапазона (Range Limits)

Эта панель становится доступной только если для Файла задан Ключ в окне диалога **Определение Схемы Файлов (File Schematic Definition)**. Поскольку ограничение диапазона использует при работе ключи, то проверка попадания в диапазон обычно выполняется намного быстрее проверки условия фильтра.

### **Поле Ограничения Диапазона (Range Limit Field)**

Совместно с **Типом Ограничения Диапазона (Range Limit Type)**

определяет запись или группу записей, подвергающихся обработке.

Ключевое поле, по значению которого следует ограничивать отбираемые записи, выбирается путем нажатия эллиптической кнопки.

### **Тип Ограничения Диапазона (Range Limit Type)**

Определяет используемый тип ограничения диапазона. Выбирается из выпадающего списка, представленного ниже.

Текущее значение (Current Value)

Ограничивает ключевое поле его текущим значением.

#### **Единственное значение (Single Value)**

Позволяет ограничивать ключевое поле единственным значением.

Переменная, содержащая это значение, задается в поле

**Ограничивающее диапазон значение (Range Limit Value)**.

#### **Ряд значений (Range of Values)**

Позволяет ограничивать ключевое поле некоторым рядом значений. Переменные, содержащие верхнюю и нижнюю границы диапазона, задаются в полях **Нижняя граница диапазона (Low Limit Value)** и **Верхняя граница диапазона (High Limit Value)**.

#### **Связь между файлами (File Relationship)**

Позволяет ограничивать ключевое поле соответствующим текущим значением в связанном (родительском) файле. Для выбора файла, содержащего ограничивающие значения, нажмите эллиптическую кнопку **Связанный файл (Related file) (...)**. Это ограничивает процесс обработки включением только тех дочерних записей, которые соответствуют текущей записи родительского файла. Например, если Ваш отчет представляет собой список Заказов, то процесс обработки можно ограничить только теми заказами, которые относятся к текущему Клиенту.

## Опции Предварительного просмотра (Preview Options)

Закладка Опции Предварительного Просмотра позволяет Вам управлять первоначальным отображением окна предварительного просмотра отчета. Эта закладка доступна для выбора только в том случае, если установлен флажок

**Предварительный Просмотр Печати (Print Preview)** на закладке *Общая*.

### **Задание масштаба первоначального отображения (Initial Zoom Setting)**

Устанавливает масштаб первоначального отображения отчета равным одному из четырех возможных вариантов. Начальные установки могут быть изменены конечным пользователем.

### **Может ли пользователь изменять масштаб? (Allow User Variable Zooms?)**

Чтобы позволить конечному пользователю задавать в дополнение к представленным вариантам другое значение масштаба установите этот флажок.

### **Установить начальное положение окна (Set Initial Window Position)**

Чтобы получить доступ к четырем нижеследующим пунктам диалога, позволяющим устанавливать начальное положение и размер окна предварительного просмотра, взведите этот флажок.

**Координата X (X Position)** Начальное положение левого края окна по горизонтали.

**Координата Y (Y Position)** Начальное положение верхнего края окна по вертикали.

**Ширина (Width)** Начальная ширина окна.

**Высота (Height)** Начальная высота окна.

**Максимизировать Окно Предварительного Просмотра (Maximize Preview Window)**

Чтобы максимизировать окно предварительного просмотра при первоначальном показе взведите этот флажок. При этом подменяются **Установки начального положения окна (Set Initial Window Position)**, заданные которыми координаты применяются только при восстановлении окна к его нормальному состоянию (состоянию с немасштабированными размерами).

## **Горячие Поля**

Закладка *Горячие Поля* позволяет выбрать дополнительные поля, добавляемые во VIEW-структуру. При просмотре файла сгенерированный исходный код считывает данные из VIEW-структуры, а не с диска, что позволяет оптимизировать скоростные характеристики. Элементы первичного и текущего ключа всегда включаются во VIEW, так что надобность в добавлении их к списку *Горячих Полей* отсутствует. Любое же другое поле, используемое при расчетах или в выражении фильтра, должно присутствовать во VIEW.

Кроме того, этот диалог можно использовать для связывания (BIND) полей. Любое поле, используемое в фильтре, должно быть связано.

### Фильтр (Filter)

Закладка Фильтров позволяет установить выражение, которое определяет, следует ли печатать текущую запись. На этапе выполнения, если значение этого выражения для текущей записи принимает значение Истина, процедура печатает запись. Причем условное выражение должно ссылаться как минимум на одно из полей, определенных во VIEW-структуре этой процедуры.

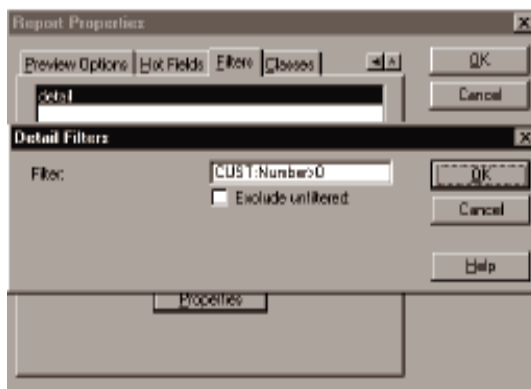
Этот фильтр печати представляет собой дополнение к **Фильтру Записей (Record Filter)** установленному на закладке Общая (General). **Фильтр Записей** определяет, какие записи считываются и подлежат обработке для построения отчета, в то время как **Детальный Фильтр (Detail Filters)** определяет только те из отфильтрованных записей, которые следует напечатать.

### **Свойства (Properties)**

Выберите структуру ДЕТАЛИ (DETAIL), подлежащую фильтрации, а затем нажмите кнопку Свойства, что даст возможность задать выражение фильтра. При этом откроется диалог **Детальный Фильтр (Detail Filters)**, который содержит следующие пункты диалога.

### Фильтр (Filter)

Здесь можно напечатать выражение, соответствующее стандартам языка Clarion. Если при работе программы значение выражения принимает значение Истина, процедура печатает структуру ДЕТАЛИ (DETAIL).



**Исключить нефильТРованное (Exclude unfiltered)**

Если взвести этот флажок, то заданный фильтр будет применен ко всем остальным структурам ДЕТАЛИ в отчете, которые не имеют фильтра печати. Это позволяет установить один и тот же фильтр для всех структур ДЕТАЛИ в Вашем отчете.

## **Классы**

Закладка Классов позволяет управлять классом (и объектом), используемым в Вашей процедуре Процесса. Можно либо использовать предлагаемый по умолчанию Класс Разработки Приложений (ABC) и его объект (рекомендуется), либо выбрать другой ABC-класс из выпадающего списка, либо определить свой собственный класс. Использование Вашего собственного класса может дать возможность точно управлять работой процедуры, в то время как стандартный Класс разработки приложений (ABC) может не быть в точности тем, что нужно именно Вам.

См. *Обзор Шаблона — Пункты Диалога Закладки Классов — Локальный* для получения полной информации об этих опциях.

## **Процедура-заставка (Splash Template)**

Шаблон Заставки генерирует код, обеспечивающий отображение окна с помещенным на него некоторым изображением и каким-либо текстом. По истечении указанного промежутка времени окно автоматически закрывается. Кроме того, можно разрешить пользователю закрывать окно в любой момент с помощью щелчка мышью по нему.

В стандартных Окнах Приложения предусмотрен необязательный вызов Процедуры-заставки. См. *Шаблон Окна Приложения* для получения дополнительной информации. С другой стороны, Процедура-заставка может быть вызвана с помощью исходного кода, помещенного в точку вставки. См. *Генератор Приложения — Вставленный Исходный Код*.

В соответствии с принятым соглашением, процедура-заставка сопровождает начало программы визуальными, звуковыми, или и теми, и другими эффектами. На экране заставки может быть размещен легко узнаваемый логотип или изображение, предназначенные как для поднятия уровня комфорта пользователя (вследствие узнавания привычного образа), так и в рекламных целях. Кроме того, это позволяет отвлечь внимание пользователя от подчас скучной процедуры загрузки и инициализации программы.

### **Пункты диалога Шаблона Процедуры-заставки (Splash Template Prompts)**

В дополнение к стандартным командным кнопкам Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** шаблона Процедуры-заставки содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение и Поведение Окна** — смотри *Шаблоны Процедуры Окна — Пункты Диалога Шаблона Окна*), а также:

#### **Отображать Время (в секундах) (Display Time)**

Определяет максимальное количество времени, в течение которого отображается окно заставки. По истечении указанного промежутка времени окно закрывается автоматически.

#### **Закрывать по щелчку пользователя (Close when the user clicks on the splash window)**

Взведение этого флажка позволяет пользователю закрывать окно в любое время с помощью щелчка мышью.

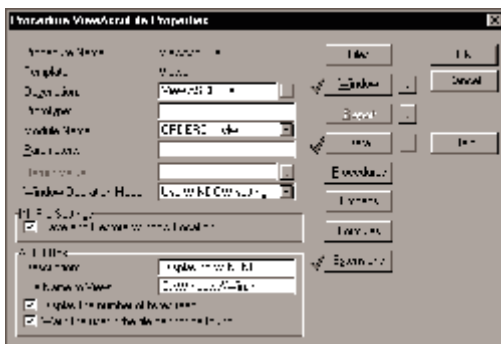
### **Шаблон Просмотрщика (Viewer Template)**

Шаблон Просмотрщика (Viewer) получен из Шаблона Окна. Он предлагает стандартное окно списка и такие кнопки, как ASCIIsearch, ASCIIprint и кнопка «Закреть» (Close).

Если Вы будете использовать шаблон для просмотра одного и того же ASCII-файла, можно использовать его в том виде, в каком он представлен. Если же ставиться целью просмотр любого ASCII-файла, выбранного с помощью стандартного окна диалога, то необходимо добавить поле ввода для принятия имени выбранного файла, плюс шаблон Элемента управления Поиск Файла DOS, предназначенный для выбора просматриваемого файла.

### **Пункты диалога Шаблона Просмотрщика (Viewer Template Prompts)**

В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), в окне диалога **Свойства Процедуры (Procedure Properties)** шаблона Просмотрщика содержатся пункты диалога, унаследованные от Шаблона Окна (**Параметры, Возвращаемое Значение и Поведение Окна** — смотри *Шаблоны Процедуры Окна — Пункты Диалога Шаблона Окна*), а также пункты диалога, предоставленные AsciiViewControl.



### Свойства AsciiViewControl (AsciiViewControl Properties)

Шаблон Элемента управления AsciiViewControl предлагает дополнительные пункты диалога, с помощью которых можно управлять тем, какой файл должен быть просмотрен, может ли конечный пользователь осуществлять поиск, печать файла, или и то, и другое. См. *Шаблоны Элемента управления — AsciiViewControl* для получения полного описания их пунктов диалога.

## **Другие Процедурные шаблоны**

### **Внешний Шаблон (External Template)**

Внешний Процедурный шаблон объявляет процедуру, содержащуюся во внешней библиотеке или в объектном файле (только \*.LIB). Генератор приложения не генерирует для этого шаблона никакого исходного кода, вместо этого проектная система просто привязывает названный внешний файл как отдельный модуль. См. *Стратегии Разработки и Развертывания* для получения дополнительной информации.

Для Внешнего Процедурного шаблона требуется связанный внешний Lib или Obj модуль (см. *Генератор Приложения — Меню Приложения в Руководстве Пользователя*). Если у Вашего приложения отсутствуют какие-либо внешние модули, Внешний Процедурный шаблон открывает окно диалога **Выбор Типа Модуля (Select Module Type)** позволяющее Вам создать их. Выберите в диалоге **Выбор Типа Модуля (Select Module Type)** Obj или Lib. Все остальные опции для Внешнего Процедурного шаблона непригодны.

В поле **Имя Модуля (Module Name)** необходимо выбрать из выпадающего списка имя файла внешней библиотеки или имя объектного файла. Причем в списке



появляются только те внешние модули, которые уже включены в проект. Поэтому если нужный Вам модуль отсутствует в списке, то сначала добавьте его в проект. Для добавления модуля в Генераторе Приложения выберите **Приложение** и **Вставить Модуль (Application** и **Insert Module)**. См. *Генератор Приложения – Меню Приложения в Руководстве Пользователя*.

Можно (но не обязательно) впечатать прототип внешней процедуры в поле **Прототип (Prototype)**. См. *Генератор Приложения – Прототипирование и Пропуск Параметра в Руководстве Пользователя*.

## **Шаблон Исходного Кода (Source Template)**

Процедурный шаблон Исходного кода предлагает изящный и простой способ добавления в приложение написанного вручную кода. Здесь предлагаются две точки вставки, в которые может быть включен написанный Вами текст: секция данных, и секция кода.

Шаблон просто объявляет процедуру, обрабатывает любые необязательные параметры, размещает внедренные объявления данных в секцию данных, начинает секцию CODE, а затем размещает любой внедренный выполняемый текст в секции CODE:

... (локальные данные)

CODE

... (вставленный Вами выполняемый текст)

### **Пункты диалога Шаблона Исходного Кода**

В дополнение к стандартным командным кнопкам и пунктам диалога Генератора Приложения (см. *Генератор Приложения в Руководстве Пользователя*), диалог **Свойства Процедуры (Procedure Properties)** шаблона Исходного Кода содержит следующие дополнительные пункты:

#### **Параметры (Parameters)**

Определяет для вашей процедуры список параметров. См. *PROCEDURE* и *Прототипы Процедуры в Описании Языка*, а также *Прототипирование и Пропуск Параметра в Руководстве Пользователя* для получения дополнительной информации.

Список параметров представляет собой необязательный список типов данных и меток, которые появляются в сгенерированном операторе PROCEDURE. Полный список размещается в круглых скобках. Для

каждого параметра, определенного в прототипе процедуры должно быть определено значение в списке параметров. Рекомендуется определять типы данных *и* метки параметров *как* в списке параметров, *так и* в прототипе процедуры. Например:

(SHORT Id,STRING Name)

Непосредственная же обработка указанных параметров должна быть реализована во вставленном исходном тексте процедуры.

## Диалоговые шаблоны

### Обзор

Почти все, что можно видеть в окне или документе, это элементы диалога (controls). Например, переключатель, кнопка, область ввода и список - все это диалоговые элементы.

Диалоговые шаблоны генерируют в исходных текстах программы объявления диалоговых элементов и код, управляющий связанными с ними данными. Например, диалоговый шаблон `BrowseBox` генерирует в исходной программе не только объявление спискового диалогового элемента, он также генерирует код для загрузки данных в `QUEUE` и последующего отображения `QUEUE` в списковом элементе, полностью реализуя функции прокрутки, поиска, сортировки, обновления и выбора с помощью мыши.

Диалоговые шаблоны могут также управлять файловым вводом-выводом: например, диалоговый шаблон `SaveButton` может выдать предупреждение, что были внесены изменения, если конечный пользователь попытается закрыть окно, не сохранив изменения на диске.

Совет: Обычно использование диалогового шаблона имеет преимущество перед просто диалоговым элементом.

Эта глава описывает все включенные в поставку Clarion диалоговые шаблоны и служит руководством по заполнению их параметров.

### Добавление диалоговых шаблонов

---

Для того чтобы добавить диалоговый шаблон к новой процедуре:

1. В Форматере Окна или Форматере Документа добавьте диалоговый шаблон, выбирая на панели диалоговых элементов инструмент .

2. Выберите диалоговый шаблон в диалоге **Select Control template (Выбор диалогового шаблона)**, затем поместите диалоговый элемент в нужном месте окна или документа, нажимая в этой точке кнопку мыши.

Форматер размещает в окне или документе один или несколько диалоговых элементов (тип элементов зависит от диалогового шаблона).

3. Затем щелкните правой кнопкой мыши по диалоговому элементу и выберите во всплывающем меню пункт **Actions**, чтобы получить доступ к параметрам диалогового шаблона.

Эти параметры определяют и модифицируют по Вашему заказу функциональность элемента.

4. Выберите в диалоге **Properties** другую закладку, чтобы установить внешний вид элемента, его позицию и другие функциональные особенности.

После добавления к процедуре диалогового шаблона в диалоге **Procedure Properties** около кнопки **Extensions** появляется галочка. С помощью этой кнопкой Вы также можете иметь доступ к настройкам шаблона.

## ***Только читающие шаблоны просмотра***

Только читающие шаблоны просмотра файлов включают AsciiViewControl шаблон и связанные с ним шаблон кнопки распечатки и шаблоны кнопок поиска контекста. В этом разделе описаны эти связанные шаблоны.

Распределенный шаблон AsciiViewInList обеспечивает те же самые функции для независимого спискового диалогового элемента LIST (размещенный нераспределенным шаблоном элемент LIST). См. дополнительную информацию в *Текстовые и распределенные шаблоны - AsciiViewInList*.

### **ASCIIViewControl**

---

Шаблон AsciiViewControl добавляет списковый диалоговый элемент LIST, в котором Вы можете показать содержимое файла в режиме только чтение, в том числе и содержимое файлов переменной длины. Это обычно используется для просмотра текстового ASCII файла. Шаблон AsciiViewControl дополнительно обеспечивает возможность поиска и распечатки выводимого файла.

Шаблон позволяет выбрать выводимый файл во время разработки, или, если нужно, оставить выбор конечному пользователю. И, наконец, шаблон дополнительно позволяет переключать вывод в списковом диалоговом элементе между указанным файлом и некоторыми другими данными.

Шаблон AsciiViewControl обеспечивает точки вставки для своего спискового элемента. Он также обеспечивает следующие настройки на закладке **Actions** диалога **List Properties** и в диалогах **Procedure Properties** и **Extension and Control Templates**:

### **Закладка General (Общие)**

#### **Initialize Viewer (Инициализация Viewer)**

Определяет, когда процедура инициализирует объект Viewer.

Инициализация включает выбор выводимого файла, его открытие и считывание.

On Open Window (При открытии окна)

Инициализирует Viewer, при открытии окна, что приводит к тому, что элемент LIST в момент появления на экране оказывается заполненным.

On Field Selection (При выборе элемента)

Задерживает инициализацию Viewer, пока конечный пользователь не выберет элемент LIST.

Manually (Вручную)

Viewer не инициализируется. Вы сами должны вставить вызов Viewer#.Initialize ROUTINE, чтобы проинициализировать Viewer.

#### **File to Browse (Просматриваемый файл)**

Определяет каталог и имя просматриваемого файла, или переменную, содержащую каталог и имя просматриваемого файла. Перед переменной нужно поставить восклицательный знак (!).

Если каталог не задается, процедура ищет файл в текущем каталоге. Если опущено (оставлено незаполненным), объект Viewer запрашивает выбор файла у конечного пользователя.

**Внимание!!!** Здесь текст руководства не соответствует шаблону.!!! (зам. Перев.)

#### **Reassign FROM attribute after Kill (Восстановить по завершении атрибут FROM)**

Отметьте этот режим, чтобы восстановить атрибут FROM элемента LIST по завершении работы Viewer. См. FROM в *Справочнике по языку*. Это позволяет использовать единственный списковый элемент для просмотра как **File to Browse**, так и других данных.

#### **Value or queue to assign (Присваиваемое значение или очередь)**

Введите метку QUEUE (или строковую константу) для присвоения атрибуту FROM Viewer-a.

#### **Allow popup menu searching (Разрешить поиск с помощью всплывающего меню)**

Отметьте этот режим, чтобы обеспечить поиск файла с помощью

выбора из всплывающего меню (нажатием правой кнопки мыши).

### **Allow popup menu printing**

Отметьте этот режим, чтобы обеспечить печать всех или некоторых записей файла с помощью выбора из всплывающего меню (нажатием правой кнопки мыши).

### **Закладка Classes**

Полное описание этих настроек см. в *Procedure Templates—Process Template*.

### **ASCIIPrintButton**

---

Шаблон AsciiPrintButton добавляет кнопку “Print” (печать) и обслуживающий ее код для печати некоторых или всех записей из связанного с шаблоном AsciiViewControl файла. Шаблон AsciiPrintButton доступен для использования только с существующим (ранее размещенным) шаблоном AsciiViewControl.

Шаблон AsciiPrintButton не имеет никаких дополнительных параметров. Он добавляет только точки вставки для BUTTON.

### **ASCIISearchButton**

---

Шаблон ASCIISearchButton добавляет кнопку “Search” (поиск) и обслуживающий ее код для поиска связанного с шаблоном AsciiViewControl файла. Шаблон ASCIISearchButton доступен для использования только с существующим (ранее размещенным) шаблоном AsciiViewControl.

Шаблон ASCIISearchButton не имеет никаких дополнительных параметров. Он добавляет только точки вставки для BUTTON.

### ***Шаблоны просмотра для чтения-записи.***

Шаблоны просмотра для чтения-записи в файл включают шаблон BrowseBox, шаблон RelationTree и взаимодействующие с ними шаблоны. В этом разделе описаны эти взаимосвязанные шаблоны.

### **Обзор BrowseBox**

---

Этот диалоговый шаблон размещает в окне “постранично загружаемый” списковый диалоговый элемент LIST и генерирует код для заполнения списка данными, перемещения, поиска, сортировки и выбора элементов данных. Он генерирует код для отбора или фильтрации данных, подсчета итогов, непосредственной корректировки данных или вызова отдельной процедуры для корректировки данных. Он также генерирует код для условного формирования

цветов и пиктограмм в каждой строке и колонке списка LIST.

Шаблон BrowseBox может сильно видоизменяться так же, как и диалоговый элемент LIST. То есть, любая функция, от прокрутки до раскраски, настраивается с помощью описанных ниже параметров шаблона BrowseBox. Поведение элемента LIST шаблона BrowseBox полностью настраивается с помощью Формatera Списка (см. *Форматер Списка в Руководстве Пользователя*).

Загрузка списка в BrowseBox происходит очень быстро, поскольку в память с диска загружается только несколько записей, а не целый файл. Главным преимуществом постраничной загрузки является скорость при просмотре больших файлов. Недостаток – то, что полосы прокрутки работают не совсем так гладко, как и для “файлово-загруженных” списков.

Совет: диалоговый шаблон BrowseBox можно использовать для управления постранично загружаемым выпадающим списком, просто задавая для атрибута DROP значение, которое больше нуля (0).

Совет: для размещения в окне файлово-загружаемого списка, применяйте диалоговый шаблон FileDrop и устанавливайте в нем значение атрибута DROP равное нулю (0).

### **Размещение в окне шаблона BrowseBox**

Вы можете разместить в окне диалоговый шаблон BrowseBox, выбирая шаблоны на панели диалоговых элементов () , затем выбирая в диалоге выбора шаблона **Select Control Template** элемент **BrowseBox – File Browsing List Box**. После того, как Вы выбрали шаблон BrowseBox, Генератор Приложений автоматически открывает Форматер Списка, чтобы Вы могли выбрать файлы, поля и переменные для отображения в списке и могли разработать представление списка и его полей.

### **Размещение и форматирование списковых полей**

Кнопка **Populate** позволяет Вам добавлять в список поле или переменную, по одному полю или переменной за раз. Диалог **Select Field** (выбор поля) предоставляет вам схему файлов. В пределах схемы шаблон BrowseBox появляется с надписью <To Do>. Чтобы добавить поле из определенного в словаре файла данных:

Нажмите кнопку **Insert**

Выберите файл в диалоге **Insert File**.

Если Вы хотите использовать ключ, нажмите кнопку **Key**, чтобы выбрать ключ в диалоге **Key Access**. Если Вы не указываете ключ, список отображается в порядке записей, что также отключает возможность устанавливать границы диапазона.

Выберите поле из списка **Fields**, который появляется в правой части диалога **Select Field**. После того, как Вы выбрали файл, ключ и поле (или переменную), появляется диалог **List Field Properties**. Он позволяет Вам точно определить представление полей в пределах списка.

Для дополнительной информации о файловой схеме см. *Генератор Приложений – Файлы Процедуры в Руководстве Пользователя*. Для дополнительной информации о форматировании списка LIST см. главу *Форматер Списка*.

Когда Вы закончите с Форматером Списка, укажите в окне место, где должен быть помещен списковый диалоговый элемент шаблона BrowseBox.

### **Свойства списка**

Чтобы вызвать диалог **List Properties**, нажмите правой кнопкой мыши на элементе LIST и выберите **Properties** во всплывающем меню. Для дополнительной информации о доступных в этом диалоге общих возможностях элемента LIST см. *Диалоговые элементы и их свойства -List в Руководстве Пользователя*

### **Прокрутка в BrowseBox**

Прокрутка постранично загружаемого, а возможно, и дополнительно профильтрованного файла базы данных весьма отличается от прокрутки более типичных Windows файлов, таких, как, например, документы текстовых процессоров, электронные таблицы или списки каталогов как в Менеджере Файлов. Главное различие - это большой размер базы данных (требующий постраничной загрузки) и возможное изменение размера из-за фильтрации. Эти различия могут привести к отличному от ожидаемого поведению зоны вертикальной прокрутки.

### **Соображения о размерах**

Даже при полной и точной калибровке, вертикальный движок полосы прокрутки может обеспечить лишь весьма далекие от оптимальных результаты для больших наборов данных. Например, для результирующего набора в сто тысяч записей, идеально откалиброванный движок зоны прокрутки может обеспечить только очень грубое позиционирование. Дело в том, что минимальное расстояние, на которое можно переместить движок, (пиксель) составляет от 1/200 (типичная зона прокрутки при разрешении 640x480) до 1/400 (длинная зона прокрутки при разрешении 640x480) от общей длины зоны прокрутки, и, следовательно, от 1/200 (500 записей) до 1/400 (250 записей) в результирующем наборе. Для результирующего набора в миллион записей, числа подсказывают до 2,500 - 5,000 записей на пиксель, приводя к несостоятельности вертикальной прокрутки в качестве инструмента навигации.



Для больших наборов данных локаторы в сочетании с VCR кнопками (без скользящего движка) обеспечивают лучшее функционирование и более довольных конечных пользователей.

### **Соображения о калибровке**

Вместе с преимуществом в скорости выполнения, которое приходит с постранично загружаемыми списками, приходит и недостаток отсутствия в памяти всего набора, и, следовательно, незнание общего количества записей в результирующем наборе и относительной позиции текущей записи в пределах этого набора. Для того, чтобы обеспечивать “стандартное” поведение зоны вертикальной прокрутки для постранично загружаемого списка, процедура прокрутки должна знать три вещи: количество записей в результирующем наборе, относительную позицию записи в результирующем наборе, и запись, которая находится на относительной позиции в результирующем наборе. Поскольку процессоры баз данных не предоставляют такую информацию для результирующих наборов, построенных по ключу или с применением фильтров (это было бы слишком медленным), процедура для постранично загружаемой прокрутки должна оценивать эти величины, когда конечный пользователь тащит движок зоны прокрутки или выбирает или находит локатором запись в результирующем наборе.

Шаблон BrowseBox предлагает разнообразный комплект вариантов, оценки этих величин и калибровки вертикальной зоны прокрутки для наилучшего соответствия Вашей базе данных. Для дополнительной информации см. *Поведение зоны прокрутки*.

### **Возможности BrowseBox**

BrowseBox Шаблон имеет следующие параметры (закладка **Actions** в диалоге **List Properties**, диалоги **Browse Procedure Properties** и **Extension and Control Templates**), а также точки вставки для LIST:

#### **Default Behavior (стандартное поведение)**

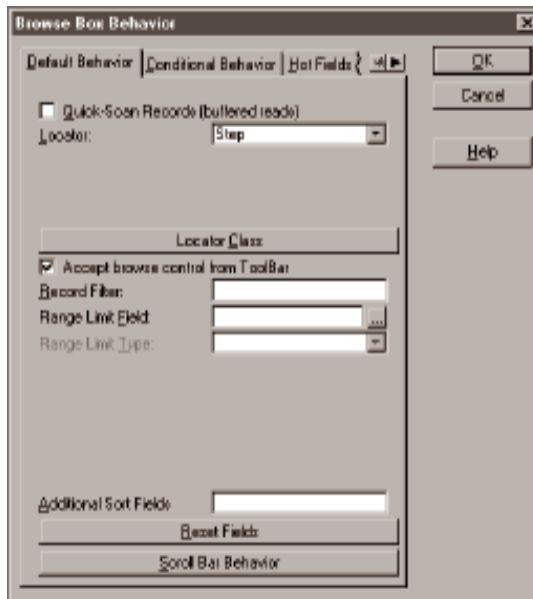
Эта закладка содержит параметры, которые управляют стандартным поведением BrowseBox.

#### **Quick-Scan Records (быстрое считывание записей)**

Определяет буферизованный доступ для тех файловых систем, где используются буферы на несколько записей (прежде всего ASCII, BASIC и DOS). Дополнительную информацию см. в *Драйверы баз данных*. Эти файловые драйверы читают буфер за один раз, обеспечивая быстрый доступ.

В многопользовательской среде эти буферы - не 100% надежны, поскольку другой пользователь может изменить запись между доступами. Для безопасности драйвер повторно считывает буферы перед каждым обращением к записи.

Быстрое считывание является нормальным способом чтения записей для просмотра. В то же время, повторное считывание буфера может обеспечить слегка улучшенную целостность данных в некоторых многопользовательских обстоятельствах ценой значительного замедления обработки.



## Locator (Локатор)

Локатор позволяет пользователю искать в списке указанные записи без ручной прокрутки всего списка. **Locator** доступен только при просмотре файл в ключевом порядке (задайте KEY на файловой схеме). Поле поиска должно быть первым свободным элементом ключа, то есть первым элементом среди компонент ключа, диапазон которого не ограничен единственным значением.

Для просмотров по нескольким ключам (их создают Мастера (Wizards)), можно иметь несколько локаторов. Используйте закладку **Conditional Behavior** (условное поведение), чтобы задать дополнительные локаторы для дополнительных сортировок. Выберите в выпадающем списке один из следующих типов локатора:



- None (нет)** Определяет отсутствие локатора.
- Step (шаг)** Определяет односимвольный локатор, не требующий диалогового элемента локатора. Когда BrowseBox имеет фокус и пользователь набирает символ, список позиционируется на первое появление поля ключа, начинающегося с этого символа (или следующий больший символ, если никакой ключ не соответствует символу локатора). Повторный ввод того же символа позиционирует список на следующую запись с ключом, начинающимся с этого символа.

Используйте step-локатор, когда первый свободный ключевой элемент - это STRING, CSTRING или PSTRING, и Вы хотите, чтобы поиск выполнялся немедленно по нажатию на клавишу пользователем. Step-локаторы не годятся для числовых ключей. Если нет ключа для просмотра, Генератор Приложений преобразовывает к отсутствию локатора.

- Entry (ввод)** Определяет многосимвольный локатор, который активизируется при завершении диалогового элемента локатора (не на каждом нажатии клавиши). Диалоговый элемент локатора может быть типа ENTRY, COMBO или SPIN. Используйте entry-локатор, когда нужен поиск в числовых или текстовых ключах, и Вы хотите задержать поиск, пока пользователь не закончит ввод в диалоговый элемент локатора (нажмет клавишу Enter или Tab). Этот отложенный поиск уменьшает сетевой трафик и обеспечивает более плавный поиск в клиент-серверной среде.

Диалоговый элемент локатора должен находиться после элемента LIST в диалоге **Set Control Order (установить порядок элементов)**.

По умолчанию, диалоговый элемент локатора - это тот диалоговый элемент, у которого атрибут USE является первым свободным элементом ключа просмотра. Свободный элемент - это тот элемент, диапазон которого *не* ограничен единственным значением. Если нет такого диалогового элемента, Генератор Приложений преобразовывает в step-локатор. Если нет ключа для просмотра, Генератор Приложений преобразовывает к отсутствию локатора.

Когда конечный пользователь помещает один или более символов в диалоговый элемент локатора, а затем *завершает* ввод, нажимая Tab, кнопку локатора - (см. *FrameBrowseControl* , или *Свойства List*  (*списка*)), или, выбирая другой диалоговый элемент экрана, список позиционируется на ближайшую подходящую запись.

## Incremental (наращиваемый)

Определяет многосимвольный локатор, которому не требуется (но настоятельно рекомендуется) диалоговый элемент локатора. Используйте наращиваемый локатор, когда нужен поиск по числовым или текстовым ключам, и Вы хотите, чтобы поиск происходил немедленно по нажатию клавиши пользователем.

Диалоговый элемент локатора должен находиться после элемента LIST в диалоге **Set Control Order**.

Диалоговый элемент локатора может быть типа STRING, ENTRY, COMBO, или SPIN, однако любой диалоговый элемент, кроме STRING, заставляет наращиваемый локатор вести себя подобно entry-локатору - поиск откладывается до завершения ввода в диалоговый элемент.

С диалоговым элементом типа STRING, когда список имеет фокус, при каждом нажатии на клавишу символы автоматически заносятся в строку локатора, и список немедленно позиционируется на ближайшую подходящую запись. Клавиша возврата удаляет символы из строки локатора.

Мы настоятельно рекомендуем использовать диалоговый элемент типа STRING в качестве диалогового элемента наращиваемого локатора, потому что поиск происходит *немедленно* с каждым нажатием клавиши, и пользователь может *видеть* то значение ключа, которое ищется в BrowseBox.

По умолчанию, диалоговый элемент локатора - это тот диалоговый элемент, у которого атрибут USE является первым свободным элементом ключа просмотра. Свободный элемент - это тот элемент, диапазон которого *не* ограничен единственным значением. Если нет такого диалогового элемента, Генератор Приложений преобразовывает в step-локатор. Если нет ключа для просмотра, Генератор Приложений преобразовывает к отсутствию локатора.

## Filter (фильтр)

Определяет многосимвольный локатор, которому не требуется (но настоятельно рекомендуется) диалоговый элемент локатора. Используйте фильтр-локатор, когда нужен поиск по текстовым ключам, и Вы хотите *минимизировать сетевой трафик*.

Этот локатор похож на наращиваемый локатор с фильтром записей. Он определяет *область* значений, в которой ведется поиск и возвращает ограниченный результирующий набор - только те записи, которые находятся в пределах указанной области. Каждый последующий заданный

(дополнительный) поисковый символ приводит к меньшему, более точно отобранному результирующему набору. Например, поисковое значение 'A' возвращает все записи от 'AA' до 'AZ'; поисковое значение 'AB' возвращает все записи от 'ABA' до 'ABZ'.

Фильтр-локатор определяет границы для поиска, основываясь на заданном пользователем поисковом значении. Реализация ограничений зависит от драйвера базы данных - для SQL баз данных, фильтр-локатор использует LIKE; для ISAM баз данных (с индексно-последовательным методом доступа) он задает верхнюю и нижнюю границы.

Локатор возвращает *только* те записи, которые соответствуют поисковому значению, обеспечивая, по существу, в просмотре фильтр с динамическими границами.

Совет: Фильтр-локатор очень хорошо работает в SQL базах данных и на младших компонентах ключей; однако производительность может падать при использовании с неключевыми полями или на старших компонентах ключа в не-SQL базах данных.

### **Override default locator control (переопределить стандартный диалоговый элемент локатора)**

Стандартным диалоговым элементом локатора является такой диалоговый элемент, у которого атрибут USE является первым свободным элементом в ключе просмотра. Включите этот режим для того, чтобы переопределить этот стандарт и задать другой диалоговый элемент в качестве локатора. Это нужно на тот случай, когда у вас есть несколько диалоговых элементов с одним и тем же свободным ключевым элементом, в качестве USE-атрибута - например, когда у Вас есть как возрастающий, так и убывающий ключ для одного и того же поля.

Выберите один из диалоговых элементов для использования в качестве диалогового элемента локатора из списка **New Locator Control**.

### **Locator Class (класс Locator)**

Нажмите эту кнопку, чтобы переопределить глобальную настройки Locator Manager. См. *Обзор шаблонов—Classes Tab Options—Global и Local*.

### **Accept browse control from Toolbar (использовать органы управления инструментальной панели)**

Включите этот переключатель, чтобы обрабатывать навигационные и другие управляющие просмотром события, сгенерированные диалоговым

шаблоном `FrameBrowseControl` на инструментальной панели приложения (APPLICATION). Дополнительную информацию о кнопках на инструментальной панели и их функционировании см. в *FrameBrowseControl*. Выключите этот переключатель, чтобы выключить для этой процедуры кнопки на инструментальной панели и пользоваться только локальными средствами навигации. См. также *SetToolbarTarget*.

#### Record Filter (фильтр записей)

Наберите правильное выражение Clarion, чтобы ограничить содержимое списка просмотра только теми записями, для которых выражение оценивается как истина (не нуль или не пробел). Процедура последовательно проходит по всем воспроизводимым записям, чтобы выбрать только те, которые удовлетворяют фильтру. Фильтры обычно работают значительно медленнее, чем ограничения диапазона.

Вы должны связать (BIND) все поля файла, которые используются в выражении фильтра. Связывание (BIND) полей выполняется на закладке **Hot Fields**.

#### Range Limit Field (поле ограниченного диапазона)

В сочетании с **Range Limit Type** определяет запись или группу записей для включения в список. Выберите поле, нажимая кнопку с многоточием (...). Ограничения на диапазон зависят от ключа. Ограничения по диапазону – обычно работают значительно быстрее, чем фильтры.

#### Range Limit Type (тип ограничений диапазона)

Определяет применяемый тип ограничений по диапазону. Выберите один из вариантов из выпадающего списка.

#### Current Value (текущее значение)

Ограничивает ключ текущим значением **Range Limit Field**.

#### Single Value (единственное значение)

Позволяет ограничивать ключ заданным значением. Определите переменную, содержащую это значение в появившемся поле **Range Limit Value**.

#### Range of Values (диапазон значений)

Позволяет определять верхний и нижний предел. Укажите переменные, содержащие граничные значения в полях **Low Limit (нижний предел)** и **High Limit (верхний предел)**.

#### File Relationship (связь с файлом)

Позволяет Вам выбирать ограничивающий диапазон файл в отношении

1:MANY. Это ограничивает список только теми дочерними записями, которые соответствуют текущей записи в родительском файле. Например, если ваш список был списком Заказов, Вы могли бы ограничить вывод только заказами текущего Клиента (в файле Клиента).

#### Additional Sort Fields (дополнительные поля сортировки)

Укажите дополнительные ко всем полям, входящим в указанный в **File Schematic** ключ, поля сортировки (разделенный запятыми список имен полей). Для дополнительной информации см. ORDER в *Справочнике по языку*.

#### Reset Fields (сторожевые поля)

Нажмите эту кнопку, чтобы добавить сторожевые поля. Если значение любого сторожевого поля изменяется, процедура перестраивает список BrowseBox (повторно применяет порядок сортировки, фильтр, и т.п.). Нет необходимости объявлять поля границ диапазонов или локаторы в качестве сторожевых полей. Применяйте сторожевые поля для перестроения списка BrowseBox на основе изменения диалоговых элементов или данных, непосредственно не связанных с BrowseBox.

#### Scroll Bar Behavior (поведение прокрутки)

Нажатие этой кнопки отображает диалог, где Вы можете определить поведение вертикальной зоны прокрутки BrowseBox.

Выберите с фиксированный (**Fixed Thumb**) или подвижный (**Movable Thumb**) движок.

Совет: Для списков, в которые загружается файл целиком, автоматически получается стандартное в Windows (movable thumb) поведение зоны прокрутки. Однако, поскольку это невозможно для постранично загружаемых списков, эти параметры позволяют Вам выбрать такое поведение, которое наилучшим образом подходит Вашему приложению.

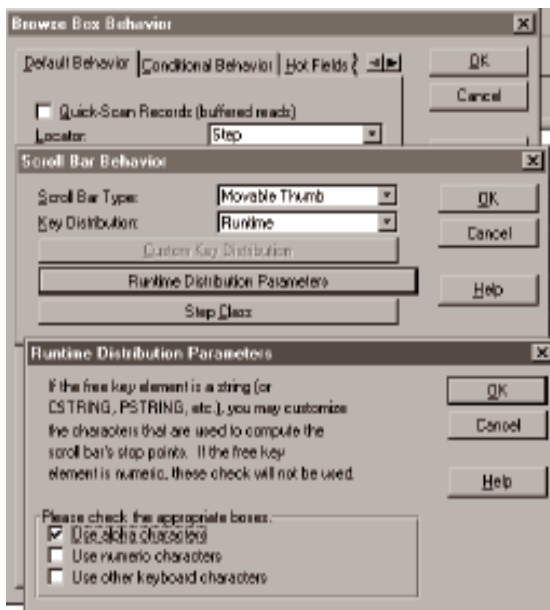
#### Fixed Thumb (фиксированный движок)

Движок (объемный квадратный блок в середине зоны прокрутки) остается в центре зоны прокрутки. Щелкните мышкой выше движка, чтобы прокрутить список на одну “страницу” вверх. Щелкните ниже движка, чтобы прокрутить список на одну “страницу” вниз. Перетащите движок в начало или конец зоны прокрутки для того, чтобы переместиться на начало или конец файла.

Совет: Используйте фиксированный указатель при просмотре больших SQL таблиц для улучшения производительности.

## Movable Thumb (подвижный движок)

Щелкните и перетащите движок для того, чтобы прокрутить список на пропорциональное расстояние. Движок остается там, куда Вы его перетащили, и его положение в зоне прокрутки показывает относительную позицию в пределах списка просмотра. Щелкните мышкой выше движка, чтобы прокрутить список на одну “страницу” вверх. Щелкните ниже движка, чтобы прокрутить список на одну “страницу” вниз. При выборе **Movable Thumb** можно также задать распределение ключа (**Key Distribution**), чтобы уточнить, как именно BrowseBox должен оценивать относительную позицию движка в пределах списка просмотра.



## Key Distribution (распределение ключа)

Определяет распределение точек в зоне прокрутки. Выберите из выпадающего списка одно из двух встроенных распределений (алфавитное или по фамилиям), настраиваемое или истинное.

### Alpha (по алфавиту)

Определяет 100 равномерно распределенных в алфавитном порядке точек.

### Last Names (по фамилиям)

Определяет 100 точек, распределенных как обычные фамилии в Соединенных Штатах. Если ключ доступа числовой, то нужно применять



настраиваемое или истинное распределение.

### **Custom (настраиваемое)**

Позволяет Вам определить свои собственные точки

### **Run-time (истинное)**

Считывает первую и последнюю запись и рассчитывает значения 100 равномерно распределенных между ними точек.

### **Custom Key Distribution (настраиваемое распределение ключа)**

Позволяет определять опорные точки распределения в зоне прокрутки (это полезно, когда у вас используются данные с несимметричным распределением). Внесите в список значения для каждой опорной точки. Строковые константы должны быть заключены в одиночные кавычки ( ‘ ‘ ).

### **Run-time Distribution Parameters (параметры истинного распределения)**

Позволяет Вам задать характер учитываемых при определении дистрибутивных точек символов. Это необходимо только, когда свободный элемент ключа имеет тип STRING или CSTRING. Отметьте те типы символов, которые быть включены в рассмотрение. Выберите из **Use alpha characters (Aa-Zz) (использовать буквы)**, **Use numeric characters (0-9) (использовать цифры)**, и **Use other keyboard characters (использовать другие символы клавиатуры)**.

### **Step Class (класс Step)**

Чтобы переопределить глобальные настройки Step Manager, нажмите эту кнопку. См. *Обзор шаблонов – Закладка Classes Options - Глобальный и Локальный*.

## **Conditional Behavior (условное поведение)**

Эта закладка содержит список, который позволяет Вам определять зависимость поведения BrowseBox от различных условий или выражений. Добавляйте выражения к списку, нажимая кнопку Insert. Она вызывает диалог, где Вы определяете выражение и сопутствующее ему поведение в тех случаях, когда это выражение оценивается как истина (не ноль или не пробел).

В период выполнения эти выражения оцениваются и используется поведение для первого в списке истинного условия.

В этом диалоге Вы можете определить:

<b>Condition</b>	Любое правильное Clarion выражение.
<b>Key to Use</b>	Необязательно используемый ключ сортировки данных в BrowseBox, при истинном выражении.

Остальные поля и кнопки такие же, как и на закладке Default Behavior (стандартное поведение)

### Hot Fields (“горячие” поля)

Когда Вы выбираете закладку Hot Fields, Вы можете указать, какие из полей, не попавших в список, нужно добавить к QUEUE. При перемещениях по файлу сгенерированная исходная программа считывает данные для этих полей из QUEUE, а не с диска. Это ускоряет обновление списка.

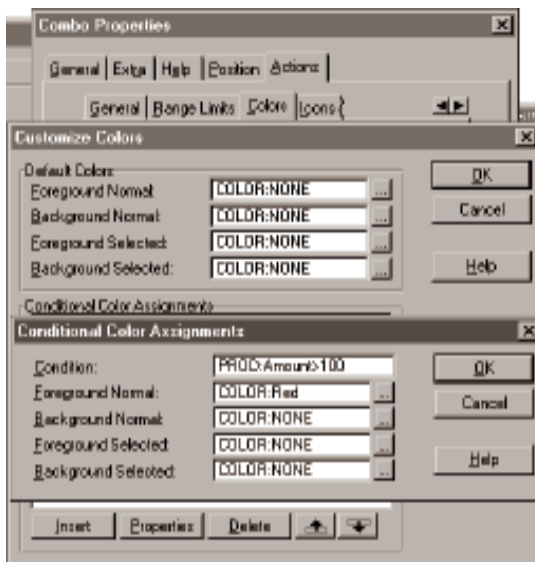


Указание “горячих” полей также позволяет Вам размещать за пределами BrowseVox диалоговые элементы, которые будут обновляться всякий раз при выборе в списке новой записи. Поля первичного ключа и текущего ключа всегда включаются в QUEUE, поэтому их не следует включать в список “горячих” полей.

Этот диалог также позволяет Вам связывать (BIND) поля. Вы должны связать любое поле, которое используется в выражении фильтра или для подсчета итогов.

### Colors (палитра)

Эта закладка доступна, только если Вы отметили режим **Color Cells** (цветные ячейки) в Форматере Списка. На ней показан список столбцов BrowseVox, которые можно окрашивать.



Для того, чтобы определять стандартные и любые условные цвета, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это откроет диалог **Customize Colors (настройка цветов)**.

### Customize Colors (настройка цветов)

Этот диалог позволяет определять стандартные и условные цвета окрашивания фона и текста для обычных (невывбранных) и для выбранных колонок.

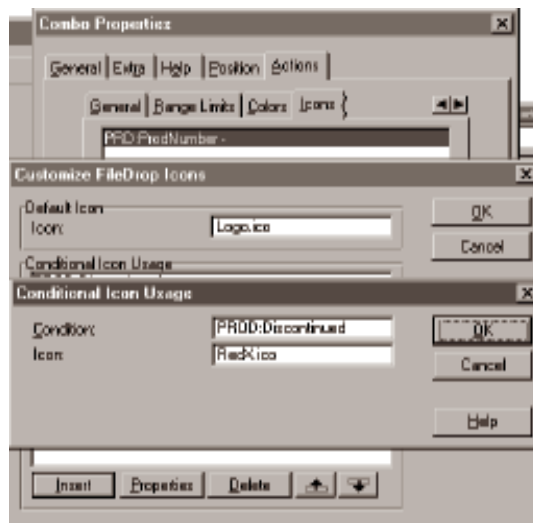
#### Conditional Color Assignments (условные назначения цветов)

Под разделом встроенных цветов находится список **Conditional Color Assignments** (условные назначения цветов). Этот список позволяет Вам определить применяемые цвета, в тех случаях, когда выражение оценивается как истина (не ноль или не пробел). Нажмите кнопку Insert, для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета.

В период исполнения выражения оцениваются и используются цвета для первого истинного выражения.

### Icons (пиктограммы)

Эта закладка доступна, только если Вы отметили режим **Icons** в Форматере Списка. На ней показан список столбцов BrowseBox, в которых можно вывести пиктограммы.



Для того, чтобы определять стандартные и любые условные пиктограммы, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это

откроет диалог **Customize BrowseBox Icons (настройка пиктограмм BrowseBox)**.

### **Customize BrowseBox Icons (настройка пиктограмм BrowseBox)**

Этот диалог позволяет определять стандартные и условные пиктограммы для столбца BrowseBox.

#### **Default Icon (стандартная пиктограмма)**

Стандартно выводимая пиктограмма. Напечатайте имя файла пиктограммы (.ICO).

#### **Conditional Icon Usage (использование условных пиктограмм)**

Под разделом **Default Icon (стандартная пиктограмма)** находится список **Conditional Icon Usage (использование условных пиктограмм)**. Этот список позволяет Вам определить применяемые пиктограммы в тех случаях, когда выражение оценивается как истина (не нуль или не пробел). Нажмите кнопку Insert для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета. В период исполнения выражения оцениваются и используется пиктограмма для первого истинного выражения.

### **Totaling (подсчет итогов)**

Эта закладка содержит список, который позволяет Вам определять итоговые поля для BrowseBox. Чтобы добавить итоговое поле нажмите кнопку Insert. Она откроет диалог **Browse Totaling (подсчет итогов в просмотре)**, где Вы можете определить итоговые поля для BrowseBox.

#### **Total Target Field (целевое итоговое поле)**

Переменная, чтобы сохранить рассчитанный итог. Это может быть локальным, модуль, или глобальная переменная. Вы можете также использовать файловую область; тем не менее, Вы должны написать код, чтобы скорректировать файл.

#### **Total Type (вид итога)**

Выберите из выпадающего списка **Count (количество)**, **Sum (сумма)** или **Average (среднее)**. **Count** подсчитывает количество записей. **Sum** складывает значения в поле Field to Total (вид итога). **Average** вычисляет среднее арифметическое по полю Field to Total (вид итога).

#### **Field to Total (обрабатываемое поле)**

Суммируемое или усредняемое поле. Этот поле выключается, когда видом итога является **Count (количество)**.

#### **Total Based On (итог по)**

Выберите из выпадающего списка **Each Record Read (каждая считанная запись)** или **Specified Condition (заданное условие)**. Это определяет использование каждой записи или только тех, которые отвечают условию Total Condition (условия подсчета).

### **Total Condition (условия подсчета)**

Условие, которое должно выполняться при использовании итога, базирующегося на заданном условии. Можно использовать любое правильное Clarion выражение. Вы должны связать (BIND) любые использованные в этом выражении имена полей. Используйте закладку Hot Fields (горячие поля), чтобы связать (BIND) названия полей.

### **Classes (классы)**

Закладка Classes (классы) позволяет Вам воздействовать на используемые в процедуре классы (и объекты). Вы можете воспользоваться стандартными ABC классами и объектами (рекомендуется), или можно определить свои собственные классы или классы третьих лиц. Вывод вашего собственного класса может дать Вам очень тонкое управление процедурой в тех случаях, когда стандартные ABC классы не вполне точно отвечают Вашим потребностям.

См. *Обзор шаблонов – Закладка Classes Options - Local* для полной информации об имеющихся возможностях.

### **BrowsePublishButton**

Шаблон BrowsePublishButton добавляет кнопку “Publish” для генерации Hypertext Markup Language (HTML) страницы для отображения записей из очереди BrowseBox. Другими словами, используйте этот шаблон, чтобы опубликовать информацию из Вашего BrowseBox в Internet на Web странице!

Результирующая WEB страница содержит заданный Вами заголовок и заголовки столбцов из Вашего списка. WEB страница форматирует данные списка, используя указанные в списке шаблоны форматирования.

В период исполнения пользователь может указать имя файла для записи сгенерированного HTML. Пользователь имеет возможность выбрать публикацию всех записей из очереди BrowseBox или только тех, которые в текущий момент отображаются на экране.

Совет: Шаблон BrowsePublishButton доступен только в цепочке шаблонов Clarion и не может использоваться с ABC шаблонами.

Шаблон BrowsePublishButton имеет следующие параметры:

#### **Use variable for HTML name (использовать переменную для имени HTML)**

Включите этот режим, чтобы определять HTML файл в переменной. Это включает поле **Variable HTML filename** (переменное имя HTML

файла) для назначения переменной для имени HTML файла и выключает поле **Default HTML Name** (стандартное имя HTML файла).

#### Default HTML Name

Определяет стандартное имя файла для HTML кода или переменную, которая содержит имя HTML файла. Нажмите кнопку с многоточием (...), чтобы выбрать файл с помощью стандартного диалога **Open File** (открыть файл), или выбирать или определить поле в словаре данных или переменную памяти с помощью диалога **Select Field** (выбор поля).

Если не будет указан полный путь к файлу, процедура запишет файл в текущий каталог. В период исполнения пользователь может определить другой файл и каталог.

#### HTML Title (Заголовок HTML)

Определяет название HTML документа. Название появляется при отображении документа в заголовке окна Web-просмотрщика.

#### Table Heading (заголовок таблицы)

Определяет заголовок, который отображается сверху Web-страницы.

#### Background Graphic (Фоновая графика)

Определяет графическое изображение, которое рисуется “за” записями из очереди. Обратите внимание, что большинство Web-просмотрщиков поддерживают фоновое графическое изображение, однако некоторые старые версии этого не делают.

#### Use Grid Lines (использовать линии сетки)

Отметьте этот режим, чтобы показать элементы очереди в прямоугольной сетке. Обратите внимание, что большинство Web-просмотрщиков поддерживают линии сетки, однако, некоторые старые версии этого не делают.

#### Grid Line Width (толщина линий сетки)

Определяет толщину рамки, окружающей прямоугольную сетку.

### **BrowseSelectButton**

---

Шаблон `BrowseSelectButton` размещает кнопку “Select” для выбора записи из списка.

Сгенерированный исходный код извлекает из списка текущую выбранную запись (делает выбранную запись текущим значением буферов записи файлов) и закрывает процедуру. Для конечного пользователя, нажатие кнопки “Select” эквивалентно двойному нажатию на строку в списке.

Шаблон `BrowseSelectButton` имеет следующие параметры:

Hide the Select button when not applicable (убрать неприменимую кнопку)

Отметьте этот режим, чтобы спрятать кнопку “Select”, когда процедура вызывается не для целей выбора (`GlobalRequest <> SelectRecord`).

Allow Select via PopUp (разрешить выбор через всплывающее меню)

Отметьте этот режим, чтобы разрешить выбор записи с помощью всплывающего при нажатии на правую кнопку меню. Шаблон добавляет во всплывающее меню пункт, текст которого соответствует тексту на кнопке “Select”. Пункт меню выключается, когда выключается или прячется кнопка “Select”.

## **BrowseUpdateButtons**

---

Шаблон `BrowseUpdateButtons` размещает три кнопки для управления файловым вводом/выводом в `BrowseBox`: “Insert” (добавить), “Change” (изменить), и “Delete” (удалить). Эти три кнопки воздействуют на записи в `BrowseBox`. При нажатии кнопка извлекает выбранную запись и запускает соответствующее действие для этой записи базы данных.

Шаблон `BrowseUpdateButtons` позволяет Вам использовать отдельную процедуру редактирования (рекомендуется для файлов с двойными связями) или редактирование-на-месте (рекомендуется для файлов справочников - файлов с односторонними связями).

Шаблон `BrowseUpdateButtons` обеспечивает следующие параметры:

### **Update Procedure (процедура редактирования)**

Введите имя процедуры или выберите его из выпадающего списка. Если Вы вводите новое имя процедуры, Генератор Приложений добавляет новую процедуру в Дерево Приложения.

Use Edit in place (использовать редактирование-на-месте)

Отметьте этот режим, чтобы позволить конечному пользователю корректировать просматриваемый файл, печатая непосредственно в списке `BrowseBox`. Это обеспечивает весьма непосредственный, интуитивный стиль редактирования типа электронных таблиц. Поведение редактирования-на-месте можно сконфигурировать с помощью кнопки **Configure Edit in place**.

### Configure Edit in place (конфигурировать редактирование-на-месте)

Нажмите эту кнопку, чтобы открыть диалог **Configure Edit in place**. Этот диалог обеспечивает следующие параметры:

### Save (сохранение)

Диалог **Configure Edit in place** предлагает настроить режим **Save** (сохранение) для четырех разных клавиатурных действий. Эти настройки определяют, будут ли изменения в редактируемой записи сохранены или нет при следующих клавиатурных действиях: клавиша Tab в конце строки, клавиша Enter, стрелки вверх или вниз, потеря фокуса (перенос фокуса на другой диалоговый элемент окна, обычно с помощью нажатием мышью). Выберите из:

#### *Default (стандартно)*

Сохранять запись, как определено в методе BrowseClass.Ask.

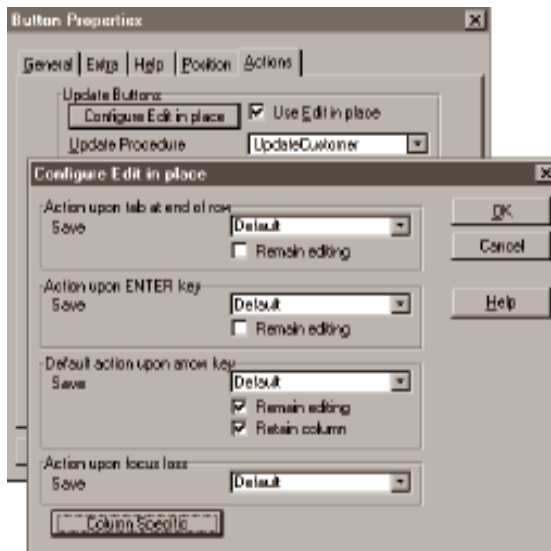
#### *Always (всегда)* Всегда сохранять запись.

#### *Never (никогда)*

Никогда не сохранять запись, отказаться от изменений.

#### *Prompted (по запросу)*

Запросить конечного пользователя, что делать - сохранить изменения, отказаться от изменений, или продолжить редактирование.



### Remain editing (продолжение редактирования)

Диалог **Configure Edit in place** предлагает настроить режим **Remain editing** (продолжение редактирования) для трех разных клавиатурных действий.



Отметьте эти режимы, чтобы продолжить редактирование при следующих клавиатурных действиях: клавиша Tab в конце строки, клавиша Enter, стрелки вверх или вниз. Выключите режимы, чтобы прекратить редактирование.

Retain column (оставаться в столбце)

Диалог **Configure Edit in place** предлагает настроить режим **Retain column** (оставаться в столбце) только для клавиш перемещения вверх и вниз. Отметьте эти режим, чтобы продолжить редактирование в новой строке в том же самом столбце списка. Выключите режим, чтобы продолжить редактирование с самого левого редактируемого столбца в новой строке.

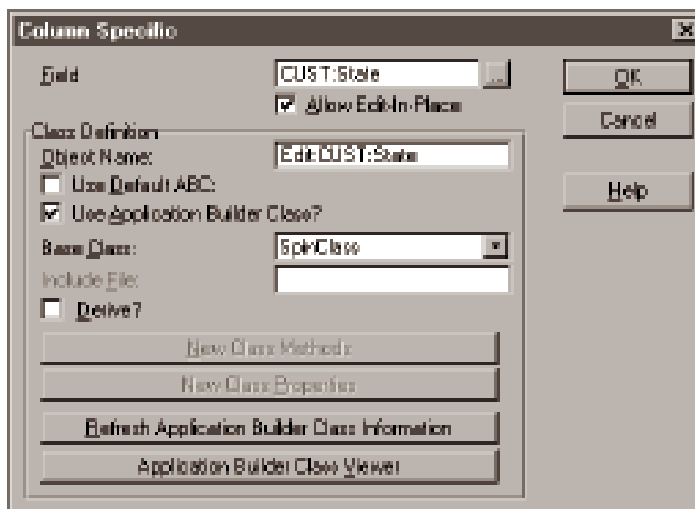
Column Specific (специфика колонок)

Нажмите эту кнопку, затем нажмите кнопку **Insert**, чтобы открыть диалог **Column Specific** (специфика колонок), для определения класса (и объекта), используемого при редактировании определенного поля списка. Диалог **Column Specific** (специфика колонок) содержит следующие настройки.

Нажмите кнопку с многоточием (...), чтобы выбрать редактируемое поле (или напечатайте в поле ввода имя поля. Это должно быть имя одного из полей, отображаемых в списке BrowseBox).

**Allow Edit-In-Place (разрешить редактирование-на-месте)**

Отметьте этот режим, чтобы позволить конечному пользователю редактировать поле. Выключите режим, чтобы не позволить конечному пользователю редактировать это поле. Основная цель этого режима - это выборочная блокировка редактирования-на-месте для некоторых, но не для всех, полей.



## Class Definition (определение класса)

Укажите Ваш собственный или сторонний класс. См. *Обзор шаблонов – Закладка Classes Options - Local* для полной информации об имеющихся возможностях.

По умолчанию шаблон BrowseUpdateButton генерирует код, использующий EditClass из ABC библиотеки (см. *EditClass* в главе *Классы просмотра*). Вы, однако, можете вывести SpinClass из EditClass, а затем использовать SpinClass для редактирования числовых полей BrowseBox.

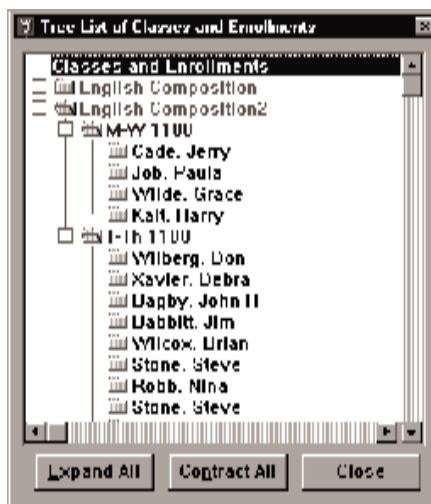
## Обзор RelationTree

Древовидный диалоговый элемент является списком, сформатированным для отображения сворачиваемого иерархического списка. Этот диалоговый шаблон предлагает альтернативу парадигме Просмотр-Форма. Единственный элемент RelationTree может заменить несколько пар Просмотр-Форма.

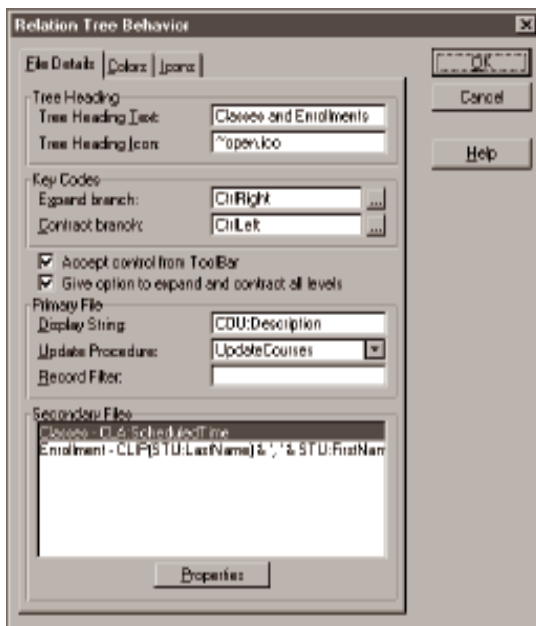
Используя диалоговый шаблон RelationTree, Вы можете определить множество связанных файлов, которые будут отображены на множестве уровней (вплоть до 29) иерархического списка *и соответствующие процедуры редактирования для каждого уровня*. Связанные файлы объявляются в файловой схеме (File Schematic) в виде первичного (родительского) файла и линейной цепочки связанных вторичных (дочерних) файлов (родитель-ребенок-внук).

Шаблон RelationTree применяет полную загрузку в QUEUE корневого уровня. Дочерние уровни загружаются по требованию при раскрытии ветки.

Совет: Этот шаблон не годится для баз данных с очень большим первичным файлом. Для больших файлов следует использовать диалоговый шаблон BrowseBox.



Знак плюс (+) указывает свернутый уровень, который раскрывается при нажатии на знак плюс (+). И наоборот, знак минус (-) указывает раскрытый уровень, который сворачивается при нажатии на знак минус (-).



Чтобы создать дерево, используйте диалоговый шаблон RelationTree:

1. Разместите в окне диалоговый шаблон RelationTree. При этом откроется **Форматер Списка (List Box Formatter)**. Используя Форматер Списка для включения раскраски (colorization), вывода пиктограмм, или горизонтальной прокрутки в древовидном диалоговом элементе (см. *Форматер Списка*). Не используйте Форматер Списка для задания полей древовидного элемента.

Совет: Древовидный элемент является списком с единственным полем, поэтому для выполнения горизонтальной прокрутки полосу прокрутки нужно задавать для столбца, а не для списка.

2. Нажмите в Форматере Списка (**List Box Formatter**) кнопку ОК.

3. Нажмите правую кнопку на Диалоговом шаблоне RelationTree и выберите **Actions (действия)** из всплывающего меню.

4. Нажмите кнопку **Files (файлы)**, чтобы сформировать файловую схему диалогового элемента.

Задайте первичный (родительский) файл и линейную цепь связанных дочерних файлов (родитель-ребенок-внук).

5. Закончите параметризацию шаблона RelationTree.

## **Возможности RelationTree**

---

Шаблон RelationTree обеспечивает следующие настройки:

### **File Details (файловые подробности)**

#### **Tree Heading Text (текст заголовка дерева)**

Необязательный текст, озаглавливающий вершину дерева. Tree Heading Text требуется, чтобы дать возможность пользователю добавлять записи в корневой уровень.

#### **Tree heading Icon (пиктограмма заголовка дерева)**

Необязательная пиктограмма для вершины дерева. Для включения этой возможности нужно разрешить использование пиктограмм (Icons) в Форматере Списка.

#### **Expand Branch (раскрытие ветви)**

Задайте клавишу для раскрытия выбранного узла списка – вывода его потомков. Нажмите кнопку с многоточием (...) для того, чтобы выбрать одну из специальных клавиш, как, например, Esc, Tab. или Enter. См. *Диалоговые элементы и их свойства – Общие атрибуты диалоговых элементов – Формирование атрибута KEY* для полной информации об этом диалоге.

#### **Contract Branch (сворачивание ветви)**

Задайте клавишу для сворачивания выбранного узла списка – упрятывания его потомков. Нажмите кнопку с многоточием (...) для того, чтобы выбрать одну из специальных клавиш, как, например, Esc, Tab. или Enter. См. *Диалоговые элементы и их свойства – Общие атрибуты диалоговых элементов – Формирование атрибута KEY* для полной информации об этом диалоге.

#### **Accept control from Toolbar (использовать органы управления инструментальной панели)**

Включите этот переключатель, чтобы обрабатывать навигационные и другие управляющие просмотром события, сгенерированные диалоговым шаблоном FrameBrowseControl на инструментальной панели приложения (APPLICATION). Дополнительную информацию о кнопках на инструментальной панели и их функционировании см. в

*FrameBrowseControl*. Выключите этот переключатель, чтобы выключить для этой процедуры кнопки на инструментальной панели.

### **Give option to expand and contract all levels (дать возможность сворачивать и разворачивать все уровни)**

Определяет, что всплывающее по правой кнопке меню для RelationTree будет содержать пункты “Expand All” (раскрыть все) и “Contract All” (свернуть все).

### **Настройки для первичного файла**

#### **Display String (строка вывода)**

Имя поля или текст для отображения на уровне первичного файла. Это может быть любым правильным Clarion выражением, например:  
`CLIP(CUST:LastName)& ' &CUST:FirstName`

#### **Update Procedure (процедура редактирования)**

Процедура редактирования, которая вызывается для первичного файла. Процедура может быть вызвана с помощью всплывающего при нажатии на правую кнопку меню, которое обеспечивается автоматически при определении процедуры редактирования. Стандартный текст всплывающего меню - “Insert” (вставить), “Change” (изменить) и “Delete” (удалить).

Процедуру можно также вызвать с помощью шаблона RelationTreeUpdateButtons – см. далее. При использовании диалогового шаблона RelationTreeUpdateButtons, всплывающее меню наследует текст кнопок.

#### **Record Filter (фильтр записей)**

Напечатайте правильное Clarion выражение, чтобы ограничить содержимое списка только теми записями, для которых выражение оценивается как истинное (не нуль или не пробел). Процедура просматривает последовательно все воспроизводимые записи, чтобы отобразить только те из них, которые отвечают условию фильтра.

Вы должны связать (BIND) любое файловое поле, которое используется в выражении фильтра. См. BIND в *Справочнике по языку* для дополнительной информации.

### **Colors (Primary File) (палитра (первичный файл))**

Эта закладка доступна, только если Вы отметите **Color Cells** в **List Field Properties** (свойства поля списка) в Форматере Списка.

#### **Default Colors (стандартная палитра)**

Для того, чтобы определить стандартную палитру для выводной строки

первичного файла, напечатайте в полях ввода цветовые EQUATE-ы (из \LIBSRC\EQUATES.CLW) или нажмите кнопки с многоточием (...) и выберите цвета в диалоге выбора цвета (**Select Color**).

#### Conditional Color Assignments (условные назначения цветов)

Для того, чтобы определять условные цвета для выводной строки первичного файла, нажмите кнопку **Insert**. Это откроет диалог условных цветовых назначений (**Conditional Color Assignments**).

#### Conditional Color Assignments (условные назначения цветов)

Этот диалог позволяет Вам определять условные цвета для выводной строки первичного файла.

#### Condition (условие)

Напечатайте правильное Clarion выражение, которое должно оцениваться в период исполнения, а затем напечатайте в полях ввода цветовые EQUATE-ы (из \LIBSRC\EQUATES.CLW) или нажмите кнопки с многоточием (...) и выберите цвета в диалоге выбора цвета (**Select Color**).

В период выполнения эти условия оцениваются, и используются цвета для первого в списке истинного условия.

#### Icons (Primary File) (пиктограммы (первичный файл))

Эта закладка доступна, только если Вы отметили **Icons** в **List Field Properties** (свойства поля списка) в Форматере Списка.

#### Default Icon (стандартная пиктограмма)

Для того, чтобы определить стандартную пиктограмму для выводной строки первичного файла, напечатайте во поле ввода имя файла пиктограммы.

#### Conditional Icon Usage (условное использование пиктограмм)

Нажмите кнопку **Insert** для того, чтобы определять условные пиктограммы для выводной строки первичного файла. Это откроет диалог **Conditional Icon Usage** (условное использование пиктограмм).

#### Conditional Icon Usage (условное использование пиктограмм)

Этот диалог позволяет Вам определить условные пиктограммы для выводной строки первичного файла.

#### Condition (условие)

Наберите правильное Clarion выражение для оценки в период исполнения.

#### Icon (пиктограмма)

Напечатайте во вводном поле имя файла пиктограммы.

В период выполнения эти условия оцениваются и используется пиктограмма для первого в списке истинного условия.

### **Настройки для дочернего файла**

Параметры настройки для дочернего файла идентичны параметрам первичного файла. Выделите дочерний файл, а затем нажмите кнопку **Properties** (свойства) под списком **Secondary Files** (дочерние файлы). См. *Обзор RelationTree* для информации о том, как определить дочерние файлы с помощью диалога **Select File** (выбор файла).

### **RelationTree Embed Points (точки встраивания для RelationTree)**

Диалоговый шаблон RelationTree обеспечивает понятный набор точек встраивания, что позволяет полностью настроить поведение диалогового элемента.

### **RelationTreeUpdateButtons**

Этот диалоговый шаблон размещает три кнопки (**Insert (вставить)**, **Change (изменить)** и **Delete (удалить)**), которые позволяют конечному пользователю вызывать соответствующую выбранному уровню RelationTree процедуру редактирования. У этого элемента нет настроек. Процедура редактирования для каждого уровня определяется в диалоговом шаблоне RelationTree.

Кнопки Change (изменить) и Delete (удалить) относятся к текущей выделенной записи. Кнопка Insert (добавить) добавляет запись потомка (на следующем, более низком уровне структуры дерева).

### **RelationTreeUpdate Embed Points (точки встраивания RelationTreeUpdate)**

Диалоговый шаблон RelationTreeUpdateButtons обеспечивает понятный набор точек встраивания, что позволяет полностью настроить поведение диалогового элемента.

## Другие оконные диалоговые шаблоны

### CancelButton

Шаблон `CancelButton` размещает однокнопочный диалоговый элемент с надписью **Cancel** (отмена). Эта кнопка позволяет пользователю закрыть окно и предоставляет разработчику удобное место, где можно добавить код для отмены действий перед закрытием процедуры. Сгенерированный исходный код устанавливает флаг “Request Cancelled” (запрос отменен) и закрывает оконную процедуру.

Шаблон `CancelButton` не имеет никаких вариантов конфигурации.

Вы можете включить необходимый для “наведения порядка” выполняемый код в точке встраивания.

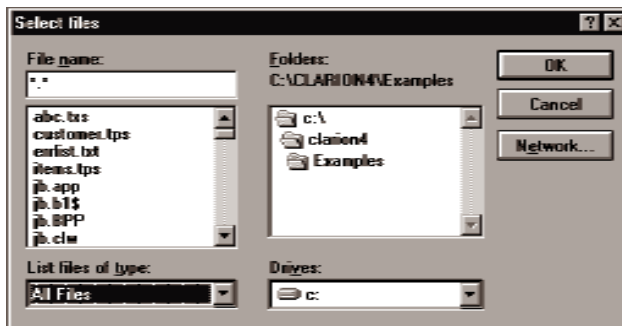
### CloseButton

Шаблон `CloseButton` размещает однокнопочный диалоговый элемент с надписью **Close** (закрыть). Сгенерированный исходный код устанавливает флаг “Request Completed” (запрос выполнен) и закрывает процедуру окна.

Шаблон `CloseButton` не имеет никаких вариантов конфигурации.

### DOSFileLookup

Шаблон `DOSFileLookup` размещает кнопку с многоточием (...), которая открывает стандартный файловый диалог Windows.



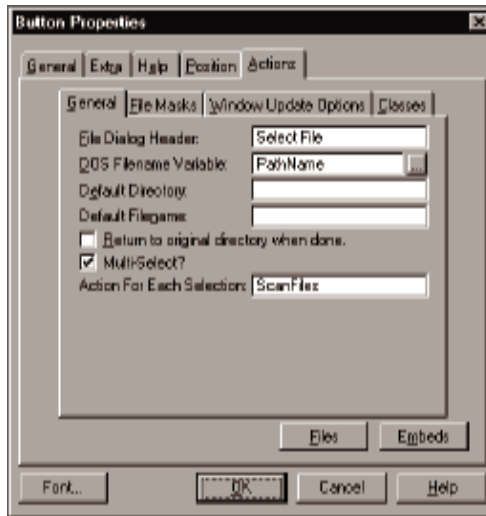
Вы можете задать маски файлов, стандартные имя каталога и файла, и переменную, которой будет присвоено имя выбранного конечным пользователем файла.



Кроме того, имеется возможность выбрать множество файлов и определить код для обработки каждого указанного файла. Шаблон генерирует цикл (LOOP) для обработки всех выбранных файлов.

Шаблон DOSFileLookup обеспечивает следующие настройки:

### General (общие)



**File Dialog Header** (заголовок файлового диалога)

Напечатайте текст заголовка файлового диалога Windows.

**DOS Filename Variable** (переменная для имени файла)

Нажмите кнопку с многоточием (...), чтобы выбрать с помощью диалога **File Schematic (схема файлов)** переменную, которая получит результат выбора конечным пользователем. Вы также можете напечатать имя переменной непосредственно в поле ввода.

**Default Directory** (стандартный каталог)

Определите начальный каталог для файлового диалога Windows. Если пусто, файловый диалог откроется в рабочем каталоге.

**Default Filename** (стандартное имя файла)

Определите начальное имя файла для файлового диалога Windows. Если пробел, файловый диалог откроется без начального имени файла.

**Return to original directory when done** (возврат по завершении в исходный каталог)

Отметьте этот режим, чтобы восстановить рабочий каталог на свое значение до файлового поиска.

### **Multi-Select? (множественный выбор?)**

Отметьте этот режим, чтобы разрешить выбор одного *или более* файлов.

### **Action For Each Selection (действия для каждого выбора)**

Напечатайте правильный оператор языка Clarion, который будет выполняться для каждого выбранного файла – обычно вызов процедуры. Вы можете передать **FileName Variable** в качестве параметра процедуры. Для выполнения кода, который Вы задаете для каждого выбранного файла, шаблон генерирует цикл. Для каждого прохода по циклу сгенерированный код загружает в переменную **FileName Variable** соответствующее имя файла.

### **File Masks (маски файлов)**

#### **Use a variable file mask (использовать переменные маски файлов)**

Отметьте этот режим, чтобы использовать переменную для масок файлов. Это задействует поле параметра **Variable Mask Value** для ввода имени переменной и выключает поля параметров **Mask Description (описание маски)**, **File Mask (маска файла)**, и **More File Masks (дополнительные маски файлов)**.

#### **Mask Variable (переменная для маски)**

Указывает переменную, содержащую маску файлов. См. *FILEDIALOG* в *Справочнике по языку* для информации о содержимом этой переменной.

#### **File Mask Description (описание маски файла)**

Напечатайте описание типа файла. Строка появляется в выпадающем списке в файловом диалоге Windows. Вы можете добавить дополнительные маски, нажимая кнопку **More File Masks (дополнительные маски файлов)**.

#### **File Mask (маска файла)**

Напечатайте спецификацию маски файла, как, например, "\*.TXT" или используйте для этой маски несколько шаблонов, разделяя их точкой с запятой, как, например, "\*.BMP;\*.GIF". Вы можете добавить дополнительные маски, нажимая кнопку **More File Masks (дополнительные маски файлов)**.

#### **More File Masks (дополнительные маски файлов)**

Нажмите эту кнопку, чтобы добавить дополнительные маски файлов. Эти маски доступны конечному пользователю через выпадающий список **List files of type (Тип файлов)** файлового диалога Windows.

## **Window Update Options (режимы обновления окна)**

### **Update entire window? (обновить все окно?)**

Отметьте этот режим, чтобы обновить содержимое всех диалоговых элементов окна после завершения выбора и обработки файлов. Выключите режим, чтобы указать лишь отдельные обновляемые поля.

### **Update Selected Fields (обновить указанные поля)**

Нажмите эту кнопку, чтобы указать специфические поля, которые должны обновляться после завершения выбора и обработки файлов. Шаблон генерирует оператор для каждого указанного поля. См. *DISPLAY* в *Справочнике по языку*.

## **Classes (классы)**

Закладка Classes (классы) позволяет Вам воздействовать на используемые шаблоном классы (и объекты). Вы можете воспользоваться стандартными ABC классами и объектами (рекомендуется), или можно определить свои собственные классы или классы третьих лиц. Вывод вашего собственного класса может дать Вам очень тонкое управление процедурой в тех случаях, когда стандартные ABC классы не вполне точно отвечают Вашим потребностям.

См. *Обзор шаблонов – Закладка Classes Options - Local* для полной информации об имеющихся возможностях.

## **FieldLookupButton**

Шаблон FieldLookupButton размещает кнопку с многоточием (...), которая позволяет Вам “найти” значение в справочном файле, как, например, в файле штатов. Чтобы поместить кнопку поиска, щелкните около вводного диалогового элемента.

Шаблон FieldLookupButton обеспечивает следующие настройки:

### **Control with Lookup (элемент с поиском)**

Выберите диалоговый элемент, с которым будет связан поиск, путем выбора из выпадающего списка метки поля. Обычно это ENTRY поле.

Выбранное поле должно иметь связанную с ним процедуру справочника. Для того, чтобы назначить процедуру справочника, нажмите правой кнопкой на поле ввода, затем выберите **Actions** для доступа к его настройкам.

## **FileDrop**

Шаблон FileDrop размещает в окне файлово-загружаемый, прокручиваемый, выпадающий список. В период исполнения конечный пользователь может выбрать элемент списка, а затем поместить значение записи выбранного пункта в заданное целевое поле. Можно отображать одно поле (как, например, поле описания), а

присваивать другое поле (как, например, поле кода) выбранной записи (см., How Do I... в оперативной справке (on-line help)).

Совет: Установите атрибут DROP в нуль (0), чтобы вывести список, а не выпадающий список.

Непосредственно перед тем, как Вы разместите в окне диалоговый шаблон FileDrop, Генератор Приложений предложит Вам определить файл для отображения в выпадающем списке. Укажите файл в диалоге **Select Field (выбор поля)**. Вы должны будете также указать поле для использования в качестве USE переменной списка LIST; однако, выбранное Вами поле имеет значение, только если Вы отображаете одно поле, а присваиваете другое.

Немедленно после того, как Вы разместите диалоговый шаблон FileDrop, Генератор Приложений откроет Форматер Списка, где Вы сможете определить поля для отображения в Вашем списке. Вы можете определить поле, содержащее искомую величину, а также другие поля из того же или из связанных файлов. См. *Форматер Списка* для дополнительной информации.

После того, как Вы определили списковые поля и вернулись в проектируемое окно, вызовите всплывающее меню правым-нажатием на диалоговом элементе, а затем выберите **Actions**, чтобы заполнить следующие настройки FileDrop:

### **General (общие)**

#### **Field to Fill From (из какого поля заполнять)**

Поле справочного файла, значение которого присваивается целевому полю (Target Field). Нажмите кнопку с многоточием (...), чтобы выбрать его в диалоге **Select Field (выбор поля)**.

#### **Target Field (целевое поле)**

Поле, которая получает значение из Field to Fill From (из какого поля заполнять). Нажмите кнопку с многоточием (...), чтобы выбрать его в диалоге **Select Field (выбор поля)**.

#### **More Field Assignments (дополнительные поля присваивания)**

Нажмите эту кнопку, чтобы определить дополнительные присваивания значений из выбранной записи.

#### **Record Filter (фильтр записи)**

Напечатайте правильное Clarion выражение, чтобы ограничить содержимое списка только теми записями, для которых выражение оценивается как

истинное (не ноль или не пробел). Процедура просматривает последовательно все воспроизводимые записи, чтобы отобрать только те из них, которые отвечают условию фильтра. Фильтры обычно значительно медленнее, чем ограничения диапазона.

Вы должны связать (BIND) любое файловое поле, которое используется в выражении фильтра. Закладка **Hot Fields** позволяет Вам связывать (BIND) поля.

### **Default to first entry if USE variable empty (установить на начало, если USE-переменная пуста)**

Отметьте этот режим, чтобы обеспечить стандартное начальное значение - выпадающий список всегда первоначально не пуст (если только первая запись файла не является пустой).

### **Range Limits (границы диапазона)**

Эта закладка доступна, только если Вы укажете ключ файла в диалоге **File Schematic Definition (определение файловой схемы)**. Поскольку ограничения по диапазону поля используют ключи, они обычно значительно быстрее, чем фильтры.

### **Range Limit Field (поле с ограниченным диапазоном)**

Совместно с **Range Limit Type** (тип ограничений по диапазону) определяет включаемую в процесс запись или группу записей. Нажимая кнопку с многоточием (...), выберите ключевое поле, по которому будут ограничиваться записи.

### **Range Limit Type (тип ограничений по диапазону)**

Определяет тип применяемых ограничений по диапазону. Выберите один из вариантов из выпадающего списка.

### **Current Value (текущее значение)**

Ограничивает ключевое поле его текущим значением.

### **Single Value (единственное значение)**

Позволяет ограничить ключевое поле единственным значением. Укажите переменную, содержащую это значение в поле **Range Limit Value (значение ограничения)**.

### **Range of Values (диапазон значений)**

Позволяет ограничить ключевое поле диапазоном значений. Укажите переменные, содержащие верхний и нижний пределы значений поля в полях **Low Limit Value (нижний предел)** и **High Limit Value (верхний предел)**.

## File Relationship (связь файлов)

Позволяет ограничить текущее значение ключевого поля значением в связанном (родительском) файле. Нажмите кнопку с многоточием (...), чтобы выбрать ограничивающий диапазон файл. Это ограничивает процесс включением только тех дочерних записей, которые соответствуют текущей записи родительского файла. Например, если Ваш документ был списком заказов, Вы могли бы ограничить вывод только заказами текущего клиента.

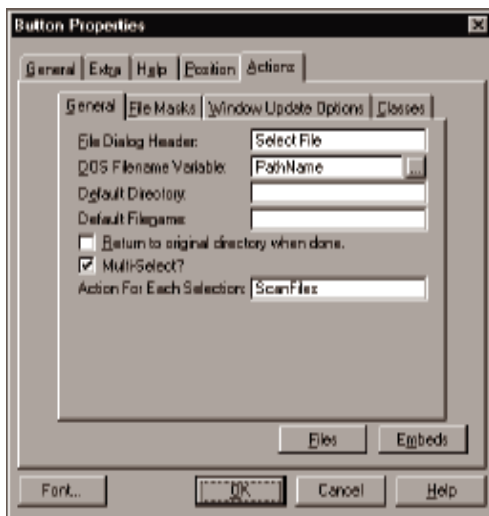
## Colors (палитра)

Эта закладка доступна, только если Вы отметили режим **Color Cells** (цветные ячейки) в Форматере Списка. На ней показан список столбцов FileDrop, которые можно окрашивать.

Для того, чтобы определять стандартные и любые условные цвета, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это откроет диалог **Customize Colors (настройка цветов)**.

## Customize Colors (настройка цветов)

Этот диалог позволяет определять стандартные и условные цвета окрашивания фона и текста для обычных (невывбранных) и для вывбранных колонок.



## Conditional Color Assignments (условные назначения цветов)

Под разделом встроенных цветов находится список **Conditional Color Assignments** (условные назначения цветов). Этот список позволяет Вам

определить применяемые цвета в тех случаях, когда выражение оценивается как истина (не ноль или не пробел). Нажмите кнопку **Insert** для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета.

В период исполнения выражения оцениваются, и используются цвета для первого истинного выражения.

### **Icons (пиктограммы)**

Эта закладка доступна, только если Вы отметили режим **Icons** в Форматере Списка. На ней показан список столбцов **BrowseBox**, в которых можно вывести пиктограммы.

Для того, чтобы определять стандартные и любые условные пиктограммы, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это откроет диалог **Customize Icons (настройка пиктограмм)**.

### **Customize Icons (настройка пиктограмм)**

Этот диалог позволяет определять стандартные и условные пиктограммы для столбца **FileDrop**.

### **Default Icon (стандартная пиктограмма)**

Стандартно выводимая пиктограмма. Напечатайте имя файла пиктограммы (.ICO).



### **Conditional Icon Usage (использование условных пиктограмм)**

Под разделом **Default Icon (стандартная пиктограмма)** находится список **Conditional Icon Usage (использование условных пиктограмм)**. Этот список позволяет Вам определить применяемые пиктограммы в тех случаях, когда выражение оценивается как истина (не нуль или не пробел). Нажмите кнопку Insert для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета.

В период исполнения выражения оцениваются и используется пиктограмма для первого истинного выражения.

### **Hot Fields (“горячие” поля)**

Когда Вы выбираете закладку Hot Fields, Вы можете указать, какие из полей, не попавших в список, нужно добавить к QUEUE. При перемещениях по файлу сгенерированная исходная программа считывает данные для этих полей из QUEUE, а не с диска. Это ускоряет обновление списка.



Указание “горячих” полей также позволяет Вам размещать за пределами BrowseBox диалоговые элементы, которые будут обновляться всякий раз при выборе в списке новой записи. Поля первичного ключа и текущего ключа всегда включаются в QUEUE, поэтому их не следует включать в список “горячих” полей.

Этот диалог также позволяет Вам связывать (BIND) поля. Вы должны связать любое поле, которое используется в выражении фильтра или для подсчета итогов.

### **Sort Fields (поля сортировки)**

Эта закладка позволяет Вам добавить поля, по которым будут сортироваться записи выпадающего списка. Поля сортировки – это дополнительные поля к заданному для FileDrop ключу. Чтобы добавить в список поля, нажмите кнопку Insert.

### **Classes (классы)**

Закладка Classes (классы) позволяет Вам воздействовать на используемые шаблоном классы (и объекты). Вы можете воспользоваться стандартными ABC классами и объектами (рекомендуется), или можно определить свои собственные классы или классы третьих лиц. Вывод вашего собственного класса может дать Вам



очень тонкое управление процедурой в тех случаях, когда стандартные ABC классы не вполне точно отвечают Вашим потребностям.

См. *Обзор шаблонов – Закладка Classes Options - Local* для полной информации об имеющихся возможностях.

### **Other Prompts (другие параметры)**

Свойства списка для этого диалогового элемента такие же, как и для списка, тем не менее, следующие параметры могут потребовать некоторых дополнительных разъяснений:

- Use**            Может содержать либо метку поля, либо метку переменной, которая получит значение первого размещенного в списке поля. В контексте диалогового шаблона FileDrop эта функция заменяется более гибкой установкой Target Field.
- From**            В этом поле стандартно формируется значение Queue:FileDrop. Queue:FileDrop – это метка QUEUE, которая используется шаблоном для заполнения списка. Обычно Вам не следует изменять это значение.
- Mark**            Содержит метку Queue:FileDrop:Mark поля QUEUE, что позволяет пользователю выбирать более, чем один элемент списка. Поле Queue:FileDrop:Mark содержит 1 для выбранных элементов и 0 - для невыбранных элементов.

### **FileDropCombo**

Шаблон FileDropCombo генерирует код для отображения файла данных в прокручиваемом списке, выбора из списка одной из записей и последующего присваивания значения из выбранной записи указанному целевому полю. Обратите внимание, что Вы можете отображать одно поле (как, например, поле описания), а присваивать другое поле (как, например, поле кода) выбранной записи (см., How Do I... в оперативной справке (on-line help)). Также, поскольку шаблон базируется на элементе COMBO, сгенерированный код, получая несуществующие в отображаемом списке вводные значения, может добавить эти новые значения в справочный файл.

Непосредственно перед тем, как Вы разместите в окне диалоговый шаблон FileDropCombo, Генератор Приложений предложит Вам определить файл для отображения в выпадающем списке. Укажите файл в диалоге **Select Field (выбор поля)**. Вы должны будете также указать поле для использования в качестве USE переменной списка COMBO. USE переменная важна, когда Вы допускаете

редактирование (Allow Updates) из FileDropCombo, или если Вы отображаете одно поле, а присваиваете другое. См. *Update Behavior (поведение при редактировании)* для дополнительной информации.

Немедленно после того, как Вы разместите диалоговый шаблон FileDropCombo, Генератор Приложений откроет Форматер Списка, где Вы сможете определить поля для отображения в Вашем списке. Вы можете определить поле, содержащее искомую величину, а также другие поля с сопутствующей информацией. См. *Форматер Списка* для дополнительной информации.

После того, как Вы определили списковые поля и вернулись в проектируемое окно, вызовите всплывающее меню правым-нажатием на диалоговом элементе, а затем выберите **Actions**, чтобы заполнить следующие настройки FileDropCombo:

### **General (общие)**

#### **Field to Fill From (из какого поля заполнять)**

Поле справочного файла, значение которого присваивается целевому полю (Target Field). Нажмите кнопку с многоточием (...), чтобы выбрать его в диалоге **Select Field (выбор поля)**.

#### **Target Field (целевое поле)**

Поле, которое получает значение из Field to Fill From (из какого поля заполнять). Нажмите кнопку с многоточием (...), чтобы выбрать его в диалоге **Select Field (выбор поля)**.

#### **More Field Assignments (дополнительные поля присваивания)**

Нажмите эту кнопку, чтобы определить дополнительные присваивания значений из выбранной записи.

#### **Record Filter (фильтр записи)**

Напечатайте правильное Clarion выражение, чтобы ограничить содержимое списка только теми записями, для которых выражение оценивается как истинное (не нуль или не пробел). Процедура просматривает последовательно все воспроизводимые записи, чтобы отобрать только те из них, которые отвечают условию фильтра. Фильтры обычно значительно медленнее, чем ограничения диапазона.

Вы должны связать (BIND) любое файловое поле, которое используется в выражении фильтра. Закладка **Hot Fields** позволяет Вам связывать (BIND) поля.

**Default to first entry if USE variable empty (установить на начало, если USE-переменная пуста)**

Отметьте этот режим, чтобы обеспечить стандартное начальное значение - выпадающий список всегда первоначально не пуст (если только первая запись файла не является пустой).

**Remove duplicate entries (удалить повторения)**

Отметьте этот режим, чтобы удалить из списка дубликаты.

**Keep View synchronized with Selection? (поддерживать синхронизацию VIEW и выбора)**

Отметьте этот режим для обновления буферов VIEW, чтобы они соответствовали выбранной записи списка.

**Case Sensitive matches? (регистрочувствительное сопоставление?)**

Отметьте этот режим, чтобы при сравнении значений введенной величины со значениями в справочном файле учитывался регистр.

**Range Limits (границы диапазона)**

Эта закладка доступна, только если Вы укажете ключ файла в диалоге **File Schematic Definition (определение файловой схемы)**. Поскольку ограничения по диапазону поля используют ключи, они обычно значительно быстрее, чем фильтры.

**Range Limit Field (поле с ограниченным диапазоном)**

Совместно с **Range Limit Type** (тип ограничений по диапазону) определяет включаемую в процесс запись или группу записей. Нажимая кнопку с многоточием (...), выберите ключевое поле, по которому будут ограничиваться записи.

**Range Limit Type (тип ограничений по диапазону)**

Определяет тип применяемых ограничений по диапазону. Выберите один из вариантов из выпадающего списка.

**Current Value (текущее значение)**

Ограничивает ключевое поле его текущим значением.

**Single Value (единственное значение)**

Позволяет ограничить ключевое поле единственным значением. Укажите переменную, содержащую это значение в поле **Range Limit Value (значение ограничения)**.

**Range of Values (диапазон значений)**

Позволяет ограничить ключевое поле диапазоном значений. Укажите переменные, содержащие верхний и нижний пределы значений поля в полях

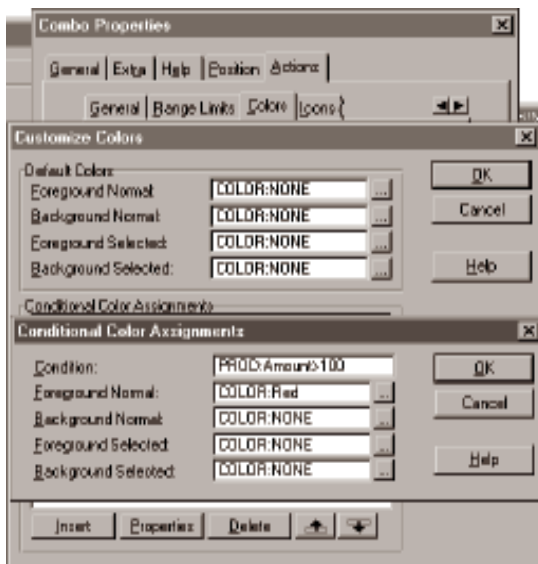
## Low Limit Value (нижний предел) и High Limit Value (верхний предел).

### File Relationship (связь файлов)

Позволяет ограничить текущее значение ключевого поля значением в связанном (родительском) файле. Нажмите кнопку с многоточием (...), чтобы выбрать ограничивающий диапазон файл. Это ограничивает процесс включением только тех дочерних записей, которые соответствуют текущей записи родительского файла. Например, если Ваш документ был списком заказов, Вы могли бы ограничить вывод только заказами текущего клиента.

## Colors (палитра)

Эта закладка доступна, только если Вы отметили режим **Color Cells** (цветные ячейки) в Форматере Списка. На ней показан список столбцов FileDropCombo, которые можно окрашивать.



Для того, чтобы определять стандартные и любые условные цвета, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это откроет диалог **Customize Colors (настройка цветов)**.

### Customize Colors

Этот диалог позволяет определять стандартные и условные цвета окрашивания фона и текста для обычных (невывбранных) и для вывбранных колонок.

## Conditional Color Assignments (условные назначения цветов)

Под разделом встроенных цветов находится список **Conditional Color Assignments** (условные назначения цветов). Этот список позволяет Вам определить применяемые цвета в тех случаях, когда выражение оценивается как истина (не ноль или не пробел). Нажмите кнопку Insert для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета.

В период исполнения выражения оцениваются и используются цвета для первого истинного выражения.

## Icons (пиктограммы)

Эта закладка доступна, только если Вы отметили режим **Icons** в Форматере Списка. На ней показан список столбцов BrowseBox, в которых можно вывести пиктограммы.



Для того, чтобы определять стандартные и любые условные пиктограммы, выберите имя поля в столбце, затем нажмите кнопку **Properties** (свойства). Это откроет диалог **Customize Icons (настройка пиктограмм)**.

## **Customize Icons (настройка пиктограмм)**

Этот диалог позволяет определять стандартные и условные пиктограммы для столбца FileDropCombo.

## **Default Icon (стандартная пиктограмма)**

Стандартно выводимая пиктограмма. Напечатайте имя файла пиктограммы (.ICO).

### **Conditional Icon Usage (использование условных пиктограмм)**

Под разделом **Default Icon (стандартная пиктограмма)** находится список **Conditional Icon Usage (использование условных пиктограмм)**. Этот список позволяет Вам определить применяемые пиктограммы в тех случаях, когда выражение оценивается как истина (не ноль или не пробел). Нажмите кнопку Insert для того, чтобы добавить к списку новое выражение и сопутствующие ему цвета.

В период исполнения выражения оцениваются и используется пиктограмма для первого истинного выражения.

### **Update Behavior (поведение при редактировании)**

Эта закладка позволяет Вам использовать вводное поле COMBO для инициализации добавления новой записи к справочному файлу. Если пользователь занесет во вводное поле величину, который еще нет в списке, сгенерированный код может непосредственно добавить новую запись или вызвать отдельную процедуру для добавления нового значения.

#### Allow Updates (разрешить обновление)

Выключите этот режим, чтобы отвергнуть данные, которые не существуют в справочном файле. Новые (неподтвержденные) данные *не* будут добавляться к справочному файлу.

Отметьте этот режим, чтобы добавить новые данные к справочному файлу и разрешить ввод параметра **Update Procedure (процедура редактирования)**.

#### Update Procedure (процедура редактирования)

Укажите процедуру, которая будет вызываться для добавления новой записи или оставьте это поле пустым, если не нужна никакая процедура редактирования.

Процедура редактирования может не требоваться только для справочных файлов с единственным обязательным полем (поле, указанное в USE переменной COMBO). Не-USE поля очищаются, если только поле не имеет ограничений диапазона или не является автоинкрементным.

### **Hot Fields (“горячие” поля)**

Используйте закладку Hot Fields для указания полей, которые не отображаются в списке и должны быть добавлены в QUEUE. При перемещениях по файлу, сгенерированная исходная программа считывает данные для этих полей из QUEUE, а не с диска. Это ускоряет обновление списка.

Указание “горячих” полей позволяет Вам эффективно обновлять другие диалоговые элементы всякий раз при выборе в списке новой записи. Поля первичного ключа и текущего ключа всегда включаются в QUEUE, поэтому их не следует включать в список “горячих” полей. Для добавления поля в список нажмите кнопку Insert.

### **Sort Fields (поля сортировки)**

Эта закладка позволяет Вам добавить поля, по которым будут сортироваться записи выпадающего списка. Поля сортировки – это дополнительные поля к заданному для FileDrop ключу. Чтобы добавить в список поля, нажмите кнопку Insert.

### **Classes (классы)**

Закладка Classes (классы) позволяет Вам воздействовать на используемые шаблоном классы (и объекты). Вы можете воспользоваться стандартными ABC классами и объектами (рекомендуется) или можно определить свои собственные классы или классы третьих лиц. Вывод вашего собственного класса может дать Вам очень тонкое управление процедурой в тех случаях, когда стандартные ABC классы не вполне точно отвечают Вашим потребностям.

См. *Обзор шаблонов – Закладка Classes Options - Local* для полной информации об имеющихся возможностях.

### **Other Prompts (другие параметры)**

Свойства списка для этого диалогового элемента такие же, как и для списка, тем не менее, следующие параметры могут потребовать некоторых дополнительных разъяснений:

**Use**            Может содержать либо метку поля, либо метку переменной, которая получит значение первого размещенного в списке поля. В контексте диалогового шаблона FileDrop эта функция заменяется более гибкой установкой Target Field; однако, USE переменная важна, когда Вы разрешаете обновление из FileDropCombo (см. *Update Behavior (поведение при редактировании)* для дополнительной информации).

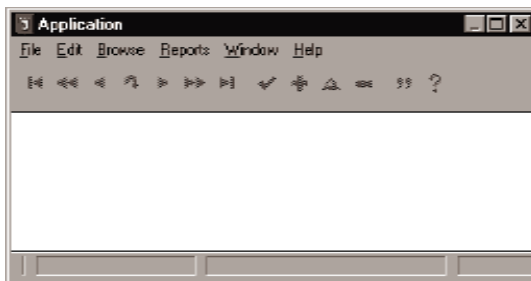
**From**            В этом поле стандартно формируется значение Queue:FileDropCombo. Queue:FileDropCombo – это метка QUEUE, которая используется шаблоном для заполнения списка. Обычно Вам не следует изменять это значение.  
Содержит метку Queue:FileDropCombo:Mark поля QUEUE, что позволяет

пользователю выбирать более, чем один элемент списка. Поле Queue:FileDropCombo:Mark содержит 1 для выбранных элементов и 0 для невыбранных элементов.

## FrameBrowseControl

Шаблон FrameBrowseControl размещает на инструментальной панели MDI приложения (APPLICATION) (Frame-процедура) тринадцать (13) стандартных командных кнопок. Когда пользователь нажимает эти кнопки, сгенерированный шаблоном код посылает диалоговым элементам активной процедуры соответствующие коды событий (прокрутить вверх, прокрутить вниз, добавить, изменить, удалить, справка, и т.п.).

Конец: Вы можете удалить кнопки, которые ваше приложение не использует. Например, ABC Templates по умолчанию не использует (расположите) кнопка.



Кнопки предназначены для работы с диалоговым шаблоном BrowseBox, диалоговым шаблоном RelationTree и распределенным шаблоном FormVCRControls; это означает, что кнопки остаются выключенными, пока программа не вызовет процедуру с шаблоном BrowseBox или RelationTree у которой включен режим **Accept browse control from Toolbar**, или BrowseBox процедура не вызовет Form-процедуру с распределенным шаблоном FormVCRControls.

Кроме того, окно вызванной процедуры должно иметь атрибут MDI, однако не стоит беспокоиться, поскольку стандартные шаблоны просмотра и формы по умолчанию объявляют MDI окна, и Вам не нужно что-либо делать по этому поводу. BrowseBox и RelationTree шаблоны также стандартно устанавливают режим **Accept browse control from Toolbar**, поэтому Вам опять не нужно что-либо делать по этому поводу.

Кнопки инструментальной панели шаблона FrameBrowseControl действуют следующим образом:





Прокручивает BrowseBox на первую строку или на предшествующую родительскую запись в RelationTree. В процедуре формы перед перемещением сохраняет текущую запись.



Прокручивает BrowseBox на одну страницу вверх или на предыдущую запись того же уровня в RelationTree. В процедуре формы перед перемещением сохраняет текущую запись.



Прокручивает BrowseBox на одну строку вверх или на предыдущую запись любого уровня в RelationTree. В процедуре формы перед перемещением сохраняет текущую запись.



Поиск записи в BrowseBox. Эта кнопка задействуется, только если в BrowseBox определен локатор типа поле ввода или пошаговый локатор. Сведения об объявлении таких локаторов см. в *Диалоговые шаблоны - BrowseBox*.



Прокручивает BrowseBox на одну строку вниз или на следующую запись любого уровня в RelationTree, раскрывая, при необходимости, ветки дерева. В процедуре формы перед перемещением сохраняет текущую запись.



Прокручивает BrowseBox вниз на одну страницу вниз или на следующую запись того же уровня в RelationTree. В процедуре формы перед перемещением сохраняет текущую запись.



Прокручивает BrowseBox на последнюю запись или на следующую родительскую запись в RelationTree. В процедуре формы перед перемещением сохраняет текущую запись.



Выбирает выделенную строку BrowseBox. Это возможно, только когда процедура вызывается для выбора записи. Например, когда вызывается как справочник.



В BrowseBox вызывает процедуру формы для добавления новой записи. В RelationTree вызывает процедуру формы для добавления дочерней записи к текущей выбранной записи. В процедуре формы добавляет другую запись того же самого типа.



Вызывает процедуру формы, чтобы изменить выделенную в BrowseBox или RelationTree запись.



Удаляет выделенную в BrowseBox или RelationTree запись. В BrowseBox поведение при удалении определяется установками параметров в диалоговом шаблоне Update Buttons.



Только в процедуре формы заносит соответствующее значение из предыдущей обработанной записи (значение из буфера записи) в имеющее фокус поле. Или другими словами, повторяет значение из последней сохраненной записи.



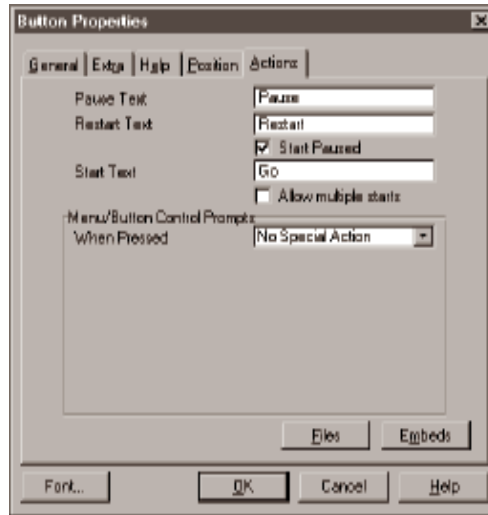
Вызывает стандартную Windows справку: вызывает WINHELPEXE с темой справки или ключевым словом, указанным для WINDOW в атрибуте HLP.

Шаблон FrameBrowseControl не имеет параметров.

## **PauseButton**

---

Шаблон PauseButton размещает кнопку в окне продвижения процесса процедур Process или Report. Когда пользователь нажимает эту кнопку, сгенерированный шаблоном код изменяет текст кнопки и приостанавливает процедуру до тех пор, пока пользователь повторно не нажмет кнопку, чтобы возобновить работу процедуры.



Шаблон `PauseButton` обеспечивает следующие параметры:

`Pause Text` (текст приостановки)

Текст, отображаемый на поверхности кнопки во время выполнения процедуры.

`Restart Text` (текст возобновления)

Текст, отображаемый на поверхности кнопки во время приостановки процедуры.

`Start Paused` (начать с приостановки)

Отметьте этот режим, чтобы первоначально приостановить процедуру так, что ее выполнение начнется только тогда, когда конечный пользователь нажимает кнопку. Выключите режим, чтобы первоначально возобновить процедуру так, что ее выполнение продолжится до завершения, если только конечный пользователь не нажмет кнопку. См. *DeferOpenReport* в главе *Менеджер Документов*.

`Start Text` (начальный текст)

Текст, отображаемый на поверхности кнопки, если процедура первоначально приостановлена.

`Allow multiple starts` (допускается перезапуск)

Отметьте этот режим, чтобы дать возможность конечному пользователю перезапускать процесс или отчет после его завершения. Это полезно для перезапуска процесса или отчета, когда фильтр и порядок сортировки задает

пользователь.

## SaveButton

Шаблон SaveButton размещает в окне кнопку ОК и дает возможность показать конечному пользователю сообщение о выполняемом действии. SaveButton управляет большинством файловых операций ввода-вывода процедуры.

## SaveButton

---

Шаблон обеспечивает следующие параметры:

### Allow (разрешается)

Отметьте любую комбинацию из трех режимов, чтобы указать разрешенные файловые операции ввода-вывода. И наоборот, выключите режим, чтобы предотвратить соответствующие действия.

### Inserts (добавления)

Генерирует код для операций добавления записей.

### Changes (изменения)

Генерирует код для операций изменения записей.

### Deletes (удаления)

Генерирует код для операций удаления записей.

### Совет:

Шаблон SaveButton не обнаруживает изменения в BLOB полях; поэтому, если изменяется только BLOB поле, шаблон SaveButton его не сохраняет. Пример школьного приложения (School) содержит решение этой проблемы.

## Field Priming on Insert (инициализация полей при добавлении)

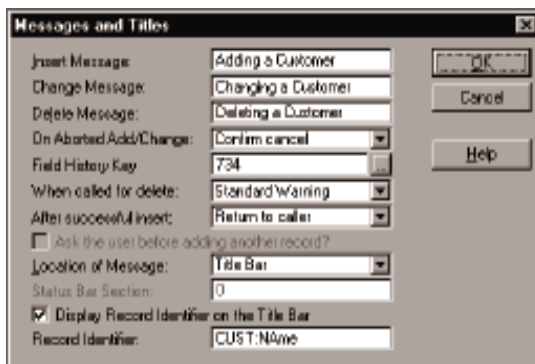
Инициализация полей позволяет Вам подготавливать стандартные значения в полях новой записи. Это значение заменяет любое, определенное в словаре данных, начальное значение. Вы можете указать поле и задать начальное значение в диалоге **Field Priming**.

## Messages and Titles (сообщения и заголовки)

Нажмите эту кнопку, чтобы открыть диалог **Messages and Titles (сообщения и заголовки)**, чтобы задать сообщения редактирования и их местоположение. Кроме того, этот диалог управляет некоторым фундаментальным поведением процедуры, как, например, требуется ли подтверждение при отмене и допускается ли повторное добавление.

### Insert Message (сообщение о добавлении)

Определяет текст для сообщения о выполняемом действии, когда процедура вызывается для добавления записи.



### Change Message (сообщение об изменении)

Определяет текст для сообщения о выполняемом действии, когда процедура вызывается для изменения записи.

### Delete Message (сообщение об удалении)

Определяет текст для сообщения о выполняемом действии, когда процедура вызывается для удаления записи.

### On Aborted Add/Change (при отмене добавления/изменения)

Определяет действия, которые следует предпринять, когда пользователь нажимает кнопку **Cancel (отмена)** при добавлении или модификации записи. Выберите из:

#### Offer to save changes (предложить сохранение изменений)

Выводит окно сообщения, предлагающее сохранить изменения вместо их отмены.

#### Confirm Cancel (подтвердить отмену)

Выводит окно сообщения, запрашивающее о том, действительно ли требуется отмена.

#### Cancel without Confirming (отмена без подтверждения)


Не выводит никаких сообщений перед выполнением отмены.

### Field History Key (клавиша предыдущего значения)

Укажите клавишу, с помощью которой восстанавливается значение из последней сохраненной записи. Когда конечный пользователь нажимает

указанную клавишу, сгенерированный код восстанавливает в имеющем фокус поле значение из предыдущей записи.

Стандартная клавиша (734) это ctrl+одиночная-кавычка ('). На большинстве американских клавиатур это - двойная-кавычка (") в нижнем регистре. На большинстве британских клавиатур это - знак @ в нижнем регистре.

Задание здесь клавиши включает также кнопку ditto  (повторить) в шаблоне FrameBrowseControl. Эта кнопка также заносит значение из последней сохраненной записи.

### **When called for Delete (при вызове для удаления)**

Укажите, что следует вывести на экран, когда процедура вызывается для удаления запись. Выберите из:

Standard Warning (стандартное сообщение)

Выводит окно сообщения, предлагающее подтвердить удаление.

Show Form (показать форму)

Выводит форму.

Automatic Delete (автоматическое удаление)

Разрешает удалять записи без вывода сообщения или подтверждения.

After successful insert (после успешного добавления)

Выберите режим поштучного или непрерывного добавления. Выберите из:

Return to caller (возврат)

После успешного добавления генерирует оператор возврата (RETURN) к вызывающей процедуре. Это приводит к режиму поштучного добавления.

**Insert another record (добавить следующую запись)**

Не генерирует оператор возврата (RETURN) к вызывающей процедуре после успешного добавления. Это приводит к режиму непрерывного добавления.

**Ask the user before adding another record (запрос перед добавлением следующей записи)**

Не генерирует автоматический возврат (RETURN) к вызывающей процедуре после успешного добавления, а запрашивает пользователя о необходимости добавления новой записи.

### **Location of Message (местоположение сообщения)**

Определяет, где выводится сообщение. Выберите из:

None/Window Control (нет/элемент окна)

Вставьте ваш собственный код для вывода сообщения в диалоговый элемент.

**Title Bar (заголовок окна)**

Показать сообщение как название окна.

Status Bar (строка состояния)

Показать сообщение в строке состояния окна. Дополнительно укажите номер

секции строки состояния в поле **Status Bar Section (секция строки состояния)**

### **Display Record Identifier on the Title Bar (вывести в заголовке идентификатор записи)**

Отметьте этот режим, чтобы добавить строку в заголовок окна. Укажите строку в поле **Record Identifier (идентификатор записи)**.

Record Identifier (идентификатор записи)

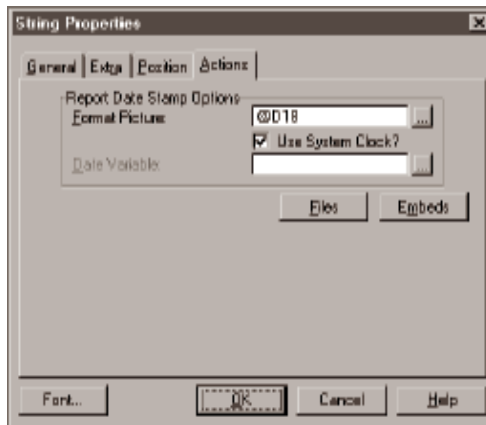
Определяет строку для добавления в заголовок окна, которую можно использовать для идентификации записи. Напечатайте строку в поле Record Identifier (идентификатор записи). Чтобы использовать имя переменной, напишите его с предшествующим восклицательным знаком (!).

## ***Диалоговые шаблоны для документов***

Шаблоны ABC содержат несколько диалоговых шаблонов, разработанных для эффективного управления некоторыми, наиболее часто повторяющимися текстами документов. Эти элементы включают отметки даты, времени и номера страницы. Этот раздел описывает шаблоны ABC для документов.

### **ReportDateStamp**

Шаблон ReportDateStamp добавляет в документ два строковых элемента: текстовая строка “Report Date:” (дата документа) и форматированная строковая переменная для вывода даты. По умолчанию шаблон ReportDateStamp отображает системную дату, используя стандартный длинный Windows формат даты (D18). Например, 18 Ноября, 1997. Тем не менее, Вы можете указать для вывода другой формат и другое значение даты.



ReportDateStamp Шаблон обеспечивает следующие параметры:

Format Picture (шаблон форматирования)

Нажмите кнопку с многоточием, чтобы выбрать формат даты. См. *Шаблоны форматирования* в *Справочнике по языку*.

Use System Clock? (использовать системные часы)

Отметьте этот режим, чтобы вывести системную дату (см. *TODAY* в *Справочнике по языку*). Выключите режим, чтобы вывести переменную, содержащую значение даты.

Date Variable (переменная даты)

Напечатайте имя переменной или нажмите кнопку с многоточием для выбора переменной в диалоге **Select Fields (выбор поля)**.

## **ReportTimeStamp**

---

Шаблон ReportTimeStamp добавляет в документ два строковых элемента: текстовая строка “ Report Date:” (время документа) и форматированная строковая переменная для вывода времени. По умолчанию шаблон ReportTimeStamp отображает системное время, используя стандартный длинный Windows формат времени (T8). Например, 12:09:22 вч. Тем не менее, Вы можете указать для вывода другой формат и другое значение времени.

ReportTimeStamp Шаблон обеспечивает следующие параметры:

Format Picture (шаблон форматирования)

Нажмите кнопку с многоточием, чтобы выбрать формат времени. См. *Шаблоны форматирования* в *Справочнике по языку*.

Use System Clock? (использовать системные часы)

Отметьте этот режим, чтобы вывести системное время (см. *CLOCK* в *Справочнике по языку*). Выключите режим, чтобы вывести переменную, содержащую значение времени.

Time Variable (переменная времени)

Напечатайте имя переменной или нажмите кнопку с многоточием для выбора переменной в диалоге **Select Fields (выбор поля)**.

## **ReportPageNumber**

---

Шаблон ReportPageNumber добавляет переменную строку (STRING) для вывода номера страницы.

Шаблон ReportPageNumber имеет никаких параметров.



## Шаблоны Кода и Распределенные шаблоны

### Шаблоны Кода

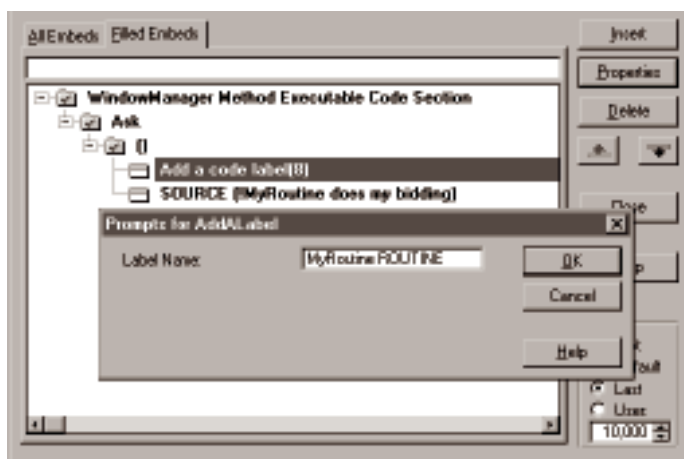
Шаблоны Кода генерируют исходный код, который будет помещен в заданную точку вставки, а иногда также и в другие. Их цель состоит в том, чтобы сделать процесс настройки процедуры более быстрым и легким. Каждый шаблон Кода имеет одну четко определенную задачу. Например, шаблон Кода Начало Процесса просто начинает новый выполняющийся процесс, и не более того. Как правило, шаблон Кода предлагает окно диалога с присутствующим на нем рядом пунктов диалога и инструкций.

Добавьте шаблоны Кода к вашей процедуре с помощью диалога **Embedded Source**. См. *Генератор Приложения – Внедренный Текст*.

В Clariion предлагаются следующие кодовые шаблоны:

#### AddALabel

Шаблон AddALabel генерирует код в первую позицию внутри точек вставки, где для кода обычно предусмотрен отступ, типа Внутренней Глобальной Карты или Секция Выполняемого Кода ABC Метода. Это позволяет Вам объявлять метки соответствия, процедуры, подпрограммы и прочие маркированные пункты всюду, где стандартами языка Clariion допускается использование меток.



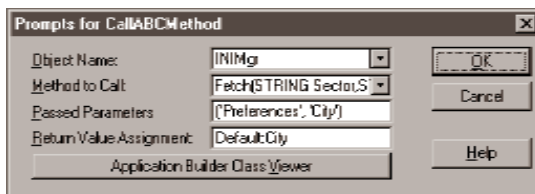
#### Имя Метки

Напечатайте код, который после генерации будет размещаться начиная с первой позиции.

## CallABCMethod

CallABCMethod шаблон генерирует код для вызова метода из ABC Библиотеки объектов. См. *Часть II – ABC Библиотека* для получения дополнительной информации об этих методах. Этот шаблон генерирует код, подобный следующему:

```
Default:City = INIMgr.Fetch('Preferences', 'City')
```



### Имя Объекта

Выберите метку объекта из списка. Список содержит все доступные для этой процедуры ABC объекты.

### Вызываемый Метод

Выберите вызываемый метод из выпадающего списка. Чтобы увидеть все параметры метода и его возвращаемые значения, прокрутите этот список горизонтально или нажмите кнопку **Обзор Класса разработки приложений (ABC)**. См. *Часть II – ABC Библиотека* для полной информации относительно этих методов, их параметров и их возвращаемых значений.

### Пропущенные Параметры

Напечатайте список пропускаемых параметров. Приложите параметры в parentheses и отделите их запятыми. Параметрами могут быть литеральные значения, выражения или имена переменных.

### Присвоение Возвращаемого Значения

Напечатайте переменную, которой будет присваиваться возвращаемое значение вызванного метода. Это поле доступно только для тех методов, которые возвращают значение.

## CallProcedureAsLookup

Шаблон CallProcedureAsLookup вызывает процедуру выбора записи. Для уведомления об успехе поиска устанавливается переменная, именуемая RequestCompleted.

### **Процедура Поиска**

Определяет вызываемую процедуру, предназначенную для выполнения поиска.

#### **Код до**

Впечатайте сюда любой исполняемый код, который выполнится до произведения поиска. Можно использовать несколько операторов, отделяя их с точкой с запятой.

#### **Код После, Закончено нормально**

Впечатайте сюда любой исполняемый код, который выполнится в случае нормального завершения поиска. Можно использовать несколько операторов, отделяя их с точкой с запятой.

#### **Код После, Отменено**

Впечатайте сюда любой исполняемый код, который выполнится в случае отмены поиска. Можно использовать несколько операторов, отделяя их с точкой с запятой.

### **CloseCurrentWindow**

---

CloseCurrentWindow шаблон просто посылает событие EVENT:CloseWindow, что приводит к нормальному завершению процедуры. Здесь отсутствуют какие-нибудь пункты, которые надо заполнить.

### **ControlValueValidation**

---

Шаблон ControlValueValidation берет значение элемента управления и ищет соответствующее ему значения в ключевом поле. Этот шаблон Кода может быть добавлен в точку вставки Accepted или Selected для таких элементов управления, как ENTRY, SPIN, LIST или COMBO. В тексте, сгенерированном с помощью этого шаблона Кода, реализован механизм проверки на соответствие значения, полученного из элемента управления, значению в ключевом поле.

Процедура поиска может быть также вызвана для предоставления конечному пользователю возможности самостоятельно выбирать значение.

#### **Ключ Поиска**

Определяет ключ поиска. Если ключ многокомпонентный, то прежде чем выполнится сгенерированный этим шаблоном код, необходимо определить значения всех остальных (непоисковых) полей ключа.

#### **Поле Поиска**

Определяет как поле, значение из которого подлежит проверке, так и поле, в

которое будет помещен результат успешного поиска. Поле Поиска должно быть компонентом Ключа Поиска.

### Процедура Поиска

Определяет вызываемую процедуру поиска.

Этот шаблон генерирует код, подобный следующему.

```

IF CUST:State OR ?CUST:State{Prop:Req} <> False
ST:StateCode = CUST:State           ! получить значение для поиска
IF Access:State.TryFetch(ST:ByCode) ! если запись не найдена
GlobalRequest = SelectRecord        ! определить действие для процедуры поиска
SelectState                         ! вызвать процедуру поиска
IF GlobalResponse = RequestCompleted ! если поиск завершен успешно
GlobalResponse = RequestCancelled   ! очистить величину, определяющую
                                     !требуемое действие
CUST:State = ST:StateCode           ! поместить найденное значение в поле
                                     !элемента управления
ELSE                                 ! ELSE (если поиск неудачен)
SELECT(?CUST:State)                 ! позиционироваться на элемент управления
CYCLE                                ! окончание обработки события
END                                  ! END (если поиск завершен успешно)
END                                  ! END (если запись не найдена)

END

```

### DisplayPopupMenu

---

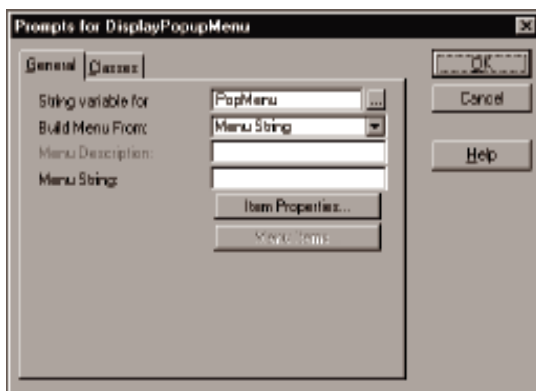
DisplayPopupMenu генерирует код, предназначенный для определения и отображения роруп-меню, и опционально, действие при сделанном конечным пользователем выборе. Можно уподобить действие некоторых пунктов Роруп-меню действию существующих на окне кнопок. Тогда текст пункта меню, соответствующий тексту на связанной с ним *кнопке*, будет доступен для выбора только в случае доступности для выбора этой *кнопки*. Когда же этот пункт меню будет выбран, то произойдет действие, аналогичное действию по нажатию *кнопки*.

Шаблон DisplayPopupMenu ссылается на Роруп-класс для выполнения его задач. См. *Роруп-класс* для получения дополнительной информации.

### String-переменная для (String variable for)

Для выбора из существующих или назначения новой переменной типа string, которая

будет носителем сделанного конечным пользователем выбора из рорир-меню, нажмите эллиптическую кнопку (...). После отображения рорир-меню в этой переменной содержится текст выбранного пункта, освобожденный от любых специальных символов. То есть переменная содержит только такие символы, как 'A-Z', 'a-z' и '0-9'. Если получившаяся величина не уникальна в пределах меню, RorirClass, чтобы сделать его уникальным, приписывает в конце его порядковый номер.



Эту переменную можно опрашивать и выполнять те или иные действия в зависимости от ее значения. Если используется способность Rorir-класса уподоблять действия пунктов меню действиям соответствующих кнопок, то это поле может быть оставлено незаполненным. См. *Свойства Пункта меню* для получения дополнительной информации об уподоблении действий.

### Строить Меню Из (Build Menu From)

Выберите, каким образом определены рорир-меню и его пункты. Предлагаемые варианты:

#### Строка Меню (Menu String)

Чтобы напечатать определение меню, используйте поле **Строка Меню**, а затем, чтобы определить поведение каждого пункта, используйте **Свойства Пункта**.

#### Список Пунктов (Item List)

Чтобы один за одним определить пункты меню, используйте кнопку **Пункты Меню**.

#### INI Файл (INI File)

Для определения той секции INI-файла, которая содержит определение меню, используется поле **Описание Меню (Menu Description)**. По умолчанию,

сгенерированный шаблоном код использует глобальный объект INIMgr, объявленный ABC шаблоном Приложения. Если не определен используемый INI-файл, объект INIMgr использует INI-файл Windows. См. *Обзор Шаблона — Панель Глобальных Свойств*.

### **Описание Меню (Menu Description)**

Напечатайте секцию INI-файла, которая содержит определение меню. См. *Рорир-класс — Сохранение и Восстановление* для получения дополнительной информации.

### **Строка Меню (Menu String)**

Напечатайте строку определения меню. В *Описании Языка* описан приемлемый синтаксис для строки определения меню (параметр *выборы* команды POPUP).

### **Свойства Пункта (Item Properties)**

Для определения свойств каждого пункта Рорир-меню нажмите эту кнопку. Пригодны только те пункты меню, которые указаны в Строке Меню. Можно уподобить действие некоторых пунктов Рорир-меню действию существующих на окне кнопок. Тогда текст пункта меню, соответствующий тексту на связанной с ним *кнопке*, будет доступен для выбора только в случае доступности для выбора этой *кнопки*. Когда же этот пункт меню будет выбран, то произойдет действие, аналогичное действию по нажатию *кнопки*. Можно также заставить пункты Рорир-меню посылать события какому-нибудь элементу управления.

### **Пункты Меню (Menu Items)**

Для определения свойств каждого пункта Рорир-меню нажмите эту кнопку. Можно уподобить действие некоторых пунктов Рорир-меню действию существующих на окне кнопок. Тогда текст пункта меню, соответствующий тексту на связанной с ним *кнопке*, будет доступен для выбора только в случае доступности для выбора этой *кнопки*. Когда же этот пункт меню будет выбран, то произойдет действие, аналогичное действию по нажатию *кнопки*. Можно также заставить пункты Рорир-меню посылать события какому-нибудь элементу управления.

### **Панель Классов**

Для переопределения установок глобального Рорир Менеджера используется панель Классов. См. *Обзор Шаблона — Опции Панели Классов — Глобальные и Локальные*.

## InitiateThread

---

При открытии MDI окна в Главном Окне Приложения (Application Frame) необходимо начать выполняемый процесс. Этот шаблон Кода предлагает простой путь инициирования процесса.



Когда процедура запускается (оператор START) в своем собственном процессе, процедура и ее окно функционируют независимо от других процессов той же самой программы; то есть конечный пользователь может по своему желанию переключать фокус между выполняемыми процессами. Иначе, они представляют собой “**немодальные**” окна.

Если новый процесс не начинается, то поведение программы зависит от того, имеет ли окно процедуры атрибут MDI. *He - MDI* дочернее окно, находящееся на той же нити, что и ее родитель, блокирует доступ к всем другим процессам в программе. Оно представляет собой окно, “**модальное по отношению приложению**”. Когда модальное к приложению окно закрывается, другие исполняемые процессы снова становятся доступны. Дочернее *MDI* окно, находящееся на той же нити, что и ее родитель, блокирует доступ только к своему родительскому окну. Когда дочернее MDI окно закрывается, его родительское окно снова получает фокус.

В диалоге **Предложения по инициированию процесса (Prompts for Initiate Thread)**, просто назовите процедуру, которая открывает MDI окно. В дополнение к этому, для нового процесса можно также изменять размер стека для размещения. По умолчанию размер стека равен 25,000 байтов.

В дополнение к этому можно добавить строчку кода, которая будет выполняться, если приложение не может открыть процесс. Впечатайте его в поле редактирования, помеченном как **Обработка Ошибки (Error Handling)**. Например,

```
MESSAGE('Процесс не может быть начат', 'Оошибка', ICON:HAND)
```

отобразит окно сообщения с надписью и иконкой ICON:HAND в том случае, если процесс не может быть начат.

Также можно добавить имя процедуры, которая будет вызываться при возникновении ошибки, путем впечатывания имени процедуры в поле **Обработка Ошибки (Error Handling)**. После этого следует добавить эту процедуру в **Дерево Приложения** с помощью команды **Вставить Процедуру**.

## LookupNonRelatedRecord

---

Шаблон LookupNonRelatedRecord используется для выполнения поиска значения, основываясь на связях, которые определены или нет в словаре данных (специальное отношение). Вы можете добавлять этот шаблон Кода в точку вставки Поиск Связанных Записей.

### Ключ Поиска (Lookup Key)

Впечатайте сюда имя ключа или нажмите эллиптическую кнопку (...) для выбора ключа из Схемы Файлов.

Ключ поиска используется для выполнения поиска в поисковом файле. Это *должен* быть уникальный ключ. Если ключ является многокомпонентным, другие поля ключа должны быть определены до выполнения сгенерированного этим шаблоном Кода.

### Поле Поиска (Lookup Field)

Впечатайте имя ключа или нажмите эллиптическую кнопку (...) для выбора поля из списка Компонентов.

Поле Поиска должно быть одним из компонентов Ключа Поиска. В пределах поискового файла оно имеет уникальное значение.

### Связанное Поле (Related Field)

Впечатайте имя ключа или нажмите эллиптическую кнопку (...) для выбора ключа из Схемы Файлов.

Связанное Поле определяет уникальное значение, используемое при поиске.

Этот шаблон генерирует код, подобный следующему:

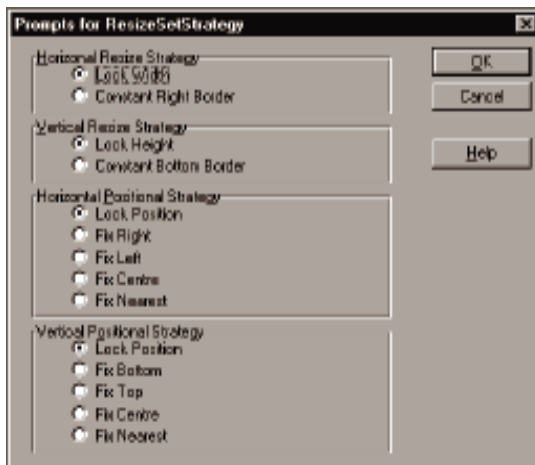
```
ST:StateCode = CUST:State           ! Получить поисковое значение  
Access:State.Fetch(ST:ByCode)      ! Получить значение из файла
```

## ResizeSetStrategy

---

Шаблон ResizeSetStrategy позволяет переопределять общую стратегию перераспределения и изменения размеров применительно к конкретному элементу управления. Предназначен исключительно для точки вставки **Определение стратегии перераспределения и изменения размеров** конкретного элемента управления. Для получения дополнительной информации о принимаемой по умолчанию стратегии смотри *Распределенные Шаблоны — Window Resize*.





Вставьте шаблон кода в точку вставки **Определение стратегии перерасмещения и изменения размеров** того элемента управления, к которому эта стратегия должна быть применена, а затем выполните последующие пункты диалога.

### **Стратегия Горизонтального Изменения размеров (Horizontal Resize Strategy)**

Определяет, каким образом определяется ширина элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Ширину (Lock Width)

Ширина элемента управления, заданная при разработке, не изменяется.

Зафиксированный Правый Край (Constant Right Border)

Фиксирует правый край, перемещает левый.

### **Стратегия Вертикального Изменения размеров (Vertical Resize Strategy)**

Определяет, каким образом определяется высота элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Высоту (Lock Height)

Высота элемента управления, заданная при разработке, не изменяется.

Зафиксированный Нижний Край (Constant Bottom Border)

Фиксирует нижний край, перемещает верхний.

### **Стратегия Горизонтального Размещения (Horizontal Positional Strategy)**

Определяет, каким образом определяется горизонтальное размещение элемента

управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Положение (Lock Position)

Левый край элемента управления сохраняет то расстояние от левого края родителя, которое было задано при разработке.

Зафиксировать правый край (Fix Right)

расстояние от правого края элемента управления до правого края родителя изменяется пропорционально.

Зафиксировать левый край (Fix Left)

расстояние от левого края элемента управления до левого края родителя изменяется пропорционально.

Зафиксировать центр (Fix Center)

расстояние от центра элемента управления до центра родителя изменяется пропорционально.

Зафиксировать ближайший край (Fix Nearest)

применяет стратегию Зафиксировать правый край (Fix Right) или Зафиксировать левый край (Fix Left) в зависимости от обстоятельств.

### **Стратегия Вертикального Размещения (Vertical Positional Strategy)**

Определяет, каким образом определяется вертикальное размещение элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Положение (Lock Position)

Верхний край элемента управления сохраняет то расстояние от верхнего края родителя, которое было задано при разработке.

Зафиксировать нижний край (Fix Bottom)

расстояние от нижнего края элемента управления до нижнего края родителя изменяется пропорционально.

Зафиксировать верхний край (Fix Top)

расстояние от верхнего края элемента управления до верхнего края родителя изменяется пропорционально.

Зафиксировать центр (Fix Center)

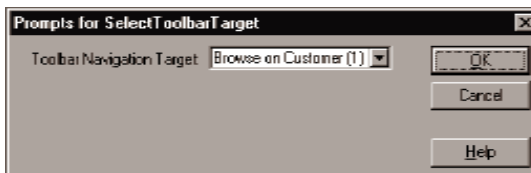
расстояние от центра элемента управления до центра родителя изменяется пропорционально.

Зафиксировать ближайший край (Fix Nearest)

применяет стратегию Зафиксировать нижний край (Fix Bottom) или Зафиксировать верхний край (Fix Top) в зависимости от обстоятельств.

## SelectToolBarTarget

Шаблон `SelectToolBarTarget` предлагает разработчикам простой способ привязывания конкретного Поля просмотра (`BrowseBox`) из данной процедуры к кнопкам навигации, расположенным на панели управления (см. *FrameBrowseControl* в главе *Шаблоны Элементов управления* и *SetTarget* в главе *Классы Панели управления*).



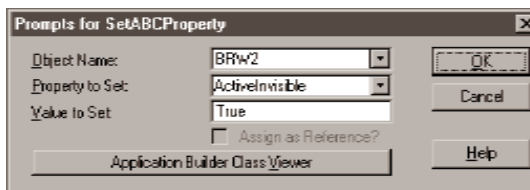
### Объект Навигации с Панели управления

Выберите то Поле просмотра (`Browsebox`), которое будет управляться с помощью кнопок навигации `FrameBrowseControl`.

## SetABCProperty

Шаблон `SetABCProperty` генерирует код для установки общественного свойства объекта ABC Библиотеки. См. *Часть II—ABC Библиотека* для получения дополнительной информации об этих свойствах. Этот шаблон генерирует код, подобный следующему:

```
BRW2.ActiveInvisible = True
```



### Имя Объекта (Object Name)

Выберите метку объекта из списка. Список содержит все ABC объекты, доступные для этой процедуры.

### Устанавливаемое Свойство (Property to Set)

Выберите устанавливаемое свойство из выпадающего списка. См. *Часть II—ABC Библиотека* для получения дополнительной информации об этих свойствах.

### Устанавливаемое Значение (Value to Set)

Напечатайте переменную, константу или соответствующее стандартам Clariion выражение, которое будет присвоено выбранному свойству.

### Присвоить как Ссылку? (Assign as Reference?)

Если взвести этот флажок, то будет генерироваться ссылочное присвоение (*Объект.Свойство &#x2190; Значение*). Если же его снять- обычное (*Объект.Свойство = Значение*). См. *Ссылочные Присвоения в Описании Языка* для получения дополнительной информации.

## SetProperty

---

Шаблон SetProperty предлагает простой способ установки исполняемого свойства любого элемента управления на окне.

### Элемент управления (Control)

Выберите из выпадающего списка метку соответствия для одного из элементов управления на окне.

### Свойство (Property)

Выберите из выпадающего списка выполняемое свойство, подлежащее установке.

### Значение (Value)

Метка переменной, константа или выражение, которое будет присвоено выбранному выполняемому свойству.

Этот шаблон генерирует код, подобный следующему:  
`?MyControl{PROP:Whatever} = значение`

## Распределенные Шаблоны

Распределенные Шаблоны добавляют функциональные возможности к процедурам, но не привязаны к конкретному элементу управления или какой-либо отдельно взятой точке вставки. Каждый распределенный шаблон имеет свою четко поставленную задачу. Например, шаблон DateTimeDisplay позволяет отображать на окне дату, время или и то, и другое.

Находясь в окне диалога **Свойств Процедуры**, распределенный шаблон можно добавить, нажав на кнопку **Расширения (Extensions)**.

Совет: С помощью кнопки «Расширения» могут быть добавлены и удалены только распределенные шаблоны. Шаблоны Элементов управления здесь могут быть только изменены, но никак *не могут* быть добавлены или удалены. Для добавления или удаления шаблонов Элементов управления используйте Конструктор Окна.

ABC шаблоны включают в себя следующие распределенные шаблоны:

## **AsciiViewInListBox**

---

Шаблон `AsciiViewInListBox` позволяет элементу управления Окно списка (LIST) чередовать отображение: то выбранный файл, то какие-нибудь другие определенные Вами наборы данных.

Шаблон `AsciiViewInListBox` добавляет те же самые функциональные возможности и те же самые приглашения, что и шаблон `AsciiViewControl`. См. *Шаблоны Элементов управления — AsciiViewControl* для получения дополнительной информации. Однако есть один дополнительный предлагаемый пункт. Поскольку предлагается распределенный шаблон, не размещающий свой собственный элемент управления, шаблон `AsciiViewInListBox` предлагает для отображения текста использовать элемент управления Окно списка (LIST):

### **Общая закладка (General Tab)**

#### **Используемое Окно списка (List box field to use)**

Выберите тот элемент управления Окно списка (LIST), в котором чередуются отображаемые данные.

#### **Инициализация объекта Viewer (Initialize Viewer)**

Определяет, когда процедура инициализирует объект Viewer. Инициализация включает в себя отбор файла во view-структуру, открытие и чтение его.

При Открытии Окна (On Open Window)

Инициализирует объект Viewer при открытии окна с тем, чтобы Окно списка (LIST) Viewer заполнялось при начальном отображении.

При Выборе Поля (On Field Selection)

Откладывает инициализацию объекта Viewer до момента выбора конечным пользователем элемента управления Окно списка (LIST).

Вручную (Manually)

Не инициализирует объект Viewer. Для его инициализации необходимо выполнить вызов подпрограммы `Viewer#.Initialize`.

#### **Просматриваемый Файл (File to Browse)**

Здесь определяется путь и имя просматриваемого файла или имя переменной, содержащей эти сведения. Имя переменной должно начинаться со знака восклицания (!).

Если не задано никакого пути, процедура ищет файл в текущей директории.

Если поле пропущено (оставлено пустым), объект Viewer предлагает конечному пользователю выбрать файл.

### **Разрешить запуск поиска из pop-up-меню (Allow pop up menu searching)**

Для добавления в pop-up-меню (появляющееся по щелчку правой кнопкой мыши) пункта, запускающего поиск файла, взведите этот флажок.

### **Разрешить запуск печати из pop-up-меню (Allow pop up menu printing)**

Для добавления в pop-up-меню (появляющееся по щелчку правой кнопкой мыши) пункта, запускающего печать всех или некоторых записей из файла, взведите этот флажок.

## **Панель классов (Classes Tab)**

Используйте панель Классов для переопределения глобальных настроек Ascii Viewer. См. *Обзор Шаблона — Опции Панели Классов — Глобальные и Локальные*.

## **DateTimeDisplay**

Шаблон DateTimeDisplay добавляет к прочим функциональным возможностям шаблона процедуры возможность отображения времени и/или даты на панели статуса или элементе управления.

Опции, появляющиеся в окне диалога Отображение Даты и Времени, разделены на две секции — Отображение Даты и Отображение Времени:

### **Отображать в окне (Display in Window)**

Взведите флажок (флажки) для добавления в окно отображения параметра (параметров).

### **Формат (Picture)**

Выберите формат отображения даты и/или времени из выпадающего списка. В списке показаны примеры отображения, например “31 октября 1959”, “5:30P.M”.

### **Другой Формат (Other Picture)**

Впечатайте выбранный Вами формат, если необходимый Вам тип формата отсутствует в списке. Смотри также: *Символы Отображения Даты* или *Символы Отображения Времени* в *Описании Языка*.

### **День Недели (Day of Week)**

Опционально показывает день недели.

## Размещение (Location)

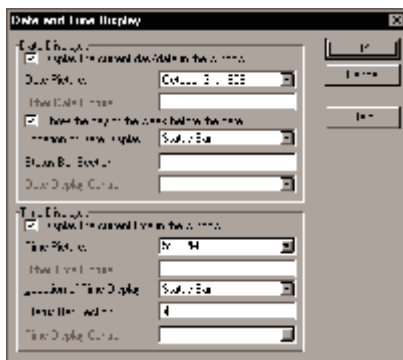
Выберите место, где будет отображаться дата и/или время: на панели статуса или на элементе управления.

## Секция Панели Статуса (Status Bar Section)

В случае отображения Даты или Времени на панели статуса необходимо определить номер секции на этой панели.

## Элемент управления, предназначенный для отображения (Display Control)

В случае отображения Даты или Времени на элементе управления необходимо определить метку соответствия этого элемента путем выбора из выпадающего списка меток всех полей окна.



## FormVCRControls

Шаблон *FormVCRControls* добавляет к процедуре Формы функциональные возможности, делаая возможной навигацию по записям с помощью *FrameBrowseControl* VCR-кнопок (кнопки, по своему виду и функциональному назначению похожие на кнопки управления видеомаягнитофоном). См. *Шаблоны Элементов управления — FrameBrowseControl* для получения дополнительной информации об этих кнопках и их функционировании.

Проще говоря, Расширение *FormVCRControls* обеспечивает «скроллинг» Формы. Вы можете показывать, добавлять, удалять или редактировать последовательно одну за одной ряд записей, не возвращаясь для выбора новой записи в вызвавшую процедуру просмотра. Однако ключи и фильтры, реализованные в вызвавшей процедуре просмотра, все же контролируют передвижение Формы. Например, Вы можете передвигаться только по тем записям, которые удовлетворяют таким условиям, как заданный предел диапазона просмотра и фильтр на отображаемые записи, и когда Вы переходите к «следующей» или «предыдущей» записи, ключ просмотра определяет последовательность, в которой появляются

записи.

Для процедур Формы, сгенерированных Мастером Приложения, если процедура Формы также содержит поле просмотра (*browsebox*), *FrameBrowseControl* кнопки управляют Формой, когда выбрана закладка “форма”, и Полею Просмотра, когда выбрана закладка “поле просмотра”. Смотри также *Шаблоны Кода—SetToolbarTarget*.

## **RecordValidation**

---

Шаблон RecordValidation добавляет функциональные возможности к Процедуре, предписывая значениям из полей элементов управления проверку на предмет соответствия данным, определенным в словаре. Также он допускает определение элемента управления, значения из которого не подлежат проверке.

### **Проверять при принятии элемента управления (Validate when the control is Accepted)**

Определяет, что проверка пригодности значения происходит в момент возникновения на элементе управления события EVENT:Accepted. Это событие возникает в тех случаях, когда конечный пользователь заканчивает обработку поля или перемещает с него фокус ввода.

### **Проверять во время Безостановочного Выбора (Validate during NonStop Select)**

Определяет, что проверка пригодности значения происходит при изменении значения любого элемента управления, если только окно находится в AcceptAll (Безостановочном) режиме и владеет фокусом.

### **Не проверять (Do Not Validate)**

Открывает окно диалога «Не Проверять», который позволяет выбирать поля из выпадающего списка. Выбранные Вами поля будут исключены из списка подлежащих проверке пригодности.

## **WindowResize**

---

Шаблон WindowResize позволяет конечному пользователю изменять размеры окна, которые изначально устанавливаются необходимой величины для размещения находящихся на нем элементов управления (Окна списка, элементы управления вводом, кнопки и т.д.).

Шаблон производит код, предназначенный для изменения местоположения элементов управления, их размеров, или и того и другого в случае изменения конечным пользователем размеров окна.



Совет: Чтобы разрешить изменение размеров окна необходимо определить тип рамки окна как Изменяющая размеры (Resizable). См. *Конструктор Окна— Диалог Свойств Окна* для получения дополнительной информации об этих установках.

### **Стратегия изменения размеров и местоположения (Resize Strategy)**

Определяет метод, используемый при изменении размеров и местоположения элемента управления с целью соответствия новым размерам окна. Методы могут быть следующие:

#### **Изменение размеров (Resize)**

Масштабирует все оконные координаты на одну и ту же величину, сохраняя таким образом относительные размеры и расположение всех элементов управления. То есть все элементы управления, такие как кнопки и поля ввода, по мере увеличения высоты и ширины окна сами становятся выше и шире. При этом шрифты на окне остаются неизменными.

#### **Распространение (Spread)**

Осуществляет изменение размеров окна таким образом, чтобы сохранялся первоначальный его вид (заданный в процессе разработки), что достигается путем применения к каждому элементу управления своей стратегии изменения местоположения и размеров. Например, размеры кнопки не изменяются, но ее положение привязано к ближайшему краю окна. Или, напротив, размеры окна списка и его положение масштабируются пропорционально окну.

#### **Поверхность (Surface)**

Размещает элементы управления (окно списка, лист, панель, изображение) на окне таким образом, чтобы максимизировать их размеры (т.е. использовать доступное количество пикселей на окне по максимуму). Эта стратегия рекомендуется для всех окон, сгенерированных с помощью Мастера.

Совет: Несмотря на то, что окна списка могут изменять свои размеры, ширина колонок в них остается неизменной. Тем не менее самая правая колонка расширяется или сжимается в зависимости от наличия свободного места.

### **Не изменять размеры элементов управления (Don't Alter Controls)**

Средство управления не меняет своих размеров при изменении размеров окна.

Совет: В случае применения этой стратегии каждому элементу управления можно добавить атрибут SCROLL и поставить атрибут HVSCROLL на окно, что позволит

использовать ‘окно, перемещающееся по листу большего размера’.

### **Ограничить Минимальный Размер Окна (Restrict Minimum Window Size)**

В случае необходимости определения минимальной высоты и ширины окна взведите этот флажок. Это позволит назначить окну минимальный необходимый размер в зависимости от количества и размеров элементов управления на окне. Другими словами, Вы можете обезопасить конечного пользователя от такого сокращения размеров окна, при котором его элементы управления станут невидимыми или неузнаваемыми.

#### **Минимальная Ширина (Minimum Width)**

Определяет минимальную ширину окна в диалоговых единицах. Диалоговые Единицы определяются размером шрифта окна и составляют  $1/4$  средней ширины символа. Нулевое значение приводит к установлению минимального размера окна, равного размеру окна при открытии (не обязательно заданные во время разработки размеры). Другими словами, принимаются во внимание любые установки INI-файла плюс любые установки Свойств во время выполнения. Таким образом, разработчику разрешается открыть окно, выполнить любые динамические преобразования элементов управления (включая изменение размеров окна), прежде чем вступит в силу ограничение минимальных размеров.

#### **Минимальная Высота (Minimum Height)**

Определяет минимальную высоту окна в диалоговых единицах. Диалоговые Единицы определяются размером шрифта окна и составляют  $1/8$  средней высоты символа. Нулевое значение приводит к установлению минимального размера окна, равного размеру окна при открытии (не обязательно заданные во время разработки размеры). Другими словами, принимаются во внимание любые установки INI-файла плюс любые установки Свойств во время выполнения. Таким образом, разработчику разрешается открыть окно, выполнить любые динамические преобразования элементов управления (включая изменение размеров окна), прежде чем вступит в силу ограничение минимальных размеров.

### **Ограничить Максимальный Размер Окна (Restrict Maximum Window Size)**

В случае необходимости определения максимальной высоты и ширины окна взведите этот флажок. Это позволит задать максимальные необходимые размеры окна.

### **Максимальная Ширина (Maximum Width)**

Определяет максимальную ширину окна в диалоговых единицах. Диалоговые Единицы определяются размером шрифта окна и составляют  $1/4$  средней ширины символа. Нулевое значение приводит к установлению максимального размера окна, равного размеру окна при открытии (не обязательно заданные во время разработки размеры). Другими словами, принимаются во внимание любые установки INI-файла плюс любые установки Свойств во время выполнения. Таким образом, разработчику разрешается открыть окно, выполнить любые динамические преобразования элементов управления (включая изменение размеров окна), прежде чем вступит в силу ограничение максимальных размеров.

### **Максимальная Высота (Maximum Height)**

Определяет максимальную высоту окна в диалоговых единицах. Диалоговые Единицы определяются размером шрифта окна и составляют  $1/8$  средней высоты символа. Нулевое значение приводит к установлению максимального размера окна, равного размеру окна при открытии (не обязательно заданные во время разработки размеры). Другими словами, принимаются во внимание любые установки INI-файла плюс любые установки Свойств во время выполнения. Таким образом, разработчику разрешается открыть окно, выполнить любые динамические преобразования элементов управления (включая изменение размеров окна), прежде чем вступит в силу ограничение максимальных размеров.

### Переопределить стратегии элемента управления (Override Control Strategies)

Для переопределения заданной по умолчанию стратегии изменения размеров у конкретного элемента управления нажмите эту кнопку. Это открывает диалог **Переопределить стратегии элемента управления**.

#### Переопределить стратегии элемента управления

Диалог **Переопределить стратегии элемента управления (Override Control Strategies)** позволяет переопределять у конкретного элемента управления заданную по умолчанию стратегию изменения размеров. Например, по умолчанию кнопки «привязаны» («fixed») к ближайшим краям окна и не меняют своего положения подобно большинству других элементов управления на окне. Однако, если необходимо, чтобы расположение какой-то кнопки в процедуре изменялось подобно другим элементам управления, определите здесь необходимые параметры. Смотри также *Класс изменения размеров окна (WindowResizeClass) — SetStrategy*.

Нажмите кнопку **Вставить (Insert)** для выбора элемента управления, у которого следует установить стратегию изменения размеров. После этого выберите необходимый Вам вариант изменения размера и расположения из следующего списка:

### **Стратегия Горизонтального Изменения размеров (Horizontal Resize Strategy)**

Определяет, каким образом определяется ширина элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Ширину (Lock Width)

Ширина элемента управления, заданная при разработке, не изменяется.

Зафиксированный Правый Край (Constant Right Border)

Фиксирует правый край, перемещает левый.

### **Стратегия Вертикального Изменения размеров (Vertical Resize Strategy)**

Определяет, каким образом определяется высота элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Высоту (Lock Height)

Высота элемента управления, заданная при разработке, не изменяется.

Зафиксированный Нижний Край (Constant Bottom Border)

Фиксирует нижний край, перемещает верхний.

### **Стратегия Горизонтального Размещения (Horizontal Positional Strategy)**

Определяет, каким образом определяется горизонтальное размещение элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Положение (Lock Position)

Левый край элемента управления сохраняет то расстояние от левого края родителя, которое было задано при разработке.

Зафиксировать правый край (Fix Right)

расстояние от правого края элемента управления до правого края родителя изменяется пропорционально.

Зафиксировать левый край (Fix Left)

расстояние от левого края элемента управления до левого края родителя изменяется пропорционально.

Зафиксировать центр (Fix Center)

расстояние от центра элемента управления до центра родителя изменяется пропорционально.

Зафиксировать ближайший край (Fix Nearest)

применяет стратегию Зафиксировать правый край (Fix Right) или Зафиксировать левый край (Fix Left) в зависимости от обстоятельств.

### **Стратегия Вертикального Размещения (Vertical Positional Strategy)**

Определяет, каким образом определяется вертикальное размещение элемента управления при изменении конечным пользователем размеров окна. Выбирается из следующего списка:

Зафиксировать Положение (Lock Position)

Верхний край элемента управления сохраняет то расстояние от верхнего края родителя, которое было задано при разработке.

Зафиксировать нижний край (Fix Bottom)

расстояние от нижнего края элемента управления до нижнего края родителя изменяется пропорционально.

Зафиксировать верхний край (Fix Top)

расстояние от верхнего края элемента управления до верхнего края родителя изменяется пропорционально.

Зафиксировать центр (Fix Center)

расстояние от центра элемента управления до центра родителя изменяется пропорционально.

Зафиксировать ближайший край (Fix Nearest)

применяет стратегию Зафиксировать нижний край (Fix Bottom) или Зафиксировать верхний край (Fix Top) в зависимости от обстоятельств.

### **Опции Конфигурирования Resizer**

#### **Автоматически определять родителя элемента управления (Automatically find parent controls)**

Взведите этот флажок для автоматической установки у элементов управления на окне отношений родитель/ребенок. Снятие этого флажка приводит к назначению родителем всех оконных элементов управления самого окна. Установка отношений родитель/ребенок делает возможным каскадное масштабирование элементов управления от родителя к ребенку. См. *Методы WindowResizeClass* — *SetParentDefaults* для получения дополнительной информации.

#### **Оптимизация перемещения (Optimize Moves)**

Если взвести этот флажок, то все элементы управления на окне при изменении размеров окна будут перемещены все сразу, что приводит к более четкому изменению размеров, а на некоторых окнах также

позволяет избежать появления ошибок. См. *Свойства WindowResizeClass — DeferMoves* для получения дополнительной информации.

### **Оптимизация перерисовки (Optimize Redraws)**

Если взвести этот флажок, то элемент управления на время выполнения операции изменения размеров окна становится прозрачным (приобретает атрибут TRN). Это обеспечивает более плавную перерисовку окна, а на некоторых окнах также позволяет избежать появления ошибок. См. *Свойства WindowResizeClass — AutoTransparent* для получения дополнительной информации.

### **Закладка Классов**

Чтобы переопределить глобальные установки Resizer, используйте закладку Классов. См. *Обзор Шаблона — Опции Закладки Классов — Глобальные и Локальные*.