SOFTARIA
serious things for serious people

Разработка приложений для Lotus Domino/Notes R5. Краткий курс, общая концепция.

Под девизом: ПУСТЬ ЛОТУСИСТОВ СТАНЕТ БОЛЬШЕ!

1. Что такое lotus Notes и с чем его едят

Вообще, в инете по запросу на google.com «Что такое lotus» можно найти немного информации на эту тему, но, рискуя повториться, я все-таки опишу общую концепцию и те ошибки в подходах, которые автор сам (сама) допустил при переходе к этой технологии с реляционных баз данных.

Итак, Lotus….

Лотус это клиент-серверное решение. То есть наличествует сервер, который хранит информацию и авторизует пользователей, выполняет серверную логику приложений, есть клиенты, которые выполняют бизнес логику серверных приложений.

Основная парадигма Лотус – это документарный подход. Если говорить о современных подходах в реляционной технологии, то какой-то аналогией является хранение данных в виде XML, который описывает документ в виде набора «поле=значение» и хранится в одном поле таблицы.
Лотус работает не с реляционными данными, его основной объект – это документ. Представьте себе документ на бумажке – где тут первичные ключи? Где наперед заданная структура? А нету! Вот что хочу – то и пишу!

И Лотус позволяет эффективно работать с такими данными. Отсюда следует ряд ограничений – ну что поделать – «всякая палка о двух концах» - а именно – Лотус неудобен для построения отчетов. На Лотус ОЧЕНЬ сложно и долго построить многомерный параметризуемый отчет. Я не говорю, что нельзя, строили экспортом в Эксель, но все-таки это называется автогеном через задний проход. Тем не менее, задача классическая и существуют классические способы ее решения через связку lotus-RDBMS, где Лотус используется в качестве системы сбора и обработки информации, а RDB storage – в качестве системы построения отчетов.

В Лотус отсутствует понятие транзакции. То есть, если вам необходимо выполнить логическую совокупность действий как одно целое, и в случае падения операции в середине – все откатить, как было – то у вас это стандартными методами не получится.
Автор слышал (а), что в R7 планируется подложить реляционный бакэнд в виде DB2, но… у меня нет точной информации на эту тему.
В силу этой причины на Лотус не рекомендуется строить финансовые системы, которые рассчитаны не на учет, а именно на перемещение средств, так как возможна ситуация, когда при переводе денег из А в Б, деньги из А ушли, сервер упал, и деньги в Б не поступили. Это замечание не касается систем, в которых просто учитываются финансовые операции.

Слабые стороны Lotus, по сути, описаны. Теперь перейдем к достоинствам:

Итак, репликация и накат дизайна.
Репликация – это синхронизация данных между копиями одного и того же приложения. Лучше на примере: Есть компания, у которой куча филиалов в разных городах. В компании функционирует система, в которую вбиваются и в которой согласуются финансовые проводки. Есть центральный офис, в который эти проводки отовсюду поступают и в котором они либо подтверждаются и отправляются в филиалы на выполнение, либо отклоняются.

Как решается задача территориальной распределенности в Lotus?
В каждом филиале ставится сервер domino, на котором в каждом филиале ставится реплика базы. Пользователи каждого филиала работают с базой на своем сервере. А сервера по расписанию, например раз в час, обмениваются всеми поступившими изменениями и новыми документами. При этом параметры репликации можно задать таким образом, что множество документов поступающих с сервера на сервер будет ограничено – например, в центральный офис будут поступать документы со всех филиалов, а в филиале будут присутствовать только его документы.

При такой схеме, в Иркутске бухгалтер вносит планируемую проводку, в состоянии черновик и отправляет ее на рассмотрение, через час она появляется в главном офисе, там он утверждается или отклоняется и информация об этом изменении еще через час достигает сервера в Иркутске. Можно и не раз в час реплицировать обновления, при кластерной репликации обновления распространяются практически мгновенно.

Дизайн базы – слой бизнес логики – тоже хранится в виде специфичных документов. И все обновления дизайна также реплицируются между серверами. В частности эта парадигма сильно облегчает задачу перехода к новым версиям. База с новой версией дизайна просто объявляется источником дизайна, а база со старой версией дизайна – наследником. После чего стандартными средствами запускается накат дизайна, и все обновления поступают в работающее приложение.

А еще клиент Lotus Notes, по сути, является облегченной версией сервера. Поэтому при отсутствии постоянной связи с сервером я могу сделать себе локальную реплику базы, работать в ней и отреплицировать ее с сервером при появлении связи. Это дает широкие возможности для создания единой

информационной системы для компаний, специфика деятельности которых подразумевает частые отъезды в командировки сотрудников. Так как сотрудники, уезжая, берут с собой ноутбук и на нем постоянно фиксируют все необходимые изменения, которые после приезда сразу поступают в информационную систему компании и становятся доступны всем заинтересованным лицам. И не надо париться – отчеты по командировкам писать или кипы бумаг привезенных просматривать.

Следующее достоинство lotus – система защиты информации.
При регистрации пользователя, сервер создает для него ИД файл, в котором хранятся публичный и приватный ключи, а также электронная подпись этого пользователя. Для обращения к серверу пользователь должен авторизоваться, используя этот ИД файл. Все изменения, которые пользователь вносит в документы, подписываются им. Будучи каким угодно крутым администратором – я все равно не смогу эмулировать, что документ был сохранен от имени конкретного пользователя. Кроме того, Lotus позволяет сохранять электронные подписи всех когда-либо сохранявших документ пользователей. Правда, для этого требуются усилия дизайнера бизнес логики конкретного приложения.
Весь трафик между серверами и клиент-сервером шифруется. Локальные реплики баз могут быть зашифрованы.
Детально об устройстве схемы раздачи прав читайте в соответствующем разделе этого документа.

Далее, Lotus тесно интегрирован с почтовой системой. Domino сам является SMTP сервером, который дополнен рядом возможностей для работы с внутренней корреспонденцией. В частности любой документ внутри Lotus может быть отправлен по почте, как в виде письма, так и в виде той формы, по которой он был создан. При регистрации пользователя, по умолчанию для него создается почтовый ящик на сервере Domino.

Domino также является HTTP сервером. Согласно бизнес логике приложения, сервер генерирует HTML для данных, которые на нем хранятся. Поэтому для большинства приложений возможна реализация как интерфейса в Lotus клиенте, так и через web-браузер. Есть, однако, ряд ограничений связанных с интеграцией Lotus-а с внешними объектами, например сохранение OLE объектов внутри Lotus документа невозможно реализовать в любом web-браузере.

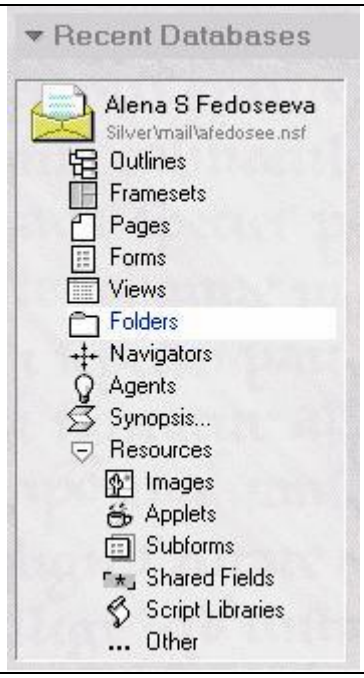
Lotus является интегрирующей платформой – то есть позволяет организовать обмен данными с множеством различных систем, функционирующих на базе другой платформы, он поддерживает связку с OLE объектами, что позволяет хранить в Lotus документах объекты MS Office и пользоваться всей их функциональностью. Так же стандартной является связка Lotus- RDBMS (то есть с различными реляционными базами данных), существует целый ряд стандартных продуктов для реализации этой связки, а также поддерживается ODBC интерфейс для обращения к данным в RDBMS из lotus script & formula language.

Lotus также является средой разработки, которая позволяет разрабатывать приложения с различной бизнес логикой.

2. Элементы дизайна базы – общая концепция

База содержит документы. Документами является как информация, так и бизнес логика приложения. Программируемые элементы бизнес логики называются в контексте этого документа дизайном базы. Дизайн состоит из различных элементов. В этом документе будет освещено большинство их, но не все. Основными элементами дизайна являются:

1. Формы
2. Подформы
3. Страницы
4. Представления
5. Папки
6. Агенты
7. Фрэймсеты
8. Схемы навигации
9. Библиотеки кнопок
10. Библиотеки полей
11. Библиотеки скриптов
12. Библиотеки картинок
13. События базы



2.1. Документы и Формы

Документ в базе является некоторой логической сущностью, имеющей ряд обязательных атрибутов и произвольный список полей.
Обязательными атрибутами являются, например UniversalID документа – это 32 символьная строка, являющаяся уникальным идентификатором документа в базе и всех репликах базы на других серверах, noteID – строка с 8-символьным идентификатором документа в базе, автор документа, дата создания и последней модификации и т.п.
Документ может содержать произвольное количество полей (за точными цифрами, ограничивающими кол-во полей сверху, обращайтесь к официальному мануалу). Поле документа - это его атрибут с именем и значением. Существуют правила конструирования имен полей, чего в них не должно быть и с чего они не должны начинаться, но если вы напишите строку длинной символов до 20, состоящую из букв и цифр, и начинающуюся с буквы, то это будет вполне нормальное имя для поля. Имена полей не являются case-sensitive. Поле документа может содержать числовые значения, текст, даты, бинарные данные и т.д.

Теперь о формах. Форма это тот формат, в котором вы хотите отображать документы. То есть вы создаете форму, компонуете на ней поля, в которые при открытии документа будет подставляться содержимое одноименных полей документа, задаете логику отображения и заполнения этих полей и т.д. Подробно о дизайне форм будет сказано далее.

Пример:
Есть физический документ, в нем есть поле Office, в котором содержится название филиала, к которому этот документ относится. В конкретном документе А поле office=”Новосибирский филиал”. При открытии этого документа формой Б, на которой есть поле office, в месте расположения этого поля пользователь увидит строку «Новосибирский филиал».
И обратно, при создании нового документа по форме Б и вписании в поле Office формы Б некоторого значения – на физическом документе после сохранения будет создано поле office со значением указанным при создании документа по форме.

То есть форма это некоторый трафарет, который накладывается на документ и задает логику отображения и редактирования документа. Документ можно открыть любой формой. В поле документа FORM содержится имя формы, которой его последний раз редактировали. Здесь у меня есть сомнения – может быть в этом поле (без принудительного перепрописывания) содержится имя формы, которой его создали, а при редактировании другой формой - ее имя не прописывается в это поле.

2.2. Подформы

Подформы - это такие формы, которые содержат логику отображения и редактирования повторяющихся во многих формах блоков. Например, блок ввода и отображения контактной информации, он может входить в

форму персоны, отдела, офиса и т.д. Все, что далее будет написано про дизайн форм и программирование бизнес логики для форм – имеет место и для подформ.

2.3. Страницы

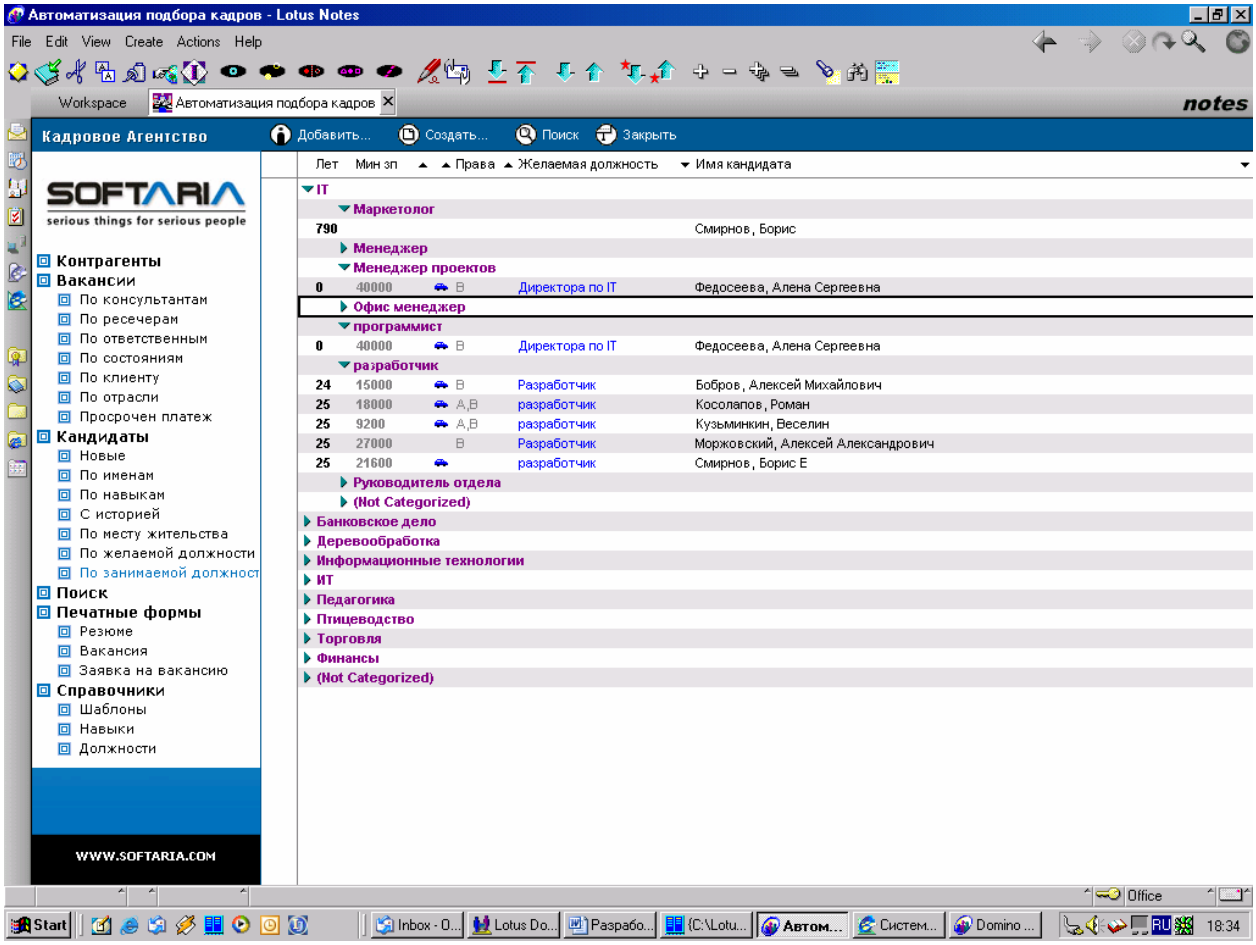
Страницы – элементы дизайна, используемые для ОТОБРАЖЕНИЯ информации, но не ввода, как следствие есть ограничения по сравнению с формами – на страницах нет полей, все остальные элементы свойственные формам (таблицы, секции, кнопки, встроенные объекты, ссылки и прочее) на них есть.

2.4. Представления

Представления - это способы отображения документов для пользователя. В представлении задается:

1. какое множество документов отображать (множество документов задается языком формул как некоторый предикат на множестве всех документов, например select form="employee" & sex="male" – выберет все документы, у которых в поле form содержится "employee" и в поле sex – "male")
2. Какие поля документов отображать в списке документов, каким образом сортировать и т.д.

Пример представления:



2.5. Папки

Папки – по сути – те же представления, отличаются тем, что не имеют формулы выбора множества документов. Документы, которые будут лежать в той или иной папке, задаются самим пользователем или в бизнес логике базы – из кода привязывая - к какой папке будет относиться тот или иной документ при возникновении какого-либо события. Один документ может лежать в произвольном количестве папок одновременно. Физически это один и тот же документ, просто в документе в специальном атрибуте **FolderReferences** прописывается – к каким папкам он относится. Этот атрибут доступен на чтение и редактирование из Lotus script, например, Folders=NotesDocument.FolderReferences.

2.6. Агенты

Агенты это куски кода, реализующие некоторую бизнес логику. Агенты могут вызываться при нажатии каких-либо кнопок или ссылок, при возникновении каких-либо событий, сервером по расписанию.

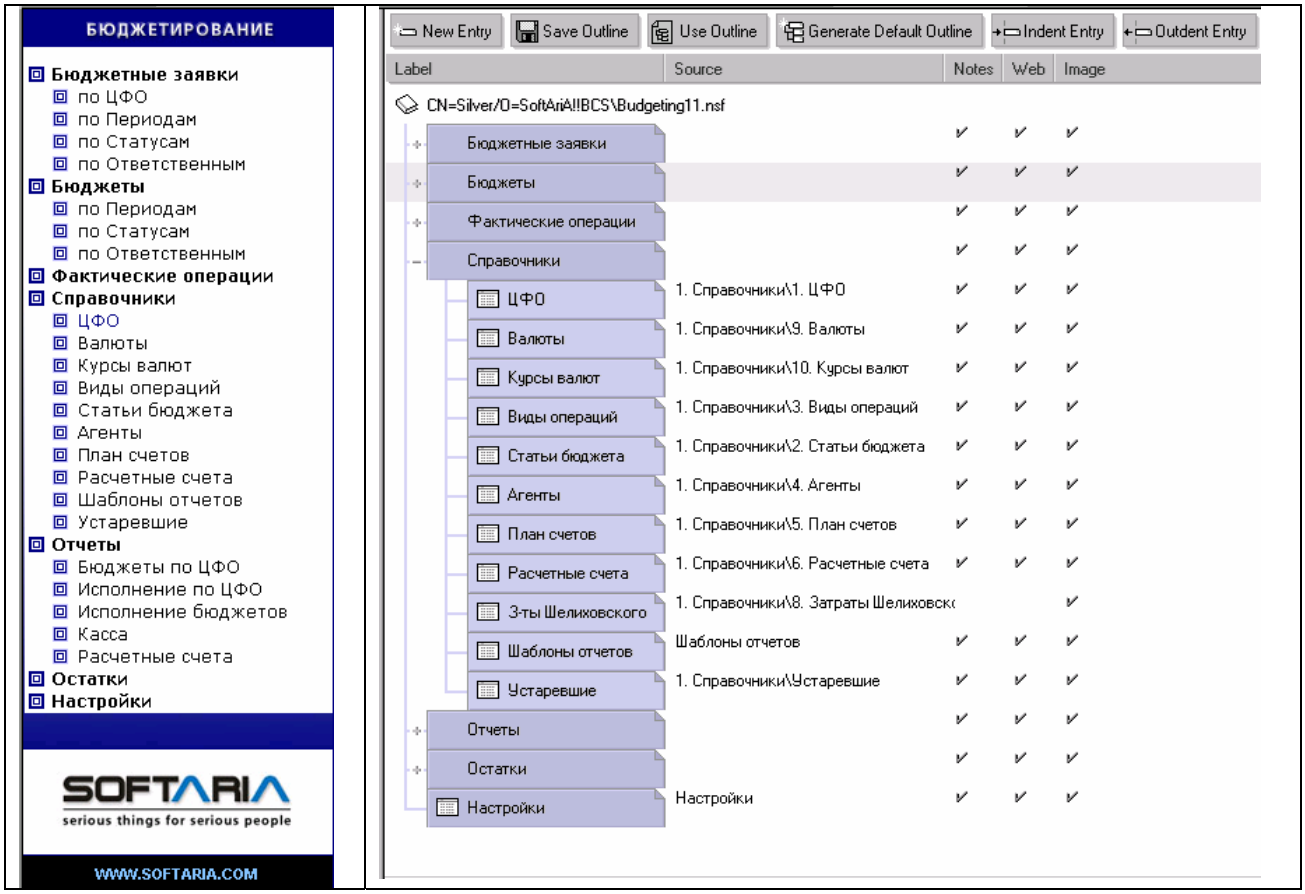
2.7. Фрэймсеты

Фрэймсеты это сущности, задающие логику расположения окошек, в которых отображаются навигация, заголовки, представления, формы и прочие элементы интерфейса пользователя.

2.8. Схемы навигации

Схема навигации – так автор назвал ту сущность, которая в англоязычном дизайнере называется Outline. Так как автор не пользуется русскоязычным дизайнером, то он (она) слагает с себя ответственность за расхождение в наименованиях, используемых в документе и в классическом переводе от IBM - названий элементов дизайна.

Итак, схема навигации – это сущность, на которой можно задать расположение ссылок на представления и прочую бизнес-логику в удобном для пользователя, логически стройном варианте. Пример схемы – в дизайнере и клиенте:



Далее схема вставляется в страницу, на странице для вставленной схемы задаются шрифты и стили, страница вставляется во фрэйм фрэймсета, а фрэймсет прописывается в свойствах базы, как загружаемый при открытии в Notes клиенте.

2.9. Библиотеки кнопок

Кнопки могут быть атрибутами конкретной формы, страницы, представления или папки, а могут входить в библиотеку кнопок и использоваться в ряде элементов дизайна, путем помещения ссылки на кнопку из библиотеки. Библиотеки кнопок (shared actions) можно редактировать в дизайнере в закладке Other (см. картинку 1)

2.10. Библиотеки полей

На формах используются поля. Одно и то же поле может присутствовать на ряде форм и иметь на всех формах одинаковые атрибуты (подробно атрибуты будут описаны далее). Такое поле лучше не тиражировать, а сделать его библиотечным и включать в формы по мере необходимости.

2.11. Библиотеки скриптов

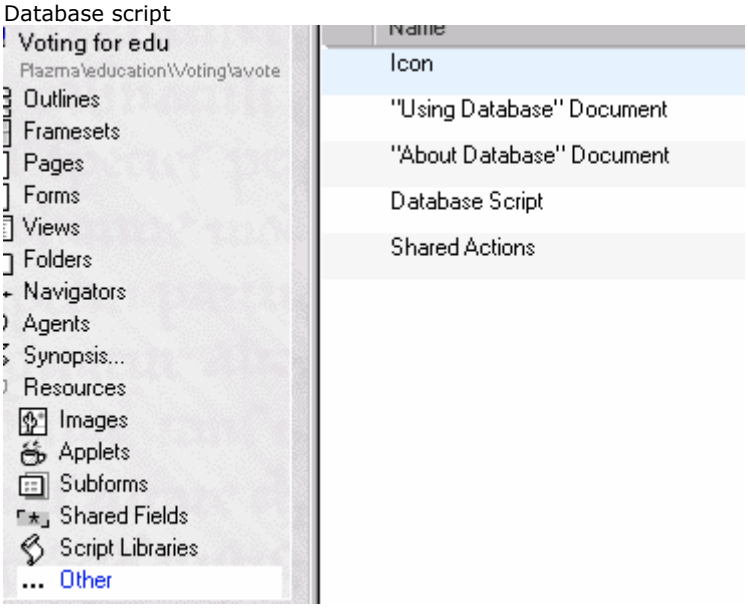
Здесь описываются элементы кода на Lotus script или Java, объединенные в виде функций и подпрограмм или объектов с методами и которые потом могут вызываться из агентов, кнопок, событий форм, представлений, страниц, подформ, папок, фрэймсетов, всех мест, где возможно использование программируемой бизнес логики. Для того чтобы вызвать библиотеку используется Use "<Имя библиотеки>", все объекты библиотеки далее вызываются просто по имени. Тут, конечно, надо быть осторожным, чтобы не породить конфликт имен или циклические вызовы библиотеки из библиотеки.

2.12. Библиотеки картинок

Вся графика, используемая на кнопках, в формах и прочих элементах дизайна в случае использования более чем в 1 месте, должна быть собрана в библиотеке картинок, для избежания хранения тяжеловесной графики в более чем одном месте базы.

2.13. События базы

Базы Lotus являются событийно-ориентированными. Есть события открытия и закрытия форм, страниц и представлений, прихода почты и т.д. У самой базы тоже есть события. Когда они происходят – на факт их свершения можно навесить бизнес логику. Ее можно написать на Lotus script или Formula language. Эти события можно запрограммировать в Database script, доступ к которому осуществляется из пункта Other -> Database script



События бывают следующие:

Sub Postopen(Source As Notesuidatabase)

End Sub

- событие, которое происходит сразу после открытия базы пользователем. В качестве параметра передается текущая база, открытая на рабочем пространстве. Здесь удобно инициализировать или считывать значения переменных окружения, проверять версию клиента и пр. Здесь же некоторые программисты помещают контроль количества уже работающих с базой пользователей и в этом месте его легко сломать раз и навсегда в приложениях, имеющих ограничение на кол-во пользовательских мест.

Sub Queryclose(Source As Notesuidatabase, Continue As Variant)

End Sub

- событие, которое происходит прямо, перед тем как база закрывается. Здесь можно, например, запомнить в переменные окружения последнее открытое пользователем представление, чтобы при следующем открытии базы открыть его же. Везде и всегда на всех событиях с префиксом Query параметр Continue обозначает - продолжать ли выполнять операцию, в результате которой возникло событие. То есть в данном случае –

продолжать ли закрытие базы. Если я поставлю в событии Continue=False – то база никогда не закроется. Придется убивать клиента ;))

Sub Querydocumentdelete(Source As Notesuidatabase, Continue As Variant)

End Sub

- событие прямо перед удалением документов. Здесь можно в случае удаления одного документа из кипы логически связанных – удалить все присоединенные к нему.

Sub Postdocumentdelete(Source As Notesuidatabase)

End Sub

- это понятно уже – сразу после удаления документов. Когда вы нажали на документе del – возникнет 1 событие, связанное с удалением. Потом при нажатии F9 – обновление представления, произойдет это событие.

Надо сказать, что при изменении событий базы они вступают в силу только после полного переоткрытия базы – в клиенте и дизайнере. То есть если вы поменяли событие, закрыли базу в клиенте, а потом снова открыли, то у вас событие сработает все равно старое. Пока вы ее полностью не закроете и не откроете.

2.14. Фенечки

В категории Other есть также документы About database & Using Database – эти документы посвящены описанию бизнес логики базы, то есть, по сути, это место для пользовательской документации. Пользователю она доступна в меню Help, когда он открывает базу.

3. Система раздачи прав

ЭТО САМЫЙ ВАЖНЫЙ РАЗДЕЛ. Его надо наизусть выучить!
Итак, права в Notes раздаются на уровне:

1. Сервера
2. Базы данных
3. Документов
4. Элементов дизайна и бизнес логики

Права считаются на сервере. При работе в локальной базе клиент не определяет маску прав текущего пользователя. Однако если вы локально произведете изменение, на которое вы не имеете права на сервере, то отреплицировать его на сервер вы не сможете. Что такое репликация я тут не рассказываю – понятие базовое.

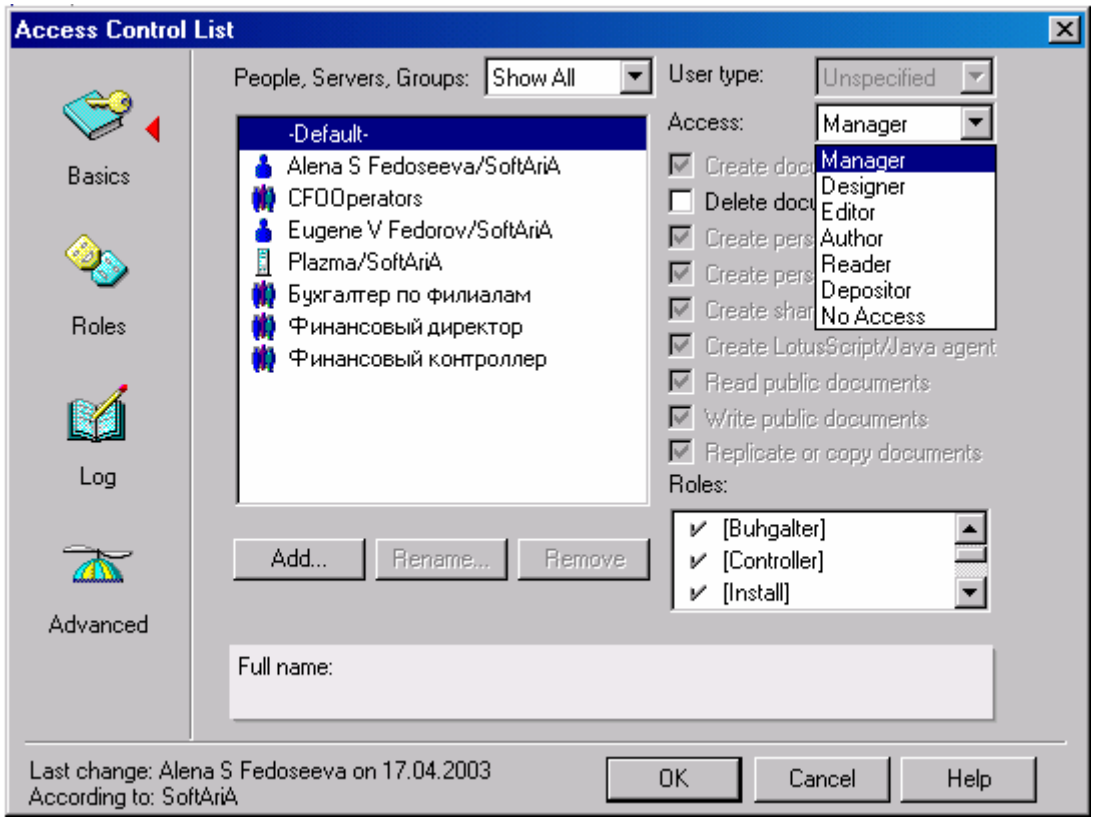
Итак, уровни раздачи прав:

3.1. Сервер

При установке сервера - в серверном документе в закладке Security указывается, кто имеет доступ к серверу, кто не имеет, кто может создавать на нем новые базы, реплики баз, запускать агентов (персональных, обычных и с внешними вызовами). Если я ошибаюсь или даю не всю информацию о настройке серверного доступа – пусть меня поправят админы, я сама не админ. Итак, создавать на сервере базы или пускать агентов, да и вообще иметь доступ к серверу может только тот, кто указан ТАМ.

3.2. База данных

В базе данных есть ACL – Access Control List. Чтобы его открыть – встаньте на базе и в меню File->Database-Access Control. Откроется окно вот такого вида:



Итак, в этом окне перечислены пользователи и группы пользователей из адресной книги Lotus Notes, справа верху указан уровень доступа к базе выделенного пользователя или группы, в низу справа – список логических ролей в базе этой группы или пользователя. Для каждой группы или пользователя к уровню доступа добавляется также набор опций разрешающих различные действия над базой.

3.2.1. Уровни доступа

1. No Access – полное отсутствие какого либо доступа к базе. Однако есть исключения, связанные с правами read/write public documents, о них будет сказано в описании опций.
2. Depositor – позволяет пользователю создавать документы в базе (но эти документы после сохранения пользователь не видит)
3. Reader – позволяет читать документы в базе, но не создавать или редактировать
4. Author – позволяет создавать документы в базе при включенной опции Create documents, читать и редактировать документы согласно логике данных логических ролей на доступ к документам и бизнес логике.
5. Editor – редактирую все документы доступные на чтение, включает в себя права доступа на уровне Author
6. Designer – являюсь editor-ом относительно документов в базе + могу модифицировать дизайн базы
7. Manager – могу делать все, то есть являюсь дизайнером + могу изменять Access Control List базы, настройки параметров репликации, могу удалить базу , в общем сделать все что придет в голову.

3.2.2. Опции

У каждого уровня доступа есть свой набор разрешенных опций. Опции Read/Write public documents можно дать при любом уровне доступа, и они дадут даже пользователям с уровнем доступа No Access или Depositor права на чтение и создание публичных документов. Публичные документы это документы созданные по форме с флагом available to public access users и имеющие поле $PublicAccess=”1”.

Итак, список опций:

1. Create documents – дает пользователю право на создание документов в базе. Пользователь с уровнем доступа Author, но не имеющий этой опции не сможет создать ни одного нового документа.
2. Delete documents – права на удаление всех доступных на чтение документов. Это является привилегией администратора базы, эту опцию нельзя давать кому попало, а то нажмут по ошибке Del, а разгребать пропажу документа придется разработчикам. В одной небезызвестной конторе, девочка, занимающая должность финансового директора, как-то нажала del на документе, в котором фиксировалась выдача кредита…. А разработчикам пришлось решать эту проблему…
3. Create personal agents – позволяет пользователю создать некоторую бизнес логику в дизайне базы, доступную на исполнение только ему

4. Create personal folders/Views – позволяет пользователю создать в дизайне базы доступные только ему папки или представления
5. Create shared folders/views – позволяет пользователю создать в дизайне базы доступные для всех папки или представления
6. Create Lotus Script/Java agents – позволяет пользователю создать некоторую бизнес логику в дизайне базы, доступную на исполнение всем пользователям базы
7. Read public documents – чтение публичных документов
8. Write public documents – создание публичных документов
9. Replicate or copy documents – позволяет реплицировать или копировать в буфер обмена документ из базы

3.2.3. Логические роли

Логические роли - это сущности, которым даются права на единицу функциональности базы. Роли создаются и им даются права в период проектировки и создания бизнес логики приложения. Например, у меня есть документы справочников, в которых хранится информация о конфигурации, о валютах, курсах валют, филиалах и пр. Некоторые пользователи отвечают за информационное наполнение справочников, некоторые их только читают, кому-то они вообще не доступны. При дизайне своей базы я могу создать 3 роли касающиеся справочников:

1. [CreateList] – этой роли я дам права на создание документов справочников
2. [ReadList] – этой роли – на чтение
3. [EditList] – этой роли – на редактирование

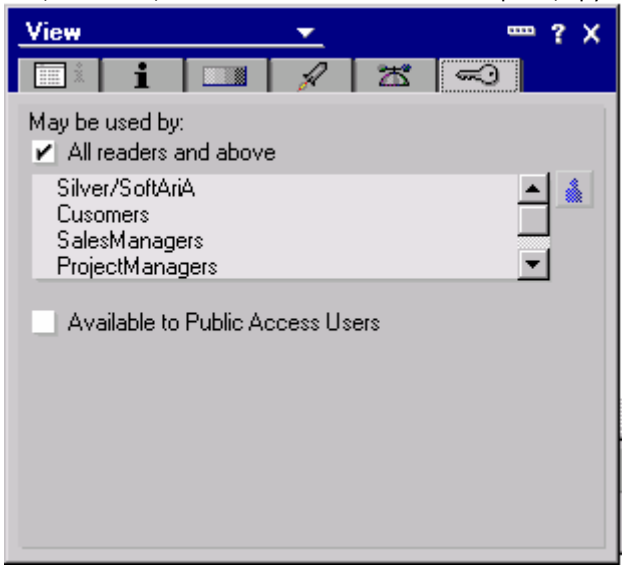
Далее, при создании форм документов справочников, я дам этим ролям права на создание документов по этим формам, чтение и редактирование соответственно. При эксплуатации базы, я даю группе пользователей роль, и группа получает права на данную этой роли функциональность.
За то, чтобы роль умела что-то делать в базе – отвечает программист, который при создании элементов дизайна – прописывает в них – какие именно роли что могут делать с этим элементом.

3.3. Документ

На уровне документа указывается - кто имеет права на редактирование и чтение этого документа. Для этого на документе при создании или в ходе работы с ним создаются поля специальных типов – Authors & Readers. Список ролей, групп, пользователей и серверов, указанный в поле, имеет права на редактирование и чтение документа соответственно.
Если на документе присутствует несколько полей одного типа, то список пользователей, ролей и групп в них – объединяются.
Для типа Readers есть одна фича – если вы хотите чтобы ВСЕ могли читать документ – создайте на нем поле типа readers с пустой строкой в качестве значения.

3.4. Элементы дизайна и формы

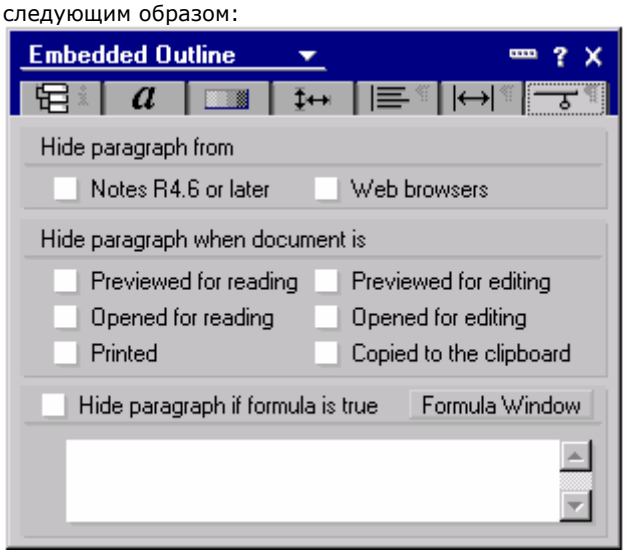
Для элементов дизайна, представляющих пользователю информацию – представлений, папок и т.д. на последней закладке свойств каждого элемента есть опции – Кто может использовать этот элемент дизайна. И там, в списке, вы можете выделить – какие роли, группы и т.д. могут видеть это представление, папку и т.д.



Для форм указывается – кто может создавать документы по этой форме и читать документы созданные с этой формой. Список читателей, заданный на форме, помещается автоматически в поле $Readers созданного документа. При этом надо отметить, что если вы создаете документ из кода, указываете в поле form его

форму и потом пересчитываете по форме, чтобы в документе появились все определенные формой поля со значениями по умолчанию, то поле $Readers в этом случае автоматически не посчитается. Его надо будет создать из кода отдельно и задать на нем список читателей документа.

Все элементы дизайна, отвечающие за представление документов и информации – страницы, формы, представления, папки – состоят в свою очередь из вложенных элементов, таких как поля, графика, таблицы, секции, объекты, кнопки, колонки и прочее. Большинство этих объектов в свою очередь имеют опции доступности – эти опции перечислены на последней закладке свойств каждого объекта и выглядят следующим образом:



Здесь можно указать – каким ролям, группам, пользователям виден этот элемент дизайна, в каком режиме – чтения или редактирования виден элемент, использовать язык формул для описания предиката, который в случае возврата True на текущем окружении, вызовет скрытие этого объекта от пользователя.
Таким способом достигается то факт, что пользователи с разными ролями будут видеть один документ, открытый одной формой – в разном виде.
Например, менеджер по персоналу будет редактировать все атрибуты карточки сотрудника, а сам сотрудник – только свою контактную информацию, потому что от сотрудника будут скрыты редактируемые поля для всей информации, кроме контактной, а показаны – только вычисляемые не редактируемые.

3.5. Права при запуске агентов

Бизнес логика, представленная агентами, может быть вызвана 2умя различными способами – на клиенте и на сервере. При запуске на клиенте бизнес логика выполняется с правами запустившего ее. При запуске на сервере (смотри способы запуска агентов в разделе, посвященном агентам), бизнес логика исполняется с правами пользователя, подписавшего агент. То есть, агент, который берет все документы, попавшие в папку «Корзина» и удаляющий их – запущенный по кнопке от имени рядового пользователя, не обладающего правами на удаление документов – выдаст ошибку. А при запуске этого агента сервером по расписанию, например, раз в час, от имени администратора, подписавшего агент – все документы будут успешно удалены.

4. Событийный механизм

В lotus есть понятие события. События происходят при произведении различных действий в базе пользователями или по факту каких-то внешних событий, типа прихода почты. Практически все элементы дизайна, являющиеся частями интерфейса пользователя имеют события - формы, подформы, представления, папки, часть объектов, такие как поля форм. События делятся на события, происходящие в lotus Notes – программируются они на языках доступных в Notes, и на события, происходящие на web. Тут надо сказать, что сервер Domino является http сервером, и на основе элементов дизайна и документов он генерирует html для доступа к данным по http-протоколу. Http-события программируются на java script.

В этом документе не будут полностью описаны все события всех элементов дизайна, будут описаны для примера события форм, а детальную спецификацию и поиск подходящего события для программирования какой-то частной бизнес логики, читатель может найти в Designer Help, который идет в поставке Lotus Designer.

4.1. Пример событий формы

4.1.1. События в lotus Notes клиенте

Надо сказать, что все события делятся на 2 вида – query-события и Post-события. Первые - это события, которые происходят прямо перед тем, как происходит инициируемое пользователей действие – например открытие, сохранение или закрытие формы. В этих событиях (которые в коде выглядят, как программируемые подпрограммы с передаваемыми параметрами) присутствует параметр Continue (он уже раньше встречался в описании событий базы), который регулирует - продолжать ли действие, инициированное пользователем. «Continue=False» – вызывает прерывание действия пользователя по открытию, сохранению, изменению режима чтения/редактирования или закрытию формы.
Post-события происходят после действия выполняемого пользователем.

Ниже приведен список событий формы с описанием их смысла и примерами использования:

1. QueryOpen – происходит прямо перед открытием формы в рабочей области. На этом событии из кода еще не доступны поля документа, если документ новый. Если документ уже существует, то на нем можно автоматически насчитывать какие-то поля – например, в карточке сотрудника, на которую ссылаются по уникальному Иду (ID), карточки должностей, на которые сотрудник назначен – можно насчитывать при каждом открытии – текущие занимаемые должности.
2. PostOpen – на этом событии можно работать с полями документа, даже если он новый. Сюда часто вносят бизнес логику расчета различных значений в документ. Есть тонкость – если значения насчитаны в документ на этом событии, то эти значения отобразятся в форме только после обновления документа (call source.refresh).
3. QuryModeChange – это событие происходит прямо перед изменением режима документа с чтения на редактирование и обратно. На этом событии можно запретить перевод документа в режим редактирования, при выполнении каких-то условий.
4. PostModeChange – событие сразу после перехода документа в другой режим.
5. PostRecalc – после пересчета документа. Пересчет документа это его рефреш из кода, формул или при нажатии F9 – он ведет к пересчету вычисляемых значений и формул скрытия.
6. QuerySave – прямо перед сохранением документа. Сюда мы можем положить бизнес логику проверки корректности введенной информации, в случае если валидация на уровне поля по какой-либо причине является не эффективной. И запретить сохранение в случае некорректного заполнения.
7. PostSave – после сохранения документа – сюда мы можем поместить например, синхронизацию обновленной информации в какие-либо связанные документы. Например, при изменении имени в карточке сотрудника синхронизовать эту информацию в резюме.
8. QueryClose – событие происходящее при закрытии документа прямо перед закрытием. На этом событии можно обнулить какие-то насчитанные на QueryOpen поля.

4.1.2. События на web

1. WebQueryOpen – это внутреннее Lotus событие связанное с открытием формы на Веб – оно происходит на сервере по запросу снаружи об открытии формы. В этом событии может использоваться язык программирования формул и формулами может быть написан вызов агента, который в свою очередь может быть написан на Lotus script, Java, formula language, etc. То есть фактически на событие открытия формы по http может быть навешана любая бизнес логика, которая будет выполнена сервером.
2. WebQuerySave –это событие аналогично WebQueryOpen, только выполняется прямо перед сохранением документа в базе.
3. JSHeader – сюда можно написать бизнес логику на java script, которую потом вызывать из других событий
4. OnSubmit – java скриптовое событие html-формы onSubmit.
5. Существует масса других событий, которые перекликаются к событиями java script на стандартных HTML-страницах. Здесь они не будут все освещены, так как несколько выходят за тематику данного документа

4.2. Пример событий для агента

Агенты – как уже сказано выше – законченные куски бизнес логики, которые могут срабатывать на каком-либо документе или множестве документов. Агента можно запустить вручную, повесив его на кнопку или иную логику вызова (например, с web-события формы), а можно указать событие в базе, по факту происхождения которого агент будет запущен:

1. Before new mail arrives – прямо перед тем как в базу пришла новая почта. На этом событии ее можно, например, поместить в какую-либо папку. Независимо от выполненных действий, документ письма будет помечен как непрочитанный.
2. After new mail has arrived – аналогичное событие, происходящее сразу после прихода почты. На этом событии с приходящим документом можно сделать некоторые действия – например, разобрать текст тела письма и создать в базе новый документ согласно указанной в теле информации.
3. If document have been created or modified – в случае создания или модификации каких-либо документов в базе
4. If documents have been pasted – если в базу из буфера был скопирован документ. На этом событии часто вешают блокировки копирования документов – удаляя только что скопированные документы.

Однако это не есть правильный путь, так как агент удаления запускается от имени скопировавшего документы. Рядовой пользователь не имеет прав на удаление документов из базы, поэтому процедура удаления, запущенная от его имени, выдаст ошибку, связанную с недостатком прав. Правильно сделать агента, удаляющего документы, переданные ему в качестве параметра и запускать его методом RunOnServer из агента срабатывающего по этому событию. И передавать скопированные документы в качестве параметра. Так как запущенный на сервере агент будет выполнять действия от имени подписавшего его – то есть администратора – процедура удаления не вызовет ошибок (смотри пункт посвященный правам на запуск агентов из раздела «Система прав доступа»).

5. On schedule – запуск агентов сервером по указанному расписанию (раз в некоторый промежуток времени, раз в день, раз в неделю и т.д.)

5. Языки программирования

Ранее уже упоминались и здесь будут формально перечислены способы программирования бизнес логики в Lotus Notes.

1. Lotus script – это объектно-ориентированный и расширенный набором классов для доступа к объектам Lotus, BASIC.
2. Formula language – язык формул, в 2-х словах описать его основную сущность трудно, но все таки это набор стандартных команд, функций, работающих в рамках текущего окружения. В этом языке отсутствуют циклы. Однако по скорости и лаконичности он превосходит Lotus script. Например, поиски по ключу эффективнее делать на языке формул, валидацию переменных окружения тоже. Хотя, по поводу второго это частное мнение автора.
3. Java – вся мощь Java к вашим услугам. На java также доступен набор классов, реализующих доступ к объектам Lotus. Если вы работали под Visual Age for Java – то там вы могли видеть классы для работы с объектами Lotus. Однако, на несертифицированных платформах (тех, на которые Lotus ставится, но которые не присутствуют в официальном списке платформ для Lotus) возможны глюки вплоть до полного повисания сервера при запуске Java кусков.
4. Java script – используется для программирования бизнес логики для Веб интерфейсов
5. Simple action – набор простейших действий, выбирается из меню и используется для упрощения программирования примитивных функций.
6. C API – возможность написания внешних процедур на C с использованием поставляемых процедур для работы с объектами Lotus. (это автор знает понаслышке и на уровне беглого просмотра, поэтому эта часть не будет освещена в этом документе, кто имеет опыт – может вписать свою главу.)

6. Элементы дизайна базы – детальная концепция

Чтобы дать представление о дизайне – лучше всего показать все на примере создания некоторого приложения. Для примера предлагается создать систему голосования. Я не буду здесь уделять внимание проведению анализа и сбору требований, а также написанию SAD – Software Architecture document-а. Задача будет поставлена следующим образом:

1. Система функционирует на сервере (о ее распределенности и локальной работе пока речь не идет)
2. Есть участник голосования, голос которого учитывается с некоторым весом, участник может быть активным и неактивным.
3. Есть голосование, инициируемое полномочным лицом. В голосовании указывается тема голосования, тип голосования – открытое, закрытое и вид голосования – выбрать конкретный вариант, выбрать несколько, проставить ранг для вариантов, ввести свои комментарии к каждому варианту. Голосование может также учитывать кол-во акций голосующего (его вес) или не учитывать при подсчете результатов. У голосования есть некоторое воркфлоу – то есть процесс жизни, но программирование этого процесса будет детально освещено в курсе, посвященном разработкам под Lotus workflow, а сейчас мы будем рассматривать голосование как статичный документ. У голосования есть список участников, которым рассылаются бюллетени. У голосования есть дата окончания сбора голосов.
4. Участники голосования получают в свои почтовые ящики ссылку на документ предназначенного им бюллетеня и, открыв бюллетень, голосуют. В нормальной ситуации документ бюллетеня проходит следующие фазы – черновик -> заполнен, но в этом примере мы будем рассматривать бюллетень как статический документ.
5. Инициатор голосования может динамически видеть текущие результаты голосования.
6. После наступления даты окончания голосования система отбирает права на голосование участников, и само голосование помечает как закрытое.

Модель, описывающая бизнес акторов следующая:

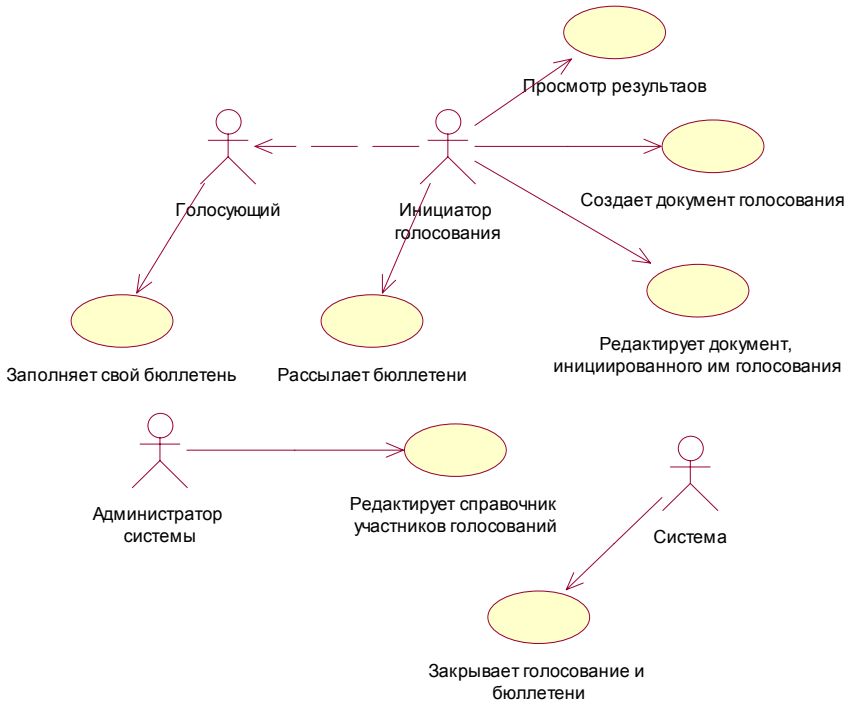
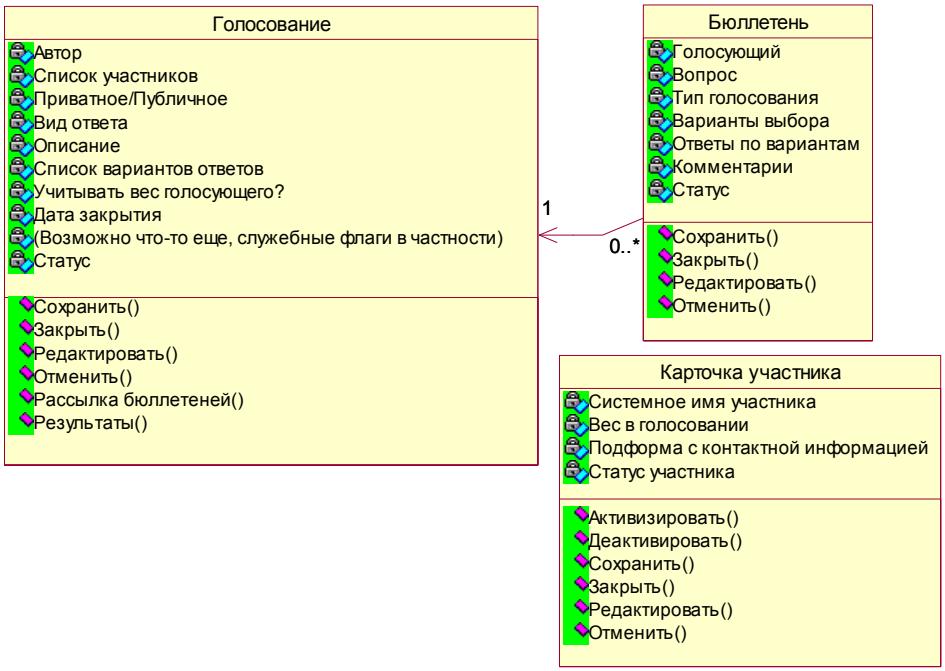


Схема бизнес сущностей и мультипликативность связей между ними следующая:



Надо создать пустую базу без дизайна- file->Database->New – в окошке указать сервер, имя файла с расширением .nsf для новой базы и не выбирать никакого шаблона. Если вы выберете шаблон, на основе которого создавать базу – то она будет создана с дизайном из этого шаблона. Набор шаблонов идет в стандартной поставке Lotus и все созданные вами приложения тоже могут быть помещены вами в список шаблонов.
Первое, с чего начинается дизайн приложения – это проектировка и создание системы прав доступа.
В нашем случае, предлагается создать следующие логические роли:

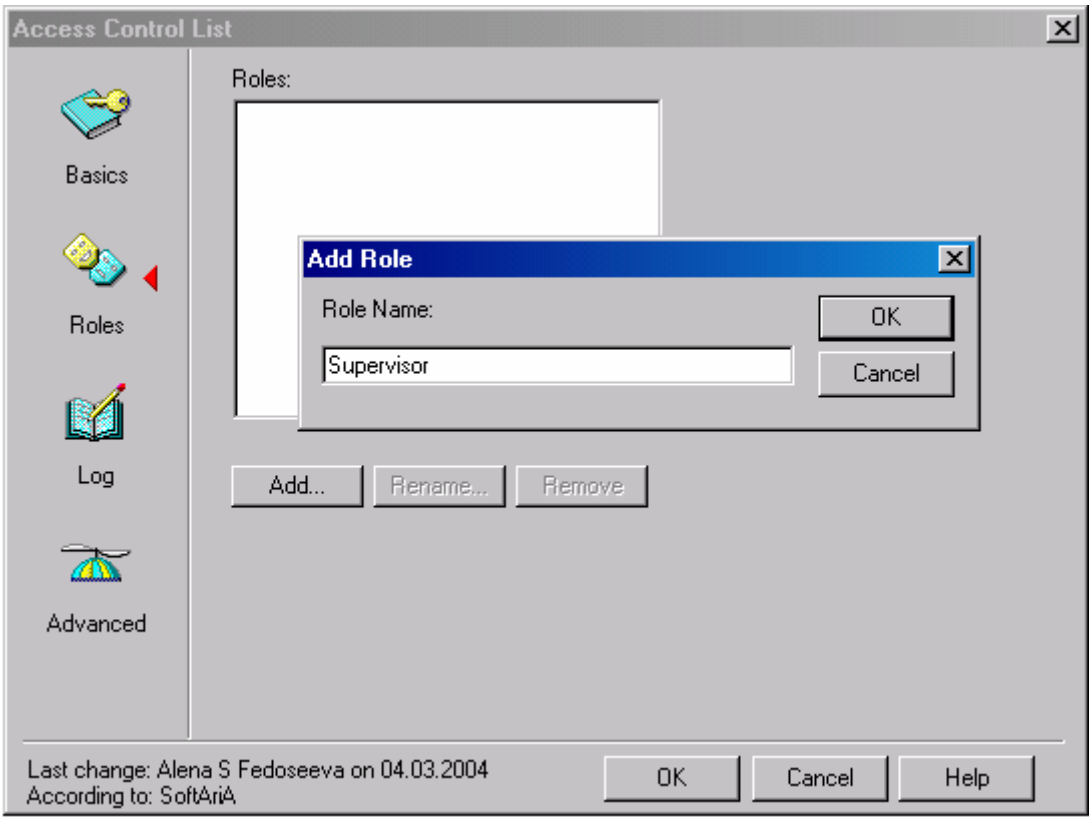
1. [Supervisor] – может все
2. [Reader] – все читает
3. [MemberCreate] – создает справочник участников (например, председатель собрания акционеров)

4. [MemberRead] – читает справочник участников – очевидно, этой ролью должны обладать инициаторы голосования
5. [MemberEdit] – редактирование справочника участников (председатель собрания акционеров)
6. [VoteCreate] – создает голосование
7. [VoteRead] – читает все документы голосований
8. [VoteEdit] – редактирует все документы голосований
9. [BillCreate] – создает документы бюллетеней (нужно дать инициатору)
10. [BillRead] – читает ВСЕ документы бюллетеней
11. [BillEdit] – редактирует все документы бюллетеней

Этот набор ролей дает нам полное описание прав доступа к типам документов в системе. Тут надо также отметить персональные права доступа:

1. Участник голосования видит документ голосования, в котором участвует и бюллетени участников, если голосование публичное. Свой бюллетень он редактирует, пока голосование не закроется.
2. Инициатор голосования видит бюллетени участников голосования и редактирует созданное им голосование, пока оно не будет закрыто

Чтобы создать список ролей file->database->Access control list и далее закладка Roles:



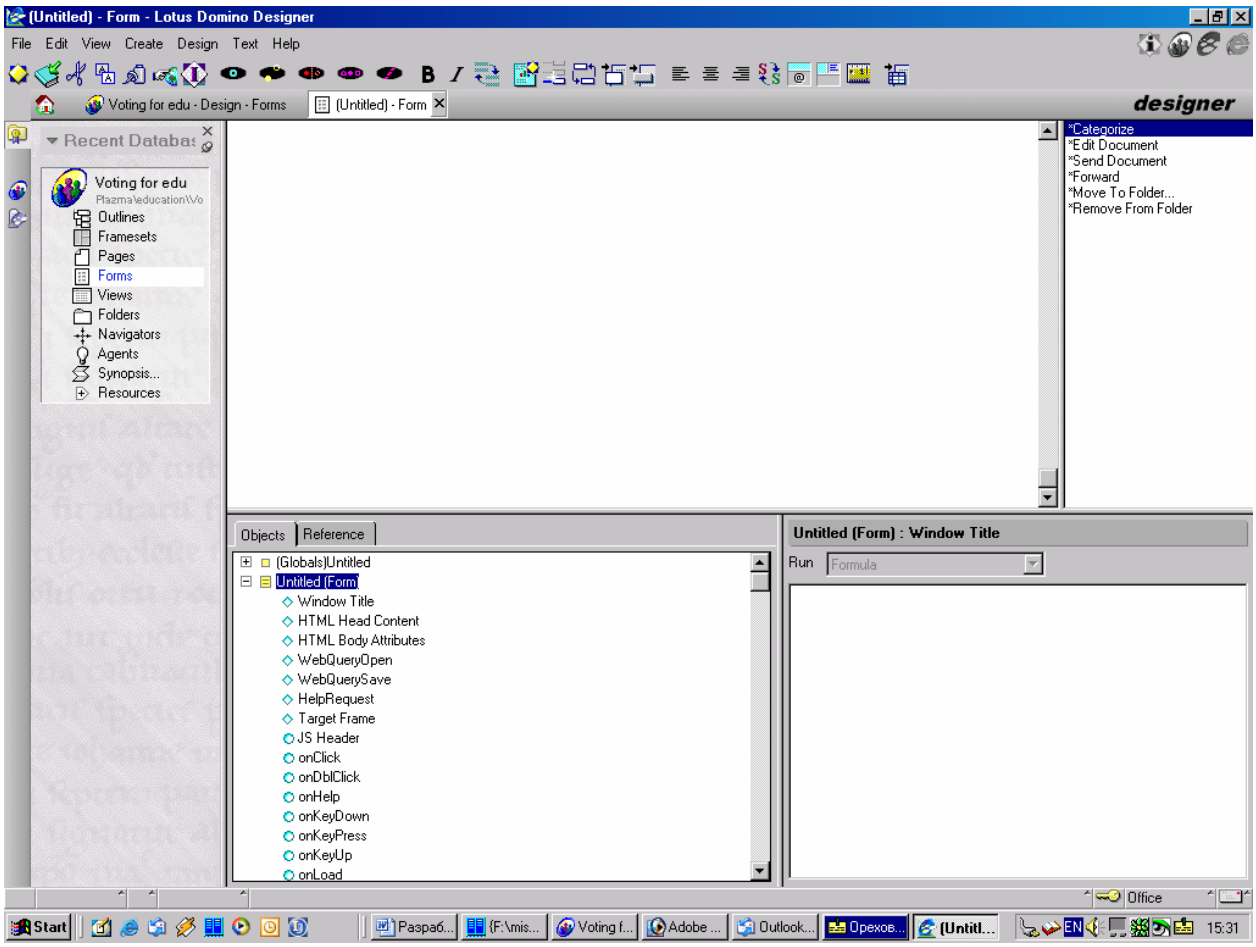
При внесении ролей не пишите квадратные скобки – система сама поместит в них, введенное вами название. Далее ВНИМАНИЕ! Когда вы будете использовать имена ролей, имейте в виду – они воспринимаются системой как текстовая строка, поэтому являются CASE-sensitive.

6.1. Формы

Итак, начнем создавать форму участника, и по ходу будем рассматривать атрибуты и объекты формы. Надо сразу понимать, что жизнь состоит из мелочей. И для того чтобы жизнь была качественной – она должна быть качественной во всех мелочах. Эта парадигма имеет отношение и к созданию приложений. Приложения должны быть эргономичны, и соответствовать требованиям пользователя во всех мелочах.

Будьте готовы – программирование под Lotus это процентов на 70 – выставление всяких галочек на свойствах, это требует массы аккуратности, формирует педантичный характер и иногда нервирует.
Ладно, поехали:
Откройте созданную базу в дизайнере. Встаньте на закладку Forms – и нажмите над списком форм кнопку "New form"

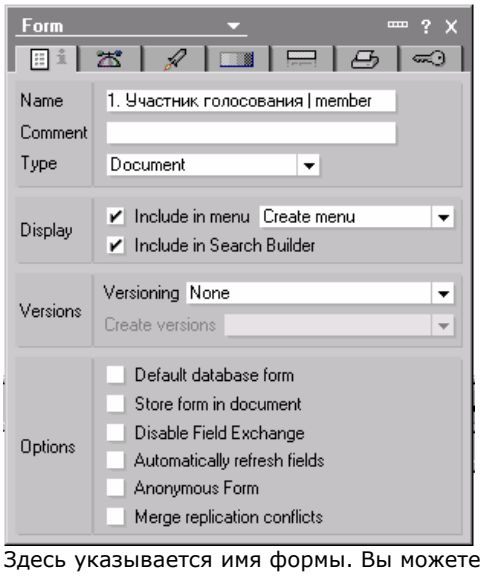
Перед вами откроется окно для дизайна формы, оно состоит из 4 областей:



Сверху слева – это тело формы, в котором создаются поля ввода и прочие объекты.
Сверху справа – область, в которую помещаются кнопки, доступные на этой форме, сейчас там стоят кнопки по умолчанию, и их отображение не включено.
Снизу слева – события и свойства того объекта, на котором стоит курсор. Сейчас курсор стоит на форме, поэтому там отображаются свойства и события формы.
Снизу справа – детализация значения того свойства или события в нижнем левом окне, ан котором стоит курсор.

Кроме этого есть также набор свойств объекта, который вызывается двойным щелчком мыши на объекте или, для контейнеров верхнего уровня типа форм или представлений, - из меню Design - > Form properties.

6.1.1. Form Properties



Здесь указывается имя формы. Вы можете для любого элемента дизайна иметь имя и набор алиасов, по которым можно обращаться к этому элементу. Алиасы перечисляются через «трубу». При создании документа по форме, в поле form пропишется название последнего алиаса из списка.

В Type указывается, что это будет за объект. В данном случае это просто документ. Возможны еще варианты – Response и Response to response, которые используются для создания форм ответов на документы.

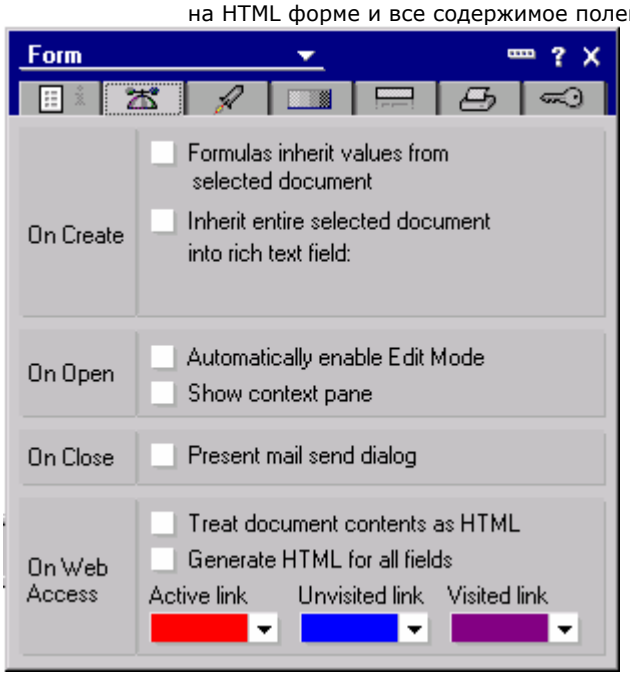
Далее идет включение создания формы в меню и в поиск. Если при создании формы не требуется выполнения какой-либо еще бизнес логики – то форму можно включить в меню Create – тогда в клиенте пользователь сможет создать документ по этой форме просто из меню.
Внимание: Если вы используете кнопки для создания документа, которые выполняют при создании некоторую бизнес логику (например, подгружают в бюджетную заявку данные за предыдущий период и пр.) то ни в коем случае не допускайте создание документа через меню Create.
Включение версионности используется для сохранения истории модификаций документа созданного по форме.

Опции:

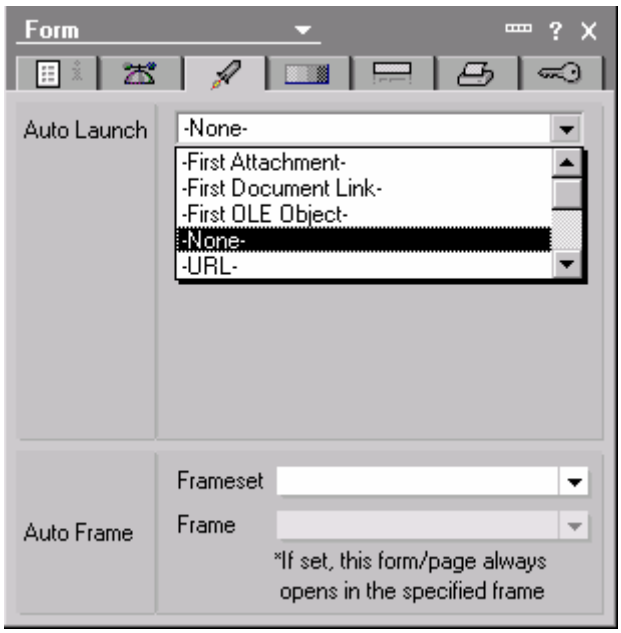
1. Default database form – в случае если у вас в базе есть документ у которого в поле FORM прописано значение, и в базе нет формы с таким именем (алиасом), тогда система возьмет форму с этим включенным флагом и откроет документ в этой форме.
2. Store form in document – эта опция включает сохранение информации о форме в документах, по ней созданных. В результате – если вы скопировали такой документ в базу, в которой нет формы, указанной в поле form этого документа, документ все равно откроется по этой форме. Однако хранение информации о форме чрезвычайно утяжелит документы.
3. Disable Field Exchange – автор не пользовался этой опцией за все 6 лет работы с Lotus, а говорить банальности, основанные на переводе официального хэлпа не хочет, поэтому кому понадобиться – разбирайтесь сами. (надо писать надо!!!!!)
4. Automatically refresh fields – эта опция включает пересчет всех полей документа при вводе каждого символа. Эта опция включается в довольно редких случаях. При пересчете документа происходит его отрисовка, что вызывает эффект дерганья экрана и сильно раздражает пользователей. Это надо иметь в виду. Но иногда нужно, чтобы вычисляемые поля документа пересчитывались при вводе каждого символа, например, я ввожу имя клиента в одно поле и при каждом вводе символа – факт введения отображается в поле, в котором это имя отображается в специальном формате.
5. Anonymous form – при включении этой опции не будет сохраняться информация о персоне последний раз редактировавшей документ. Это удобно использовать при дизайне базы «Книга жалоб на некомпетентность руководства» ☺.
6. Merge replication conflicts – эта опция включает слияние документов порожденных при конфликтах репликации. Надо отдельно сказать о них. Допустим, есть 2 сервера А и Б, на них есть реплики вашей базы, обновление происходит раз в 15 минут. Прошел обмен обновлениями и после этого Вася и Петя зашли каждый на свой сервер и поправили один и тот же документ. Что произойдет после следующего обмена данными? Правильно – возникнет конфликт, который в нормальной ситуации будет присоединен к основному документу в виде документа ответа – и выбрать главный документ – это уже прерогатива администратора. У этих документов будет одинаковый UniversalID (смотри 2-ую главу пункт Документы и Формы). В ряде случаев систему можно заставить эти документы слить в один – детального алгоритма слияния автор не знает, но слияние происходит на уровне полей, в документы забираются последние обновленные значения. Надо понимать, что при включении этой опции документ может стать внутренне противоречивым.

Итак, мы создали форму и поименовали ее. Теперь сохраните ее Control-S.
Я далее кратко пробегусь по закладкам и основным свойствам form properties:
Вторая закладка посвящена тому, что делать с формой при создании, открытии, закрытии и доступе через web.

1. Опции создания
    a. Formulas inherit values from selected documents – эта опция позволяет наследовать значения вычисляемых полей с документа, который был выбран в рабочей области в момент создания нового. Например, при создании документа должности, если в момент создания я стою на документе отдела, в котором создается должность. В поле отдел создаваемой должности будет наследоваться информация из поля «Название отдела» документа отдела.
    b. Inherit entire selected document into rich text field – эта опция позволяет полностью включить всю содержимое родительского документа в РТФ поле новосоздаваемого.
2. Опции открытия
    a. Automatically enable Edit mode – автоматически переводит документ в режим редактирования при открытии. Если эта опция включена, а прав на редактирование у меня нет, то документ откроется на чтение.
    b. Show context pane – Эта опция позволяет открывать документ в разделенном пополам окне, во второй половине которого будет открываться например документ родителя (для документа ответа) или документ, на который присутствует ссылка.
3. Опции закрытия
    a. Present mail sent dialog – автоматически включает диалог посылки документа по почте.
4. Опции доступа с Web
    a. Treat document contents as HTML – при включении этой опции по форме генерируется HTML, который показывает форматирование согласно логике формы, но интерпретирует поля как текст
    b. Generate HTML for all fields – а эта опция включает интерпретацию полей формы, как полей на HTML форме и все содержимое полей формы является содержимым полей HTML формы.

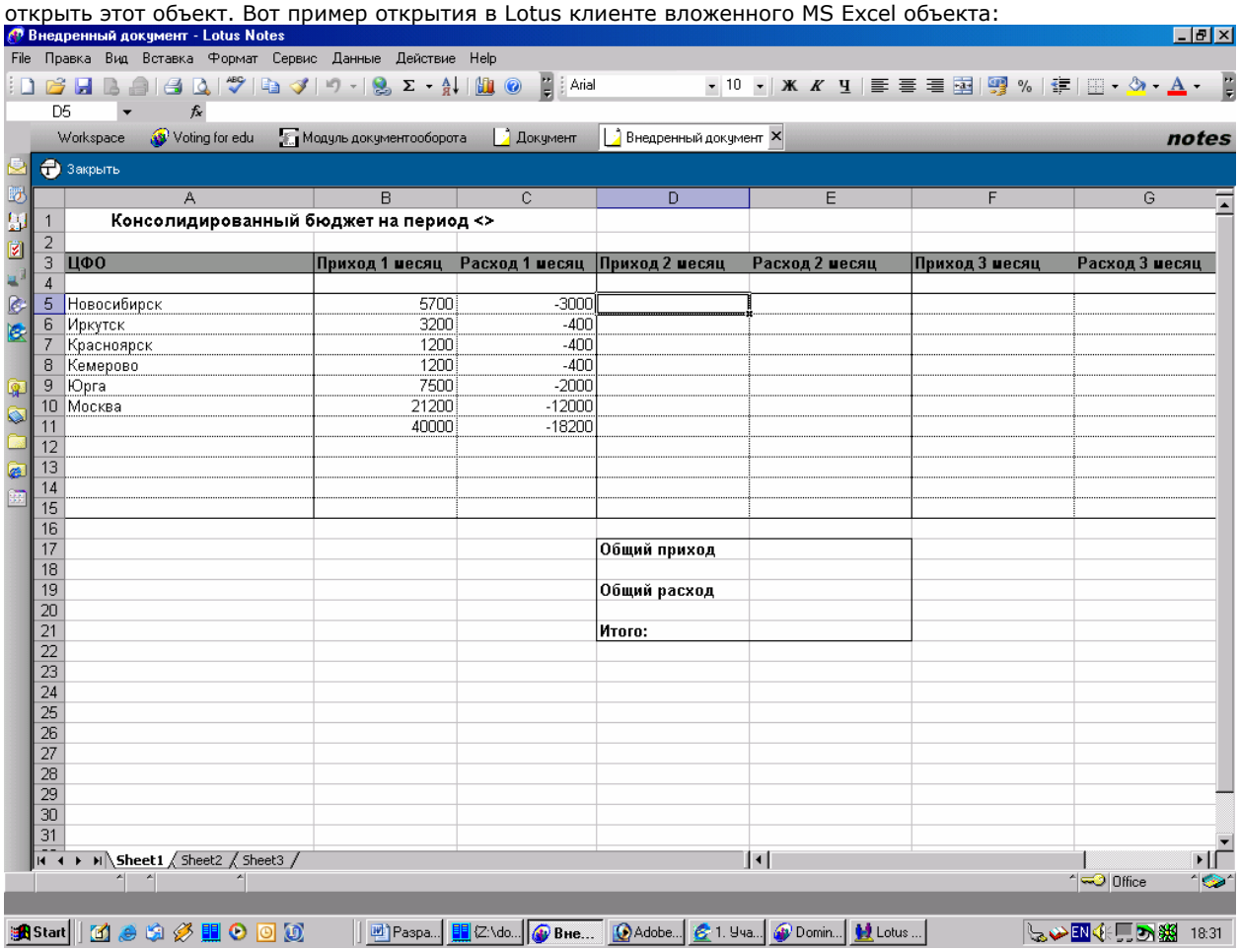


Следующая закладка содержит информацию о том, какой из вложенных в документ объектов открывать при открытии формы, а также информацию об открытии формы в нестандартном frameset-е.



Я опущу опции открытия формы в отдельном фрэймсете – если интересно – просто поэкспериментируйте с ними, а вот Auto Launch – это важная опция – о ней надо знать.

Ранее уже говорилось о том, что в документе Lotus можно хранить вложенные объекты. Объекты могут быть вложенными атачментами, ссылками на другой документ, OLE-объектом, URL. При открытии документа можно открыть этот объект. Вот пример открытия в Lotus клиенте вложенного MS Excel объекта:



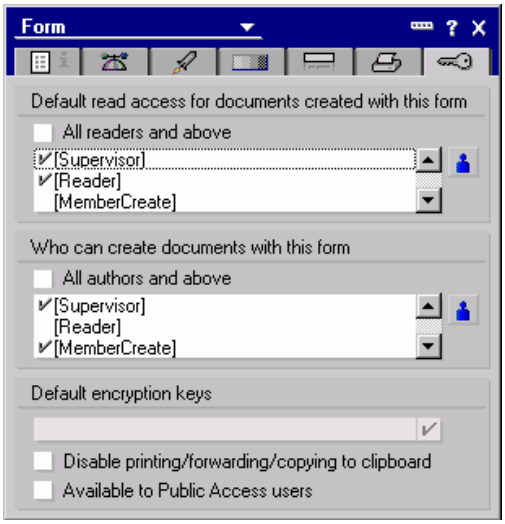
Следующая закладка посвящена параметрам визуализации формы – можно задать цвет бэграунда формы, задать графический ресурс и параметры графики. Для человека, впервые начинающего дизайн формы имеет

смысл сказать только об одном параметре Do not tile graphic – этот параметр отвечает за то – один раз нарисовать картинку или размножить ее на всю форму.

Пятая закладка также отвечает за параметры отображения и позволяет сделать у формы заголовок, в который помещается информация, которая должна быть видна пользователю постоянно, независимо от положения скроллбара.

Шестая закладка – это опции печати.

Седьмая закладка это опции прав доступа к форме и документам по ней созданным. На этой закладке для нашего приложения мы должны указать те роли, которые имеют права на создание документов по это форме и чтение документов, созданных по этой форме.

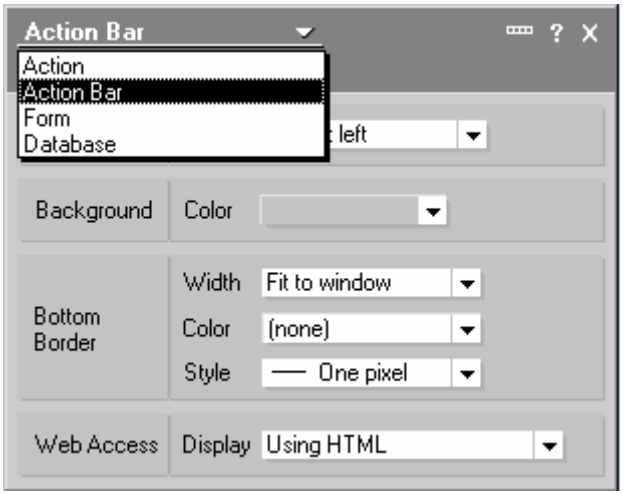


Итак, мы указываем в качестве читателей роли [Supervisor], [Reader], [MemberRead] и [MemberEdit]. В качестве создателей мы прописываем роли [MemberCreate] и [Supervisor]. Здесь мы также можем запретить копирование в буфер обмена, печать и форвард. На этой же закладке может быть дан публичный доступ к документам, созданным по этой форме. О публичном доступе было сказано в разделе, посвященном системе раздачи прав, подразделу, посвященному базам данных – уровню доступа и опциям.

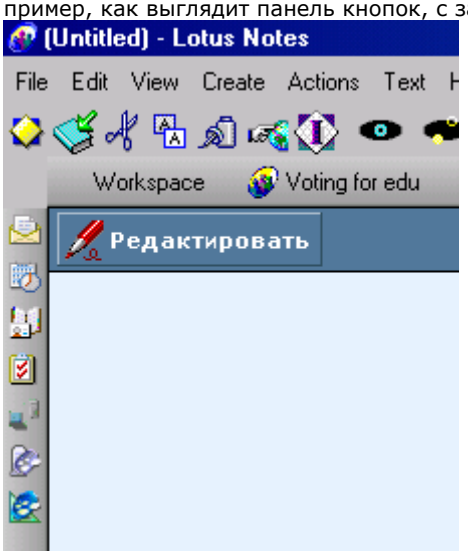
Итак, мы прописали свойства формы.

6.1.2. Графика панели кнопок

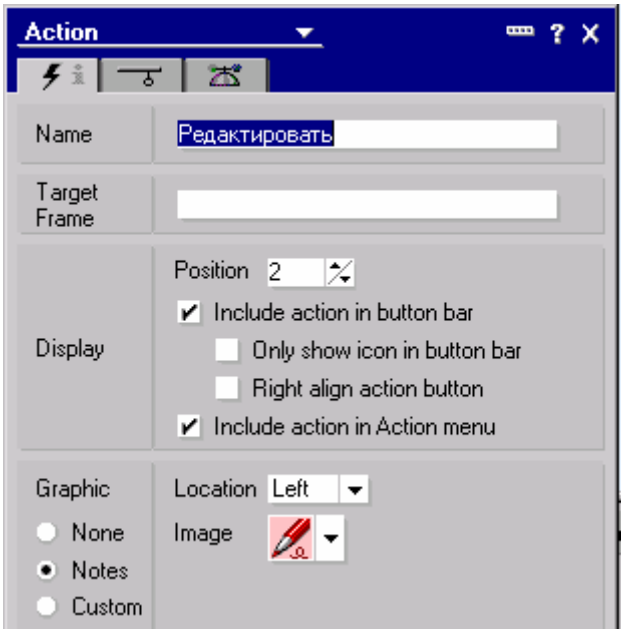
Предлагается сразу прописать свойства графики формы и графики панели, чтобы один раз разработать стиль дизайна и во все остальные формы копировать с уже созданного стиля, а не натягивать дизайн форм на все формы, после создания бизнес логики.
Графика панели задается из окна свойств - следующим образом. В окне свойств формы в самом верху окна, где написано Form – открывается выпадающее окошко, в котором перечислено – свойство каких текущих объектов можно редактировать. Поставьте курсор в правое верхнее окно, на котором задаются кнопки, доступные на форме, и после этого выберите в окошке свойств Action bar:



В этом окне на его закладках вы задаете графические свойства верхней панели и кнопок на панели. Вот пример, как выглядит панель кнопок, с заданными мной параметрами отображения цветов и кнопок:



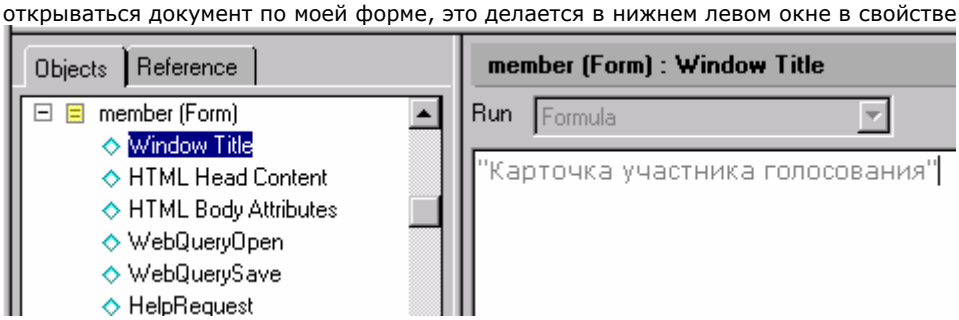
Для того чтобы на пустой форме, вы могли опробовать параметры заданной вами графики – вам необходимо визуализировать хотя бы одну кнопку. Автор выбрал кнопку Edit document, которую переименовал на русскоязычный манер. Для этого автор 2 раза кликнул на названии этой кнопки в списке кнопок верхнего правого окна и в свойствах кнопки задал следующее:



Подробно свойства кнопок будут описаны в разделе, им посвященном.

6.1.3. Заголовок формы

Теперь стиль дизайна документов у меня определен. Мне надо озаглавить окно, в котором у меня будет открываться документ по моей форме, это делается в нижнем левом окне в свойстве формы Window Title:



Теперь при открытии этой формы на рабочем пространстве – это делается из меню Action->Preview in Notes – я увижу форму с заголовком.

На текущий момент мы проставили свойства формы и привели в порядок ее визуализацию. Теперь приступим к дизайну бизнес логики формы.

Хорошим тоном считается реализовать в форме не только основную логику, но и предусмотреть такие пожелания пользователя, как видение автора и дату создания документа, автора и дату последнего изменения, возможно историю каких-либо изменений (например, переходов по статусам), предусмотреть специальные поля для внесения комментариев и вложения каких-либо объектов.

Поэтому мы создадим подформу, которой будем пользоваться во всех документах, которая будет хранить дополнительную информацию такого рода.

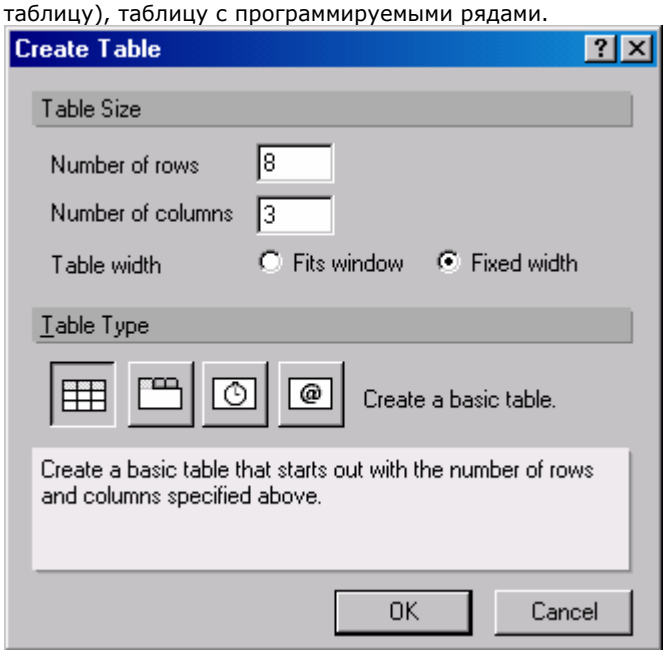
6.1.4. Создание подформы

Итак, мы заходим в Resources\Subform и создаем новую подформу. Далее в Design properties мы задаем ее имя. Опции внизу окна свойств регулируют включение имени подформы в диалоги включения подформы при создании новой формы и Create-> Subform диалог при включении подформы в существующую форму.

В общем, я создала подформу и назвала ее «Дополнительная информация | OtherInfo». Теперь, чтобы достичь удобства представления информации, мы сделаем форматирование при помощи таблицы.

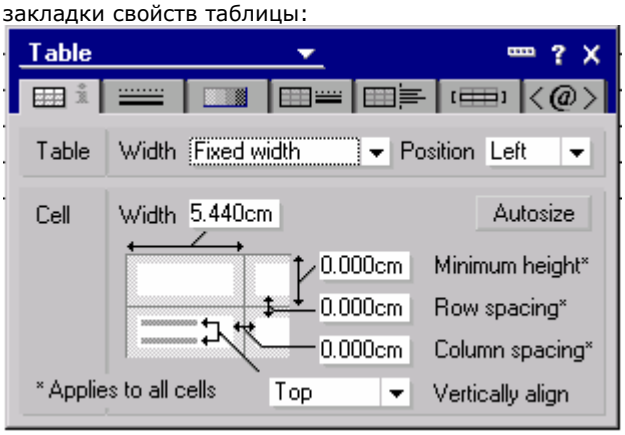
6.1.5. Создание таблицы и свойства таблицы

Для этого в меню Create->Table. Я создаю простейшую таблицу, но фактически я могу создать таблицу с закладками, таблицу с рядами, которые показываются в течение 2 секунд каждый ряд (анимационную таблицу), таблицу с программируемыми рядами.



Теперь, после нажатия «Okey», Lotus создал в подформе таблицу с 8 рядами и 3 столбцами в самом примитивном виде. Далее я приведу ее к тому виду, какой требуется мне. У таблицы есть параметры, относящиеся ко ВСЕЙ таблице и параметры, относящиеся к выделенной курсором зоне. Например, расстояние между рядами и строками – это атрибут таблицы, а цвет заливки – атрибут выделенной зоны.

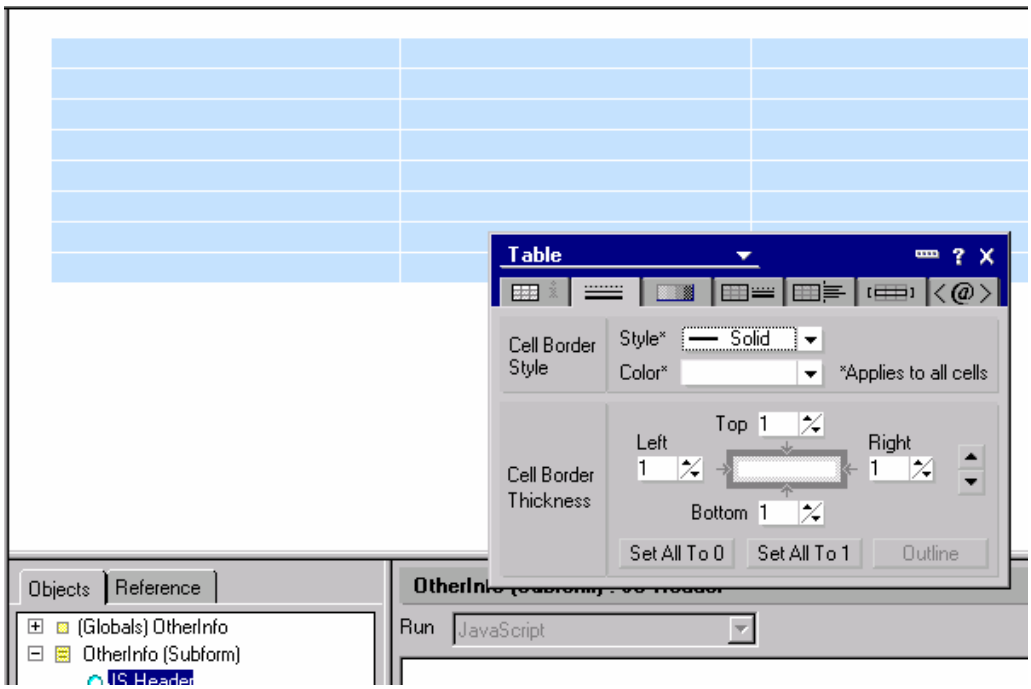
Если я буду перечислять тут все параметры – то в голове у вас возникнет каша, поэтому, встав на таблице в меню, вызовите Table->Properties и поиграйте сами со свойствами таблицы. А я кратко расскажу про закладки свойств таблицы:



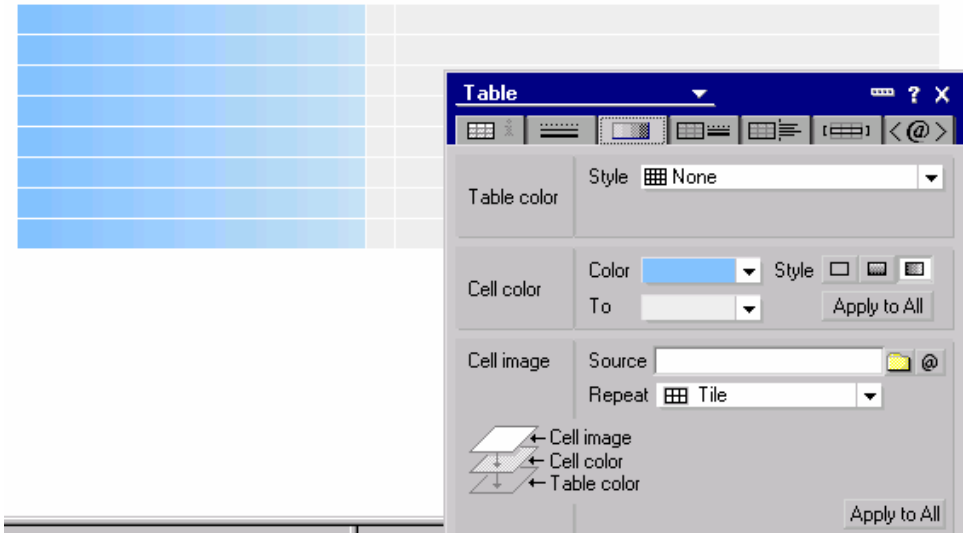
Первая закладка посвящена позиционированию таблицы на экране и ее размеру по ширине – можно ее вписать в экран или другую таблицу, можно задать фиксированную длину, состоящую из длин ее столбцов. При включении режима вписания, попытка задать ширину конкретного столбца вам не удастся, так как ширина будет меняться, но Lotus будет сам форматировать и подгонять ширину всех столбцов, что неизбежно приведет к отклонению ширины столбца, от заданного вами. Просто имейте это в виду.

Далее в этой же закладке задается ширина ячейки, которая распространяется на весь столбец, к которому относится ячейка. То есть ширина выделенного столбца задается в Cell -> Width.
Далее задаются свойства отступов, минимальной высоты и позиционирования текста, часть из которых распространяются на всю таблицу.

Вторая закладка посвящена заданию стиля и цвета разделительных полос.
Здесь вы можете отрегулировать ширину полос, цвет, стиль. Эти свойства применяются к выделенным ячейкам таблицы.



На третьей закладке задаются параметры графики выделенных ячеек, в своем варианте таблицы автор сделал градиентную заливку для первой колонки таблицы и обычный слегка серый цвет для 2-ой и 3-ей.



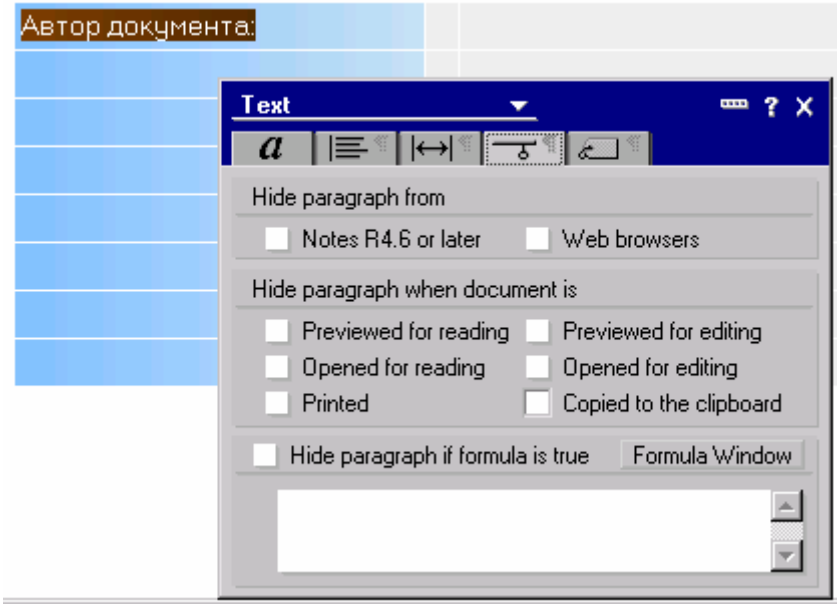
Четвертая закладка посвящена параметрам бордюра таблицы.
Пятая – подписи к закладкам для таблицы с закладками, параметры отступа таблицы и вписания текста, шестая – параметры показа ячеек, последняя закладка – возможность задания атрибутов и стилей для HTML тэгов <table>, <tr> и <td>.
Итак, таблицу мы сделали, в дальнейшем этот стиль дизайна таблицы мы будем использовать и для других форм.

Для работы со столбцами, строками и ячейками таблицы используйте пункты меню Table (insert, append, delete). Только помните, что ваши действия будут совершаться над выделенной вами ячейкой, строкой, колонкой.

Теперь нам надо внести в таблицу поля для отображения информации об авторе, последнем изменившем документ, дате создания и последней модификации, поле для вложения объектов и поле для хранения истории.

6.1.6. Текст в форме

В форму можно практически в любое место вписать текст – просто поставьте курсор и печатайте. Потом вы можете выделить этот текст и в меню выбрать Text->Properties. Далее вы можете выбрать свойства текста – фонты, размер, стиль, цвет и т.п. Важна предпоследняя закладка, она отличается от стандартного набора свойств текста доступных, например, в текстовых редакторах. Это закладка характерная для практически любого объекта формы – закладка, посвященная скрытию этого объекта.



На этой закладке можно скрыть объект в различных режимах просмотра документа, а также написать на языке формул предикат, который при возврате на текущем документе и текущих переменных окружения True – приведет к скрытию объекта долой с глаз пользователя. В нашем случае на не требуется скрывать этот текст от кого-либо.

В результате у меня получилась таблица вот с таким текстовым стилем:

Автор документа:
Последний раз изменил:
Дата создания:
Дата последнего изменения:
Вложенные (комментарии):
История изменений:

У меня получилась пара лишних строк, которые я удалю, выделив их предварительно, из меню Table ->Delete selected rows.

Теперь мы будем создавать поля.

6.1.7. Поля на форме

Поля мы будем создавать в 3-ей колонке, а вторую мы для однообразия будем использовать везде для размещения кнопочек, реализующих различную логику выбора из справочников и прочей бизнес логики, применимой к некоторым полям.

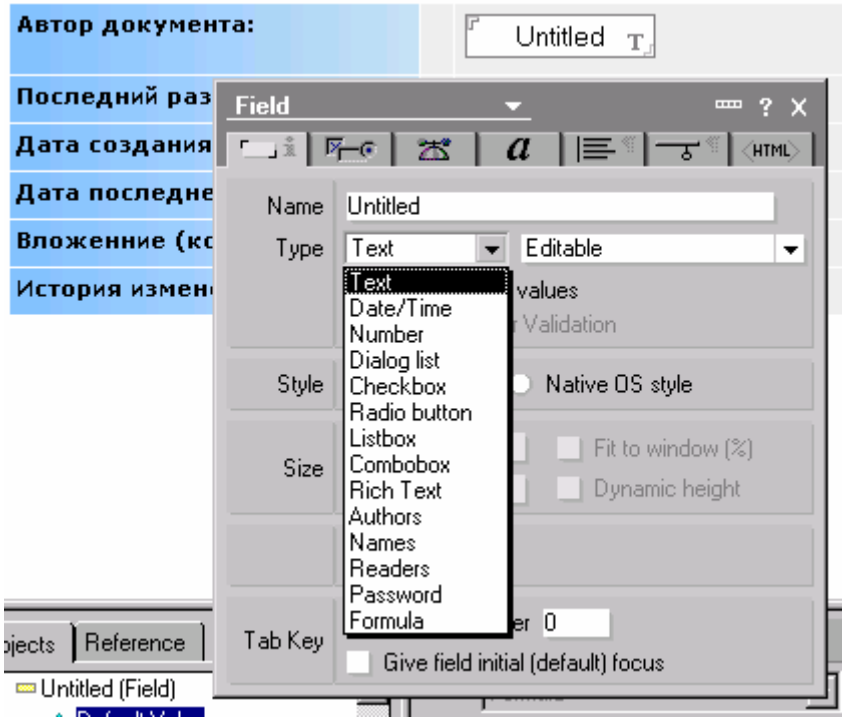
Чтобы создать поле – поставьте курсор в то место, где вы хотите его расположить и в меню выберите Create-> field. У вас появится поле. Двойным щелчком мыши на нем вы вызовете окно свойств текущего поля.

И… готовьтесь – вам предстоит переварить массу информации, вся она нужная, ее надо запомнить. Чтобы ее усвоить лучше - надо не просто следовать примеру, разбираемому в этом документе, а посоздавать свои поля разных типов и посмотреть – что из этого выйдет.

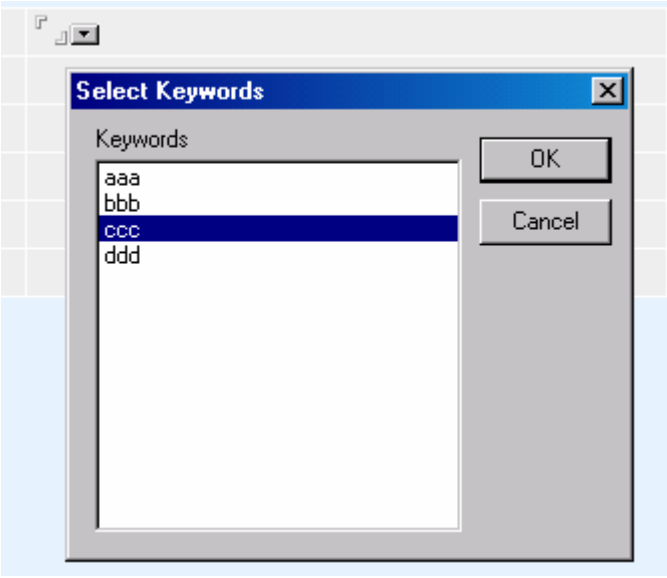
Для начала я кратко опишу смысл закладок свойств поля.

1. Здесь задается тип и вид поля, а также опции отображения и обхода полей табуляцией.
2. Эта закладка имеет разный вид, в зависимости от того – какой тип и вид поля вы выберете на первой закладке. На ней будут задаваться параметры отображения числе для числового типа, дат, параметры задания диалога для диалоговых окон, опции автоматического рефреша документа, при изменении значения поля для check box & radio button, и прочее.
3. На этой закладке задаются опции отображения и ввода списковых значений в поля, а также опции защиты информации в поле – например сохранения в поле цифровой подписи или запрещения редактирования данного поля всем персонам с уровнем доступа в ACL ниже, чем Editor.
4. Параметры стиля для текста
5. Параметры стиля для позиционирования поля влево-вправо, интервала между сроками и т.д.
6. Параметры скрытия поля (в режиме чтения, редактирования, от какой-либо роли)
7. Параметры при генерации HTML для доступа с web

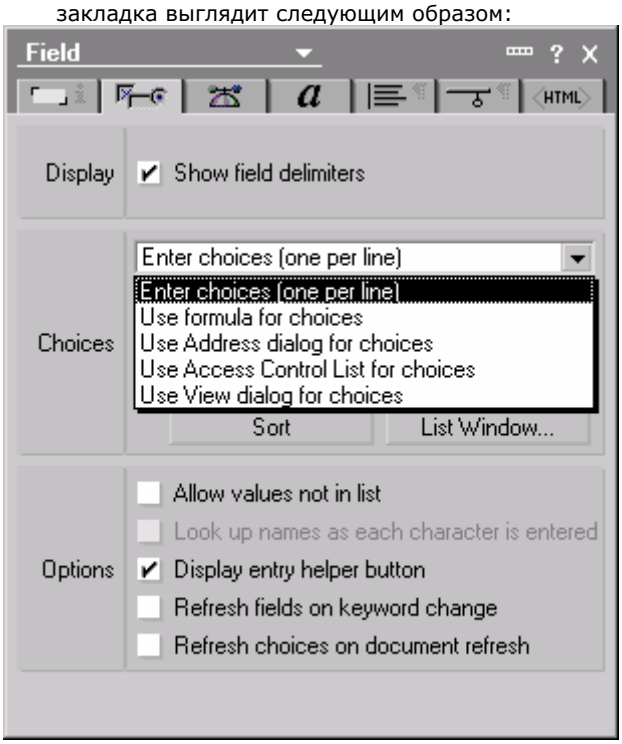
Типы полей



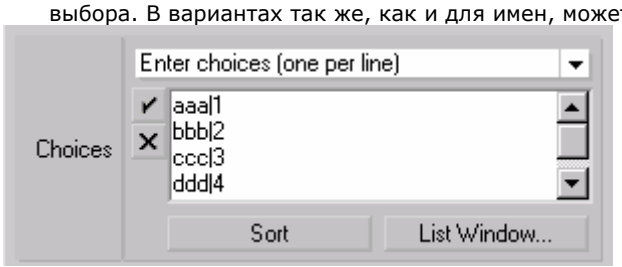
- Text – поле, хранящее текст. Сразу бОООООльшое предупреждение: размер текста хранимого в этом поле – максимум 32 килобайта. Поэтому в случае, когда в текстовое поле насчитываются какие-либо значения из справочников… - всегда есть риск, что при разрастании справочника, поле переполнится, и строка либо обрежется, либо вообще поведет себя непредсказуемо. В каждом конкретном случае способ решения этой проблемы определяется индивидуально, как правило для ее решения привлекается написание кода, который раскидывает строку, на ЭН полей, создаваемых динамически из скрипта.
- Date/Time – в поле этого типа хранится дата. Если вы попытаетесь ввести в такое поле неподходящее к формату даты значение – то при сохранении документа Lotus будет некрасиво ругаться. На второй закладке задается информация о формате отображения даты.
- Number – в таком поле могут храниться любые числовые данные. Аналогичная ситуация при неправильном введении данных. На второй закладке задается информация о формате отображения числа.
- Dialog list – этот тип поля позволяет отображать окно с диалогом выбора значения при нажатии на кнопочку рядом с полем (кнопочка появляется при выборе такого типа поля):



Список, из которого происходит выбор – может быть задан вручную, может быть задан по формуле или выбираться из справочника. Информация о выборе задается на 2 закладке свойств. Надо сказать, что вид этой закладки зависит от того, какой тип выбран для поля. Для dialog list 2-ая закладка выглядит следующим образом:

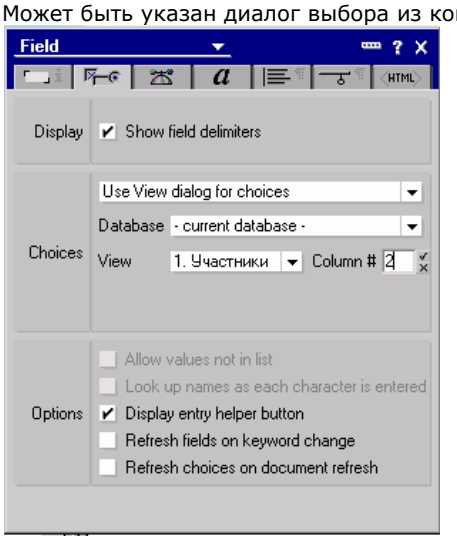


При включении первой опции – в окошке внизу просто перечислите через перевод строки варианты выбора. В вариантах так же, как и для имен, может использоваться алиасинг:



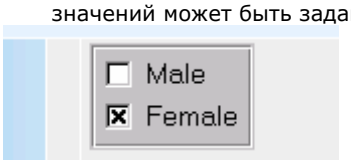
При использовании формулы для выбора – пишется код на языке формул, которых в результате возвращает строку или список строк.
Использование адресного диалога для выбора – рождает окно выбора значения из адресной книги (книга, в которой регистрируются пользователи, сервера, группы). В результате выбора в поле пропишется username выбранного пользователя или сервера или группы.

Аналогично для выбора информации о пользователях, серверах и группах может использоваться собственный Access Control List (ACL) базы.
Может быть указан диалог выбора из конкретного представления:



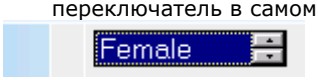
Указывается база, хранящая представление, имя представления и номер колонки, значением из которой – заполнять поле при выборе.

- Checkbox – поле, которое позволяет выделить несколько из списка возможных значений. Список значений может быть задан вручную или считаться по формуле.



А вот чтобы поле не было обведено таким ужасным четырехугольником, надо задать стиль отображения на 2-ой закладке в опции display -> borer style

- Radio button – аналог Checkbox, с ограничением выбора – не более одного значения. Аналогично, надо задать стиль отображения.
- List box – это также поле, позволяющее выбор из списка, только оно содержит весь список в себе, а не представляет его в виде диалогового окна. Ходить по элементам списка можно, щелкая на переключатель в самом поле.

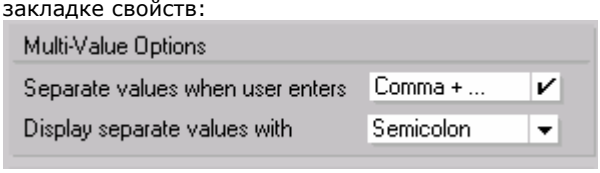


- Combobox – выпадающее вниз из поля окошко с вариантами выбора, оно отличается не только способом визуализации, но и тем, что в нем нельзя выбрать более одного значения, даже при включении на 1-ой закладке опции Allow multiple values, которая в нормальной ситуации позволяет хранить в поле не одну строку, а список строк и соответственно влияет на все диалоги выбора значений – делая в них multiselection
- RichText – о, это супер поле, в нем можно хранить до 2 гигабайт бинарных данных, объекты, атачменты, импортировать в него картинки, РТФ текст , в общем все, что в голову взбредет. Но не надо сильно усердствовать – создавая в базе много полей такого типа – они утяжеляют и тормозят базу. Но это поле самое трудное и неизмеримо сложное в обработке!!! Поэтому будьте с ним очень внимательны!
- Authors – это поле имеет отношение к системе раздачи прав. Именно в полях такого типа хранится информация о персонах, имеющих права на редактирование документа, в котором находится это поле. Фактически это поле хранит набор строк, каждая из которых – имя пользователя, сервера, группы или роли.
- Names – поле, для хранения имен. В Lotus имя пользователя является сложной иерархической структурой, которая имеет массу дополнительных параметров кроме самого имени. Вот чтобы это имя не отображалось в виде строки CN=Alena S Fedoseeva/OU=lialialia/O=SoftAriA (есть еще масса параметров, которые я здесь не перечисляю) и используют этот тип поля, так как хранит он в себе полное имя, а отображает его в приемлемом для пользователя виде.
- Readers – поле аналогичное Authors, которое контролирует кто имеет доступ к документу на чтение. (Эти поля описаны в разделе – система раздачи прав доступа \Документ, там же описана логика конкатенации этих полей). Если поле такого типа содержит пустую строку, то к документу имеют доступ ВСЕ, обладающие доступом к базе от Reader и выше.
- Password – поле, которое при вводе отображает вместо текста – звездочки, а внутри хранит текст. Однако надо иметь в виду – что поле на форме – это лишь способ визуализации информации. Если обратиться к одноименному полю документа, то там пароль будет представлен в виде нормального текста. Аналогично этот текста можно увидеть из окна со свойствами документа. Для этого щелкните правой кнопкой мыши, стоя на документе в представлении, и выберите Document properties.

Откроется окно свойств документа, в нем на второй закладке список полей документа. Встав на поле, в правой части окна вы увидите описание его типа, значения и других атрибутов.

Хранение списков

Как уже ранее говорилось, поля могут содержать массивы значений. Для того чтобы сделать поле «многозначным» используется опция на 1 закладке свойств «Allow multiple values». Эта опция включается не для всех типов полей. Вкупе с этой опцией идут опции ввода и отображения списковых значений. То есть - что является разделителем элементов списка при вводе данных и при отображении. Это включается на 3 закладке свойств:



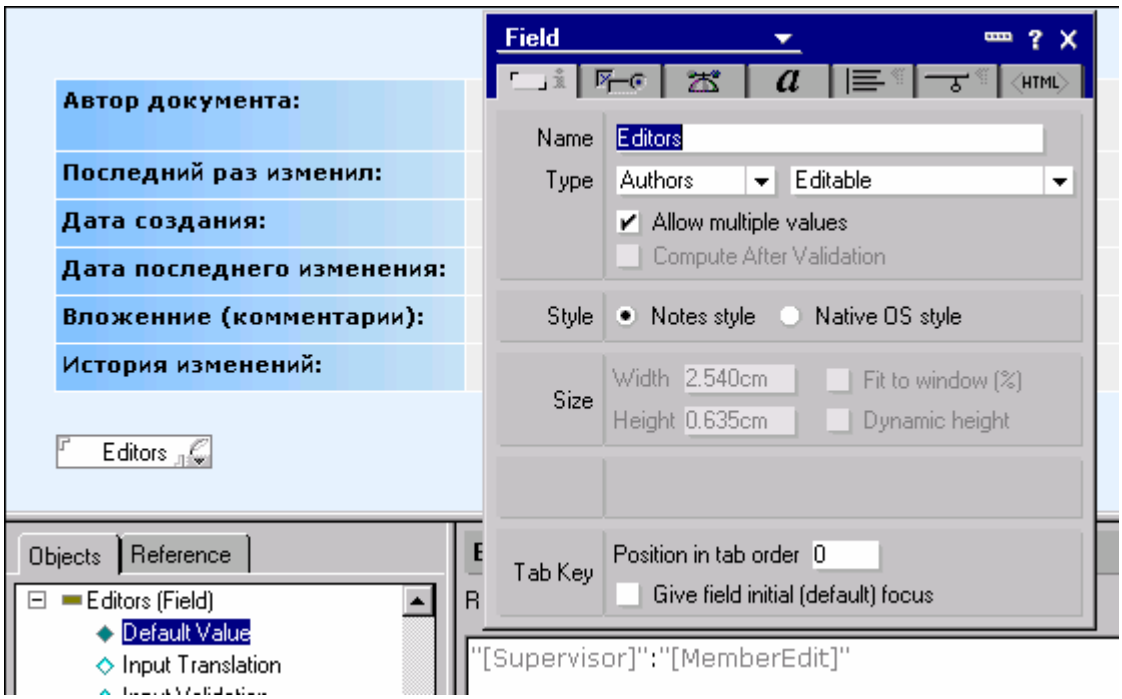
Списки могут состоять из строк, чисел, дат и т.д. О том, как эти значения задаются, читайте в следующем пункте и в разделе, посвященном программированию на языках формул и Lotus Script.

Значения полей по умолчанию

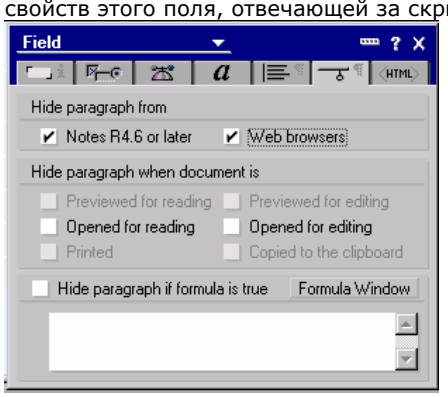
Когда курсор стоит на поле, то в нижней половине окна отображается список свойств и событий поля, а в правой – код и значения этих свойств и событий. Полю можно дать значение по умолчанию – это то значение, которое будет у поля при его инициализации (например, при открытии нового документа по форме, или при добавлении нового поля на форму и открытии по форме существующего документа или при создании документа из кода и пересчете его по форме.)
Итак, ввод значения по умолчанию делается следующим образом:

В приведенном примере, при создании документа в это поле посчитается имя пользователя, который создает документ. Используется вызов языка формул @UserName, который выдает имя текущего пользователя в сессии Lotus Notes.
Строка является частным случаем языка формул, и я могла написать в значении по умолчанию просто “ВАСЯ ПУПКИН” – в двойных кавычках. Если я кавычки не напишу, то строка будет интерпретироваться как название переменной.
Для полей, содержащих список значений, в качестве значения по умолчанию я могу задать список. Например, на форме, которую мы создали, необходимо ввести поле, которое разруливает права доступа на редактирование. Мы назовем поле editors, у него будет включена опция «Allow multiple values», тип поля будет Authors и значением по умолчанию будут роли, которые обладают правом на редактирование документа участника голосования – то есть “[MemberEdit]”:”[Supervisor]”. Обратите внимание, что названия ролей в поле пишутся в квадратных скобках. Списковые значения объединяются через двоеточие. И еще названия ролей очень чувствительны к регистру!

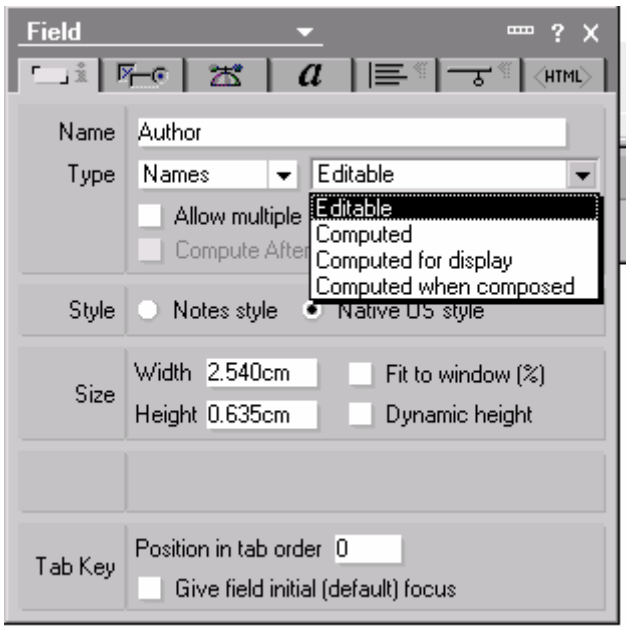
Кстати, если я впишу в качестве значения по умолчанию – список, а флаг многозначности не поставлю – то после сохранения документа – все элементы списка «слипнутся» в одну строку. Ошибка достаточно распространенная – будьте внимательны.



А еще это поле – оно не является частью интерфейса пользователя и его надо скрыть, то есть в закладке свойств этого поля, отвечающей за скрытие, будут стоять опции скрытия от Notes клиента и от web.



Типы редактируемости и вычисляемости полей

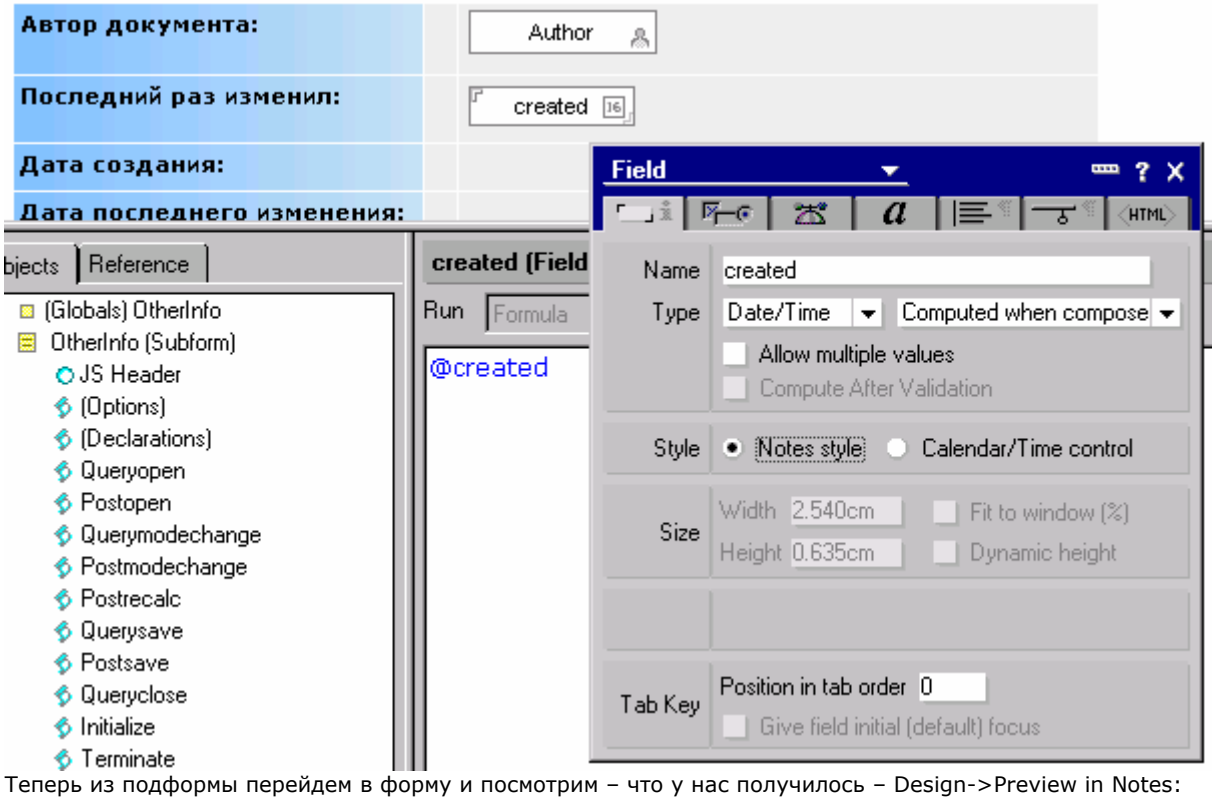


Самый простой вид полей – это поля, которые редактируются. В них вводятся значения, и они имеют тип Editable. В нашем случае – на нашей подформе поле, в котором хранится автор документа, должно показывать автора, но его (автора) никто редактировать не должен. Такие поля, которые информацию показывают, но редактировать не дают – это один из типов Computed *. Их 3 вида:

1. Computed – это вычисляемое по формуле поле, которое будет вычисляться каждый раз при открытии и пересчете документа. **Это поле сохраняется в физическом документе**. То есть, если я сделаю поле с автором документа - этого типа и поставлю формулу @UserName, то при каждом редактировании и пересчете в этом поле будет прописано имя пользователя, который в данный момент открыл этот документ. Для Васи это будет Вася, а для Пети – Петя. Каждый увидит в этом поле себя. Нас это устроит – нет.
2. Computed for display – это поле аналогично предыдущему за одним исключением – оно существует **только при открытии документа**. Оно не сохраняется в физическом документе. Такие поля бывают нужны, когда мы создаем интерфейс, в котором разным ролям доступны на редактирование разные куски формы. Тогда для каждого элемента информации мы создаем 2 поля – одно актуальное редактируемое поле, в которое эта информация вносится той ролью, которая может ее редактировать. Второе поле – типа Computed for display, которое показывает эту информацию тем ролям, которые не могут редактировать актуальное поле. И на эти поля установлены противофазные формулы скрытия, то есть могу редактировать – вижу поле А, не вижу Б, не могу редактировать – вижу Б и не вижу А. В нашей ситуации это поле также не подходит.
3. Computed when compose – Это поле которое вычисляется 1 раз – при инициализации. На физическом документе оно сохраняется. То есть при создании документа в него посчитается автор и так в нем и останется. Это-то поле нам и надо для автора. И для даты создания, кстати, тоже, так как пересчитывать ее каждый раз не имеет смысла.

Стили и размеры полей – просто попробуйте поиграть с ними сами. Самые нижние свойства первой закладки отвечают за положение поля в порядке обхода. То есть при нажатии клавиши Tab курсор будет перескакивать в следующее поле в порядке обхода. Порядок задается просто цифрами 1,2,3,4,5,6,7…. В одно из полей я могу поместить курсор по умолчанию.

Продолжим дизайн нашей подформы.
Поле – дата создания документа – в значении по умолчанию мы используем формулу, выдающую дату создания @created:

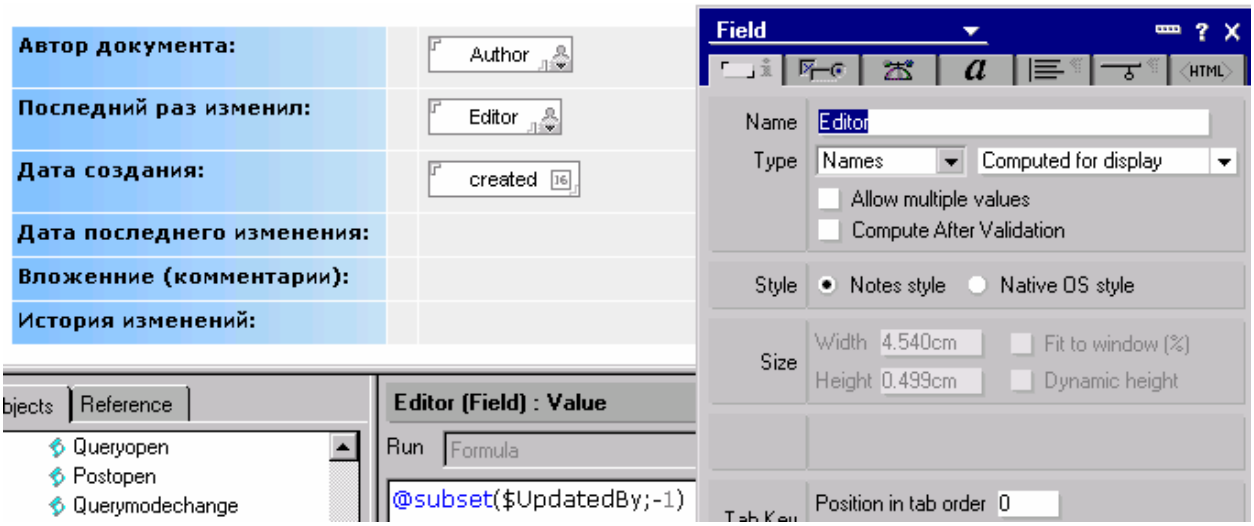


Теперь из подформы перейдем в форму и посмотрим – что у нас получилось – Design->Preview in Notes:

Но мне не очень нравятся вычисляемые поля в рамках, поэтому я сделаю их не Native OS Style, а Notes style. Кроме того, надо это поле перенести на строку вниз ;)) Это можно сделать, выделив его, и нажать CntrlX-CntrlV

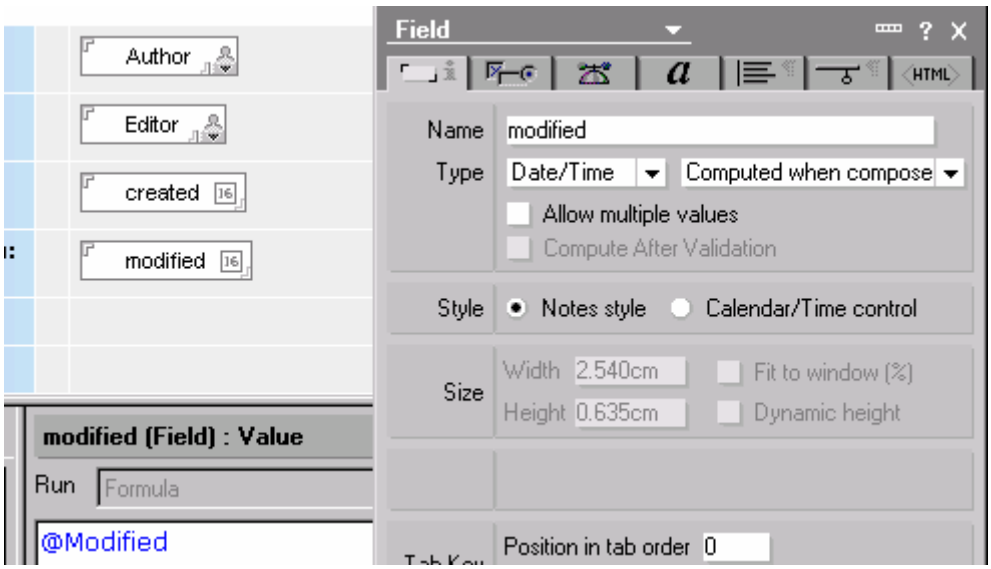
Далее я создаю поля для того, кто последний раз изменил документ и даты последнего изменения.

В документе существует поле $UpdatedBy, в котором в хронологическом порядке сохраняются все когда-либо редактировавшие этот документ. То есть последний в этом списке и есть – последний редактировавший. Очевидно, что значение этого поля считается каждый раз заново, так как список пополняется время от времени. А вот смысла перегружать документ информацией из него же взятой – нет, поэтому поле будет типа Computed for display – то есть не будет сохраняться в документе.
Итак:



Для подсчета значения поля используется формула, которая возвращает указанное количество элементов списка, переданного в качестве первого параметра. -1 означает возврат одного элемента списка с конца. В качестве списка передано поле $UpdatedBy. При обращении к полю по имени, в формулу реально при подсчете подставляется его значение.

Следующее поле – дата последнего изменения, она будет computed for display, и вычисляться по формуле @modified.



Следующее поле будет редактируемым. Мы должны дать пользователям возможность вкладывать в него различные объекты. Поэтому его тип будет Rich text. Вложение объектов делается из меню File->Attach (Import), но не все пользователи знают о такой возможности, поэтому прямо рядом с полем мы сделаем 2 кнопочки, которые помогут пользователям с вложением объектов.

Итак, поле – rich text, editable.

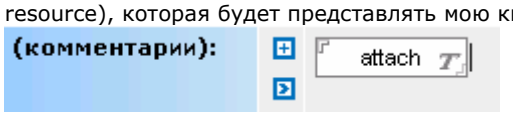
6.1.8. Hotspots

Для создания кнопки – поставьте курсор в то место где хотите ее отобразить, и далее в меню Create-> Hotspot->button.
Получится довольно некрасивая серая кнопка, смотреть на нее тошно. Поэтому я предлагаю сделать ее же, но в красивом виде.

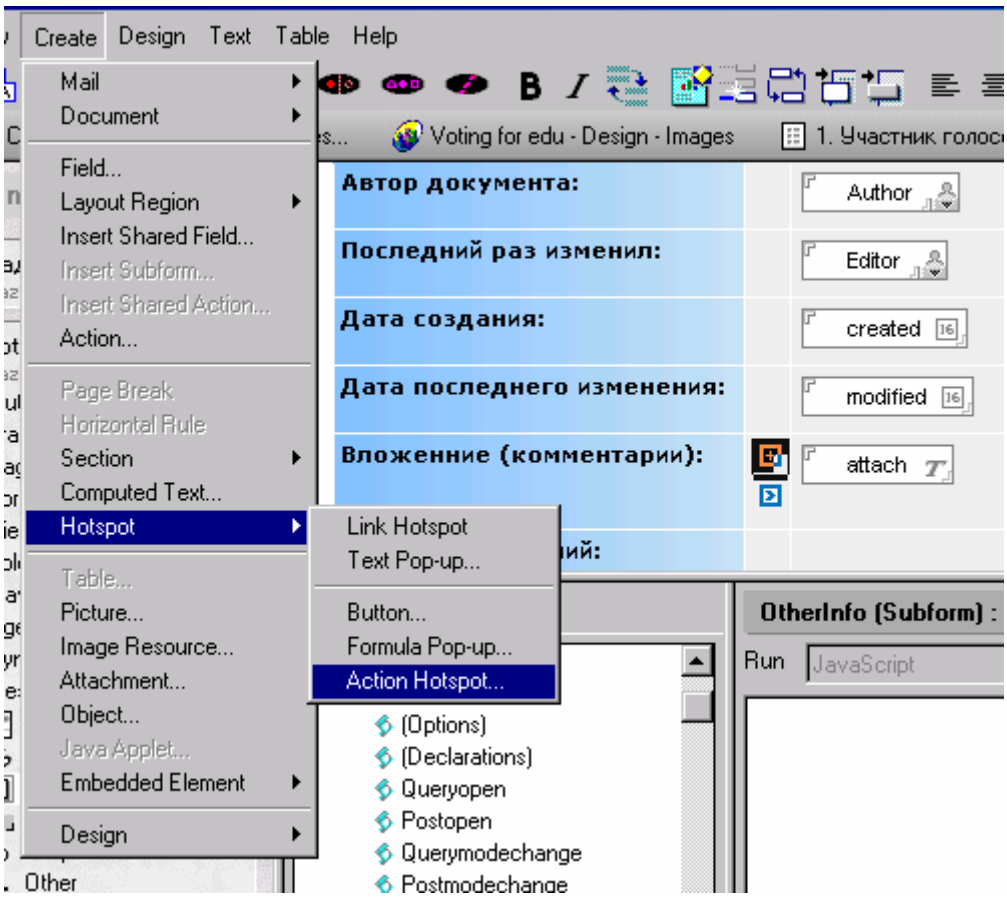
Action hotspot

Для этого используется в том же меню Create->Hotspot->Action hotspot. Это объект, который можно повесить на текст или картинку. К этому объекту привязан исполняемый код.

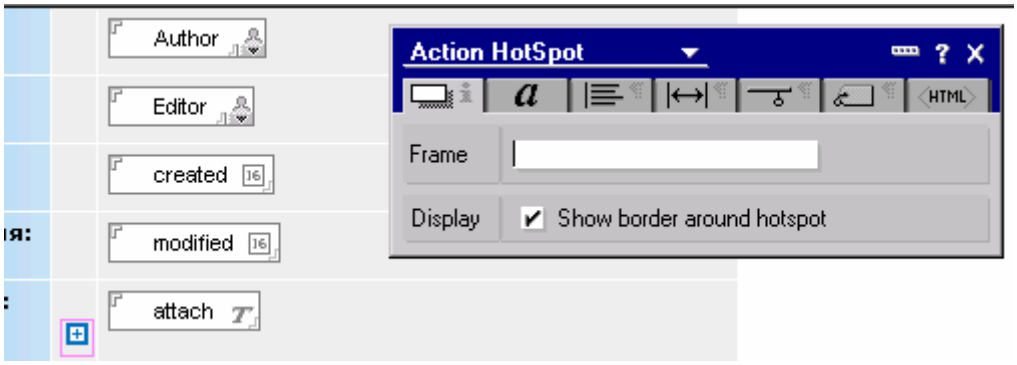
Теперь по шагам. Сначала я напишу текст или из библиотеки картинок загружу картинку (Create -> Image resource), которая будет представлять мою кнопку.



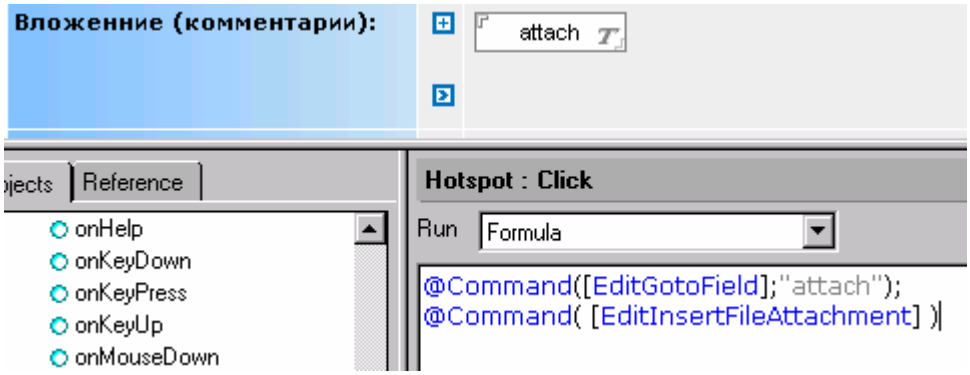
Потом выделю ее мышкой и создам на выделении action hotspot. Потом, поставив курсор на объект с hotspot, в меню я загружу Hotspot properties.
На вышеприведенной картинке у нас 2 кнопки – плюс – для присоединения и стрелка для импорта. С каждой картинкой мы работаем отдельно. Начнем с присоединения:



Теперь на картинку «одет» hotspot со специфичными свойствами:



Надо снять галочку с Show border around… - чтобы убрать этот розовый квадрат.
Остальные закладки свойств регулируют стиль отображения.
В нижнем правом окне задается код, который выполняется по факту щелчка мышкой на Hotspot, здесь может использоваться как язык формул, так и Lotus script & simple action – для программирования под lotus, а также Java script для программирования под web. Сейчас я использую команду из языка формул, которая откроет диалог выбора файла и вложит выбранный, в указанное мной РТФ поле, на которое первая команда установит курсор:



Для импорта файла также создается hotspot, только используется другая команда:
@Command([EditGotoField];"attach");
@Command([FileImport])

Теперь на закладке параметров скрытия свойств этих hotspots надо указать, что они не видны в режиме чтения, и открытия на чтение, так как мы можем редактировать поле только в режиме редактирования, и использование этих кнопочек в режиме чтения выдаст ошибку.

Теперь перейдите в форму и сделайте ее preview в клиенте Notes. Понажимайте на кнопочки.

Бегло опишу - какие еще бывают hotspot-ы:

Button

Это обычная серая кнопка – у нее можно задать параметры отображения, подпись, скрытие и т.п. Кнопка программируется на языке формул, lotus script, simple action или java script

Link hotspot

Позволяет сделать какой-то элемент формы ссылкой – на URL, на один из элементов дизайна базы и т.п.

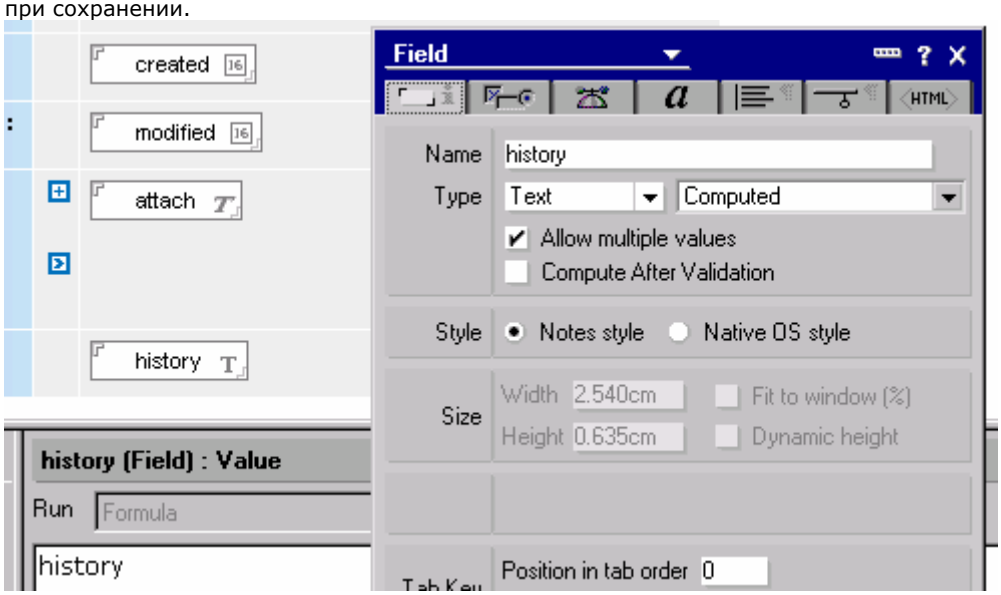
Text Pop-up

При наведении мышки на элемент формы, на который «одет» hotspot такого вида всплывает текстовая подсказка

Formula Pop-up

Формульное всплывающее окно позволяет посчитать текст в окне по формуле.

Теперь мы создадим на подформе вычисляемое поле история изменений. Это поле будет вычисляемым. Так как историю нельзя давать редактировать. Но считаться она будет не от каких-либо параметров документа или среды окружения, а только от самой себя. Поэтому в формулу для этого поля мы поставим название его самого. А модифицировать его значения будут какие-либо изменения статусов или внесения комментариев при сохранении.



При этом каждая запись в историю – будет отдельным элементом списка значений этого поля. Поэтому в нем мы включаем галочку многозначности. И так как мы ее включили – то сразу надо указать опции отображения на 3 закладке. В них я указала, что история должна отображаться через перевод строки.

Итак, с подформой мы закончили. Теперь перейдем к форме.

Сразу надо сказать, что с точки зрения удобства и выработанных стандартов восприятия документов необходимо его разбить на зоны – как минимум зона заголовка и зона основного тела документа.

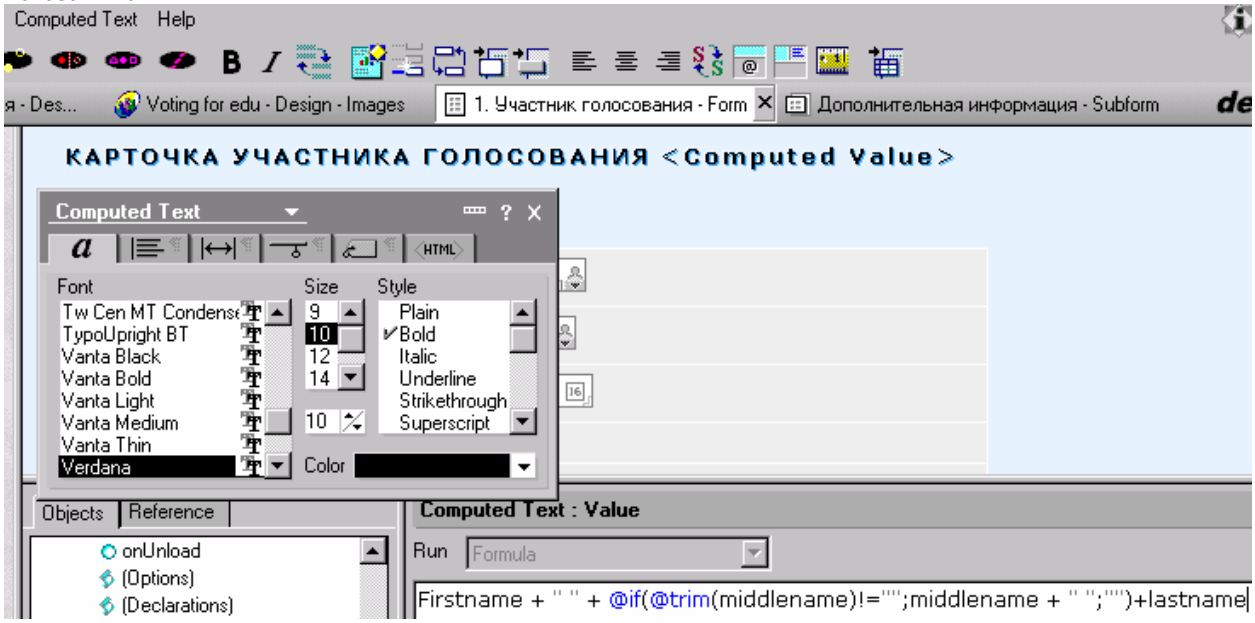
При этом очень желательно, чтобы зона основного тела зрительно умещалась на одну страницу и не требовала скроллинга. Можно разбить основную зону на несколько закладок или секций. Закладки экономят вертикальное пространство тела формы.

Сейчас мы для нашей формы участника голосования создадим заголовок и таблицу с закладками. На первой закладке мы разместим всю бизнес логику формы, а на 2 закладке – нашу подформу с дополнительной информацией.

Заголовок я оформлю в виде простого текста – КАРТОЧКА УЧАСТНИКА ГОЛОСОВАНИЯ ВАСИ ПУПКИНА. Надо отметить, что часть такого текста является у меня статической, а часть – вычисляемой. Для вычисляемой части я могу использовать поле типа computed for display, которое будет считаться по значению полей Фамилия, Имя, Отчество.

6.1.9. Вычисляемый текст

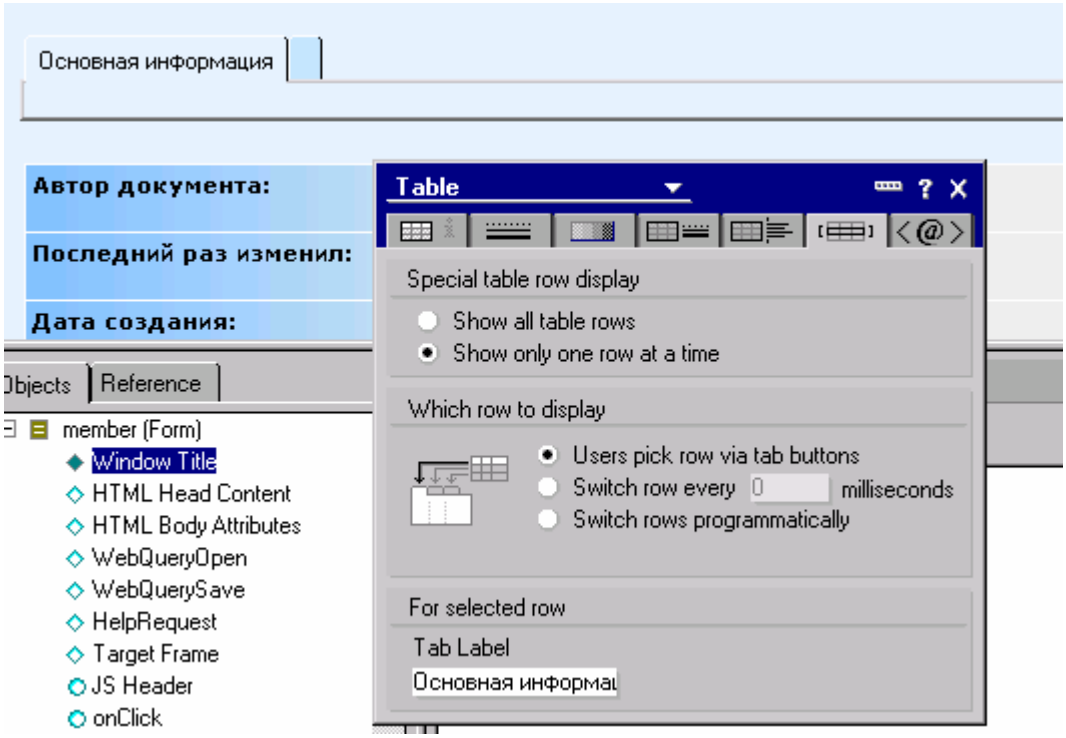
А могу использовать то, что называется Computed Text и по смыслу пересекается с полями вычисляемого для показа типа.



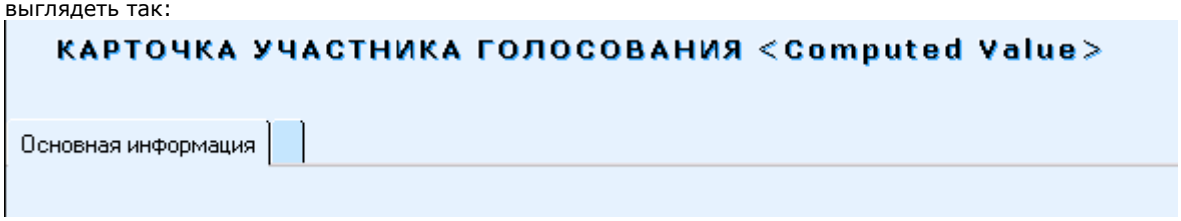
Это сущность со свойствами текста и формулой, по которой этот текст вычисляется. Здесь мы использовали формулу, которая взяла значения из поля FirstName, слила его с пробелом, потом если в поле MiddleName не пустая строка – то взяла его значение, добавила после него пробел и добавила значение поля Lastname. При этом мы сделаем поля FirstName & Lastname – обязательными для ввода в них информации.

Что такой текст создать – меню Create-> Computed text.

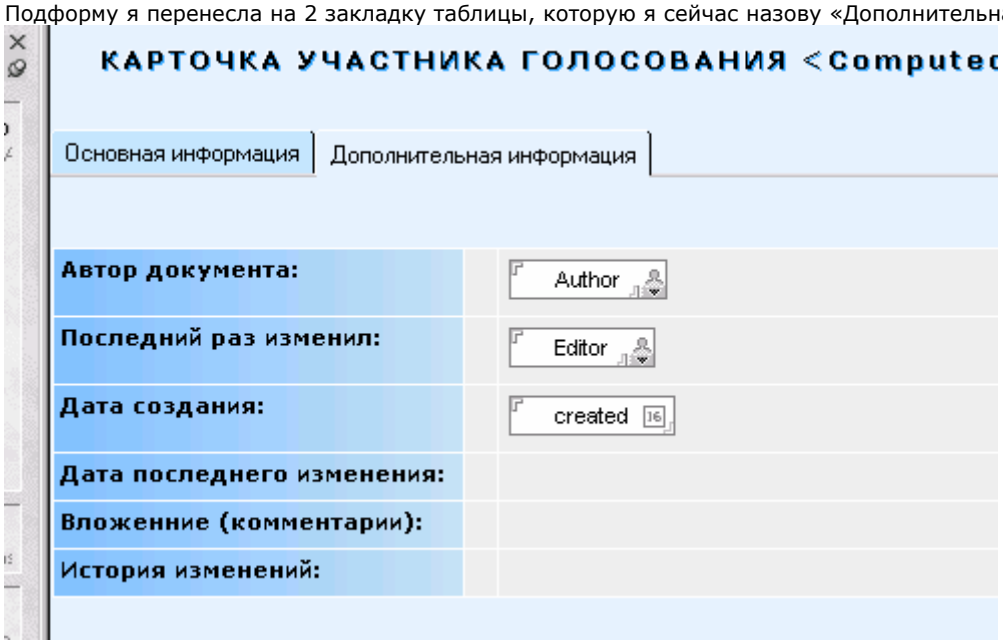
Заголовок мы создали, теперь создадим под заголовком таблицу с закладками. Обратите внимания, что в такой таблице каждая закладка – это строка таблицы. Я сделаю таблицу с 2 строками и 1 столбцом. Меню create->Table.



На предпоследней закладке свойств таблицы – мы указываем текстовую метку для закладки. По сути одна закладка - это одна ячейка для таблицы нашего вида – так как у ней 1 колонка. Я хочу, чтобы у моей таблицы не была видна граница. Вы свою отредактируйте, как вам хочется. Я пройдусь по всем закладкам и на 2 закладке свойств таблицы – для каждой ячейки проставлю 0-ую ширину границы. В итоге она будет выглядеть так:



Я еще и указала на 1 закладке свойств таблицы Fit to Window
Подформу я перенесла на 2 закладку таблицы, которую я сейчас назову «Дополнительная информация».



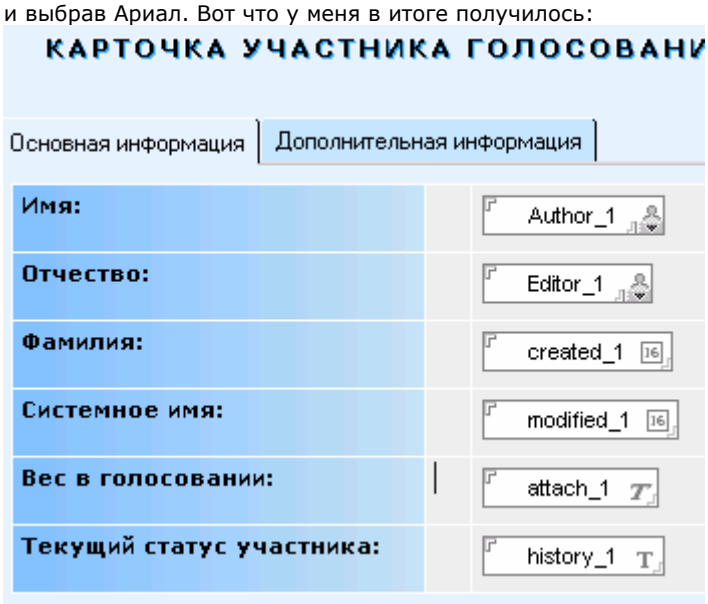
Мне не нравится нулевой отступ таблицы в моей таблице с закладками от левого края экрана. Я отрегулирую это свойствами таблицы с закладками – на 1 закладке свойств таблицы – параметры отступа.

Верхний отступ мне тоже не нравится – я его уменьшу, отрегулировав размер текста и видимость текста на подформе – того который прямо перед таблицей – и в дальнейшем везде буду придерживаться этого стиля.
Вот что у меня получилось:

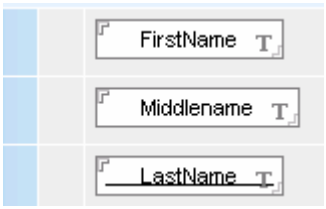
Теперь я перехожу в основную закладку формы. Для сохранения стиля я скопирую таблицу из подформы и вставлю ее на 1-ую закладку формы, далее я изменю все подписи к полям и добавлю (удалю) строки таблицы до необходимого количества.

Обратите внимание – при вставке таблицы в таблицу с закладками - система задаст вам 2 вопроса – первый – копировать ли – на него надо ответить «ДА» и второй – при копировании – вставлять в логику существующей таблицы – на него надо ответить НЕТ. Тогда новая таблица вставится в ячейку существующей, иначе же – ну просто попробуйте что будет – описывать это бесполезно ☺. Вам это пригодиться – когда вы будете переносить данные одного куска таблицы в другой – причем таблицы будут идентичной топологии – тогда вы сможете копировать содержимое ячеек в соответствующие ячейки.

Далее я отформатировала верхний отступ моей «голубенькой» таблицы от закладки – уменьшив шрифт до 4 и выбрав Ариал. Вот что у меня в итоге получилось:



Теперь будем переименовывать поля и перепрописывать свойства.
Имена первых 3 полей у меня уже предопределены – формулой на вычисляемом тексте. Поэтому сейчас я введу их названия, поставлю им свойства – редактируемый текст, без флага многозначности.



Теперь я сделаю то, о чем говорила ранее – контроль ввода значений для полей имени и фамилии.

6.1.10. Проверка значений полей (валидация) и трансляция

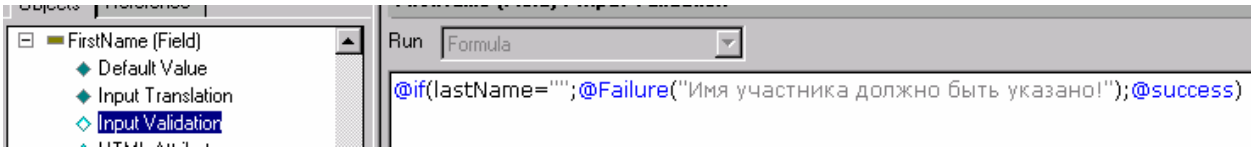
Когда вы ставите курсор на поле и поле редактируемое (список событий и свойств полей отличается для редактируемых и вычисляемых) – то в нижнем левом окне кроме значения по умолчанию вы видите – Input Translation & Input Validation (для полей имени человека не забудьте удалить значения по умолчанию оставшиеся от старых полей).

Первое событие – трансляция происходит при сохранении документа и его пересчете – здесь вы можете что-то сделать со значением, введенным в поле, и результат вашей операции запишется в поле при сохранении или пересчете. Ну, например, обрезать с начала и конца пробелы @trim(FirstName), @trim(Lastname) & @Trim(MiddleName) или наоборот присоединить префикс и постфикс.

Событие валидации – это проверка значения поля на корректность. Здесь если проверка не прошла успешно вы можете выдать предупреждение и сохранение или пересчет остановятся.

У полей есть связанная с этой опцией свойство на 1-ой закладке свойств – compute after validation – эта опция включает вычисление значения поля только после того как «провалидированы» значения других полей формы.

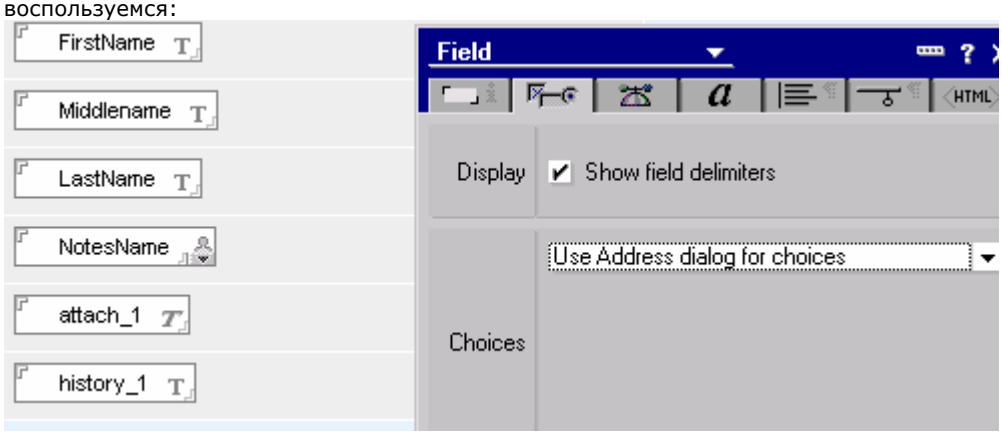
Теперь перейдем к практике – я прописала в трансляцию обрезание пробелов на конце и в начале введенного значения, а на событие валидации я впишу следующую формулу:



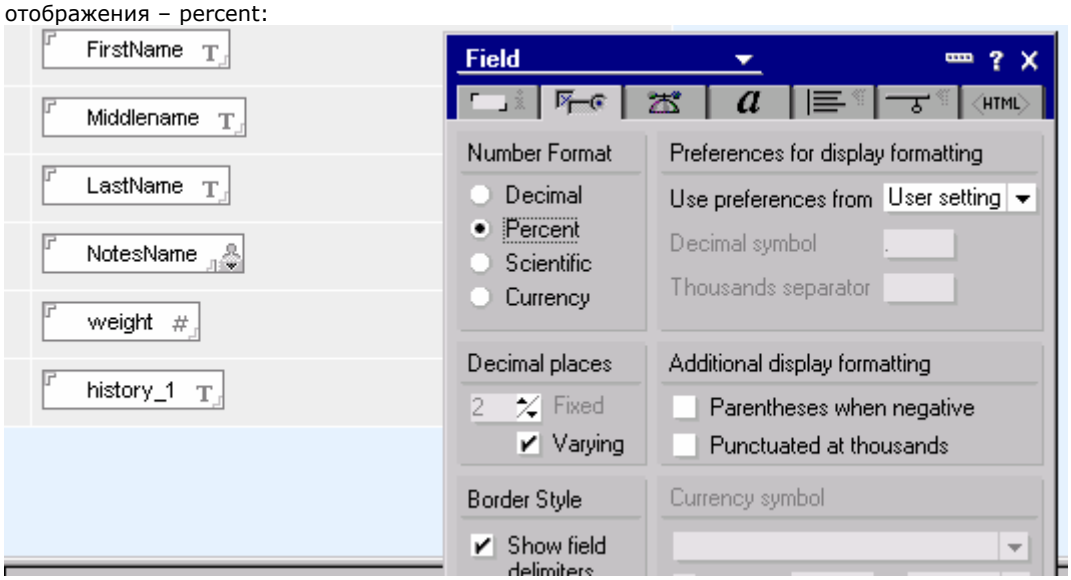
Формула состоит из @if(условия; что делать, если условие выполнено; что делать, если условие не выполнено), все параметры и вызов формулы от вызова отделяются точкой с запятой.
Мы проверяем – а не пусто ли наше поле и если пусто – вызываем @Failure – которая просто выдает сообщение и возвращает не успешное выполнение команды. А если значение поля не пусто – то выполнится команда @success, которая даст знать системе, что валидация прошла успешно.

Теперь внесите аналогичный код на поле LastName. И исправьте ошибку, которая была на картинке в имени поля – FirstName, а не LastName. (Я специально вношу такие опечатки, так как исправление ошибок в документе – запоминается и способствует лучшему восприятию и запоминанию ☺)

Теперь продолжаем работать над следующими полями. Поле с именем, зарегистрированным в системе – это Names, редактируемое, у этого поля есть опция включения диалога из адресной книги – ей мы и воспользуемся:



И включите на 2 закладке опцию display entry helper button. Теперь сделайте preview формы в клиенте и попробуйте выбрать в это поле значение из адресной книги.
Далее поле Вес в голосовании – это числовое редактируемое поле, которое содержит информацию о проценте акций, принадлежащих голосующему. Поэтому на второй закладке свойств поля я указываю тип отображения – percent:



Следующее поле – статус и оно не редактируется вручную – для его редактирования мы сделаем 2 кнопки. В принципе можно было бы дать и пользователю это поле переключать – но это значимая операция, поэтому ее хочется выделить как нечто значимое. Посему мы сделаем на форме кнопки.

Думаю, поле статус вы сделаете сами – не редактируемый Radio button с вариантами выбора «Черновик», «Активный», «Неактивный» - мы будем его переключать из кода кнопок. Значение по умолчанию «Черновик». Так как поле вычисляется само от себя, то значение по умолчанию задается @if(status="";"Черновик";status)

Мы можем создать кнопки на форме – в виде hotspot-button, а можем вынести их на панель кнопок вверху формы. На панель выносят кнопки, действие которых относится к документу в целом. В нашем случае – вопрос - куда их поместить. Это дело вкуса – можно рассмотреть работу со статусом – как работу с документом в целом, а можно рассмотреть изменение статуса как работу с полем СТАТУС. Мне больше нравится вынести кнопки наверх.

6.2. Кнопки

Итак, я создам кнопки. Сразу, как только я решала создать кнопку, возникает вопрос – куда ее положить – на конкретную форму? Или в библиотеку кнопок? Логика проста – если она относится только к этой конкретной форме и не может быть переиспользована – то ей место на форме. Хотя можно все равно вынести ее в библиотеку – если это удобно и декларируется как общее место хранения всех кнопок при разработке. Но если кнопка переиспользуется – то НЕЛЬЗЯ хранить ее на форме. (Более того, когда кнопка хранится в библиотеке, и используется в большом количестве форм, то вы тем самым экономите в размерах базы данных, В самом деле, совсем плохо, когда один и тот же код листов на 10, будет заполнять половину базы данных).

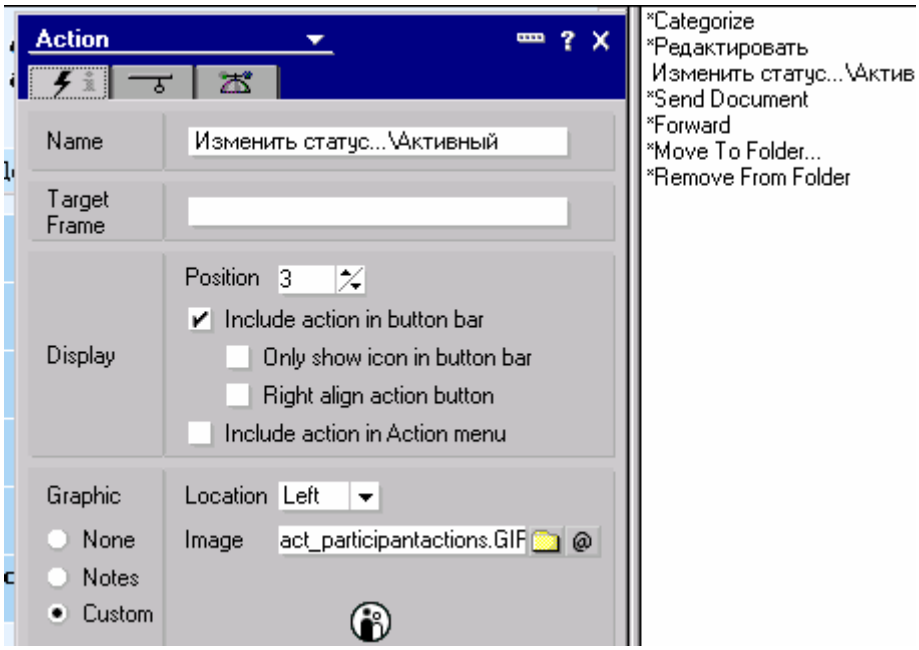
В данном конкретном случае – изменение статуса участника – это приватное дело формы участника, поэтому для демонстрации такой возможности я создам кнопки изменения статуса – на форме.

Итак, я перехожу в правое верхнее окошко (если оно у вас закрыто – просто сдвиньте его границу). Поставьте курсор на список кнопок, потом в меню create -> action.

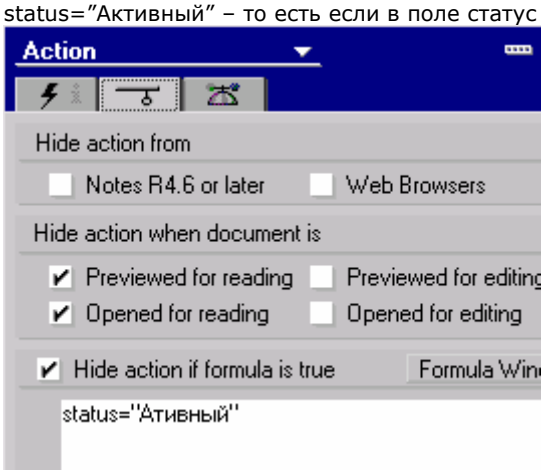
У вас в списке кнопок появится еще одна кнопка (Untitled) и откроется окно ее свойств, которое можно открыть, просто 2 раза по ней щелкнув мышкой.

Я задаю имя кнопки (если использовать слеш – то кнопка будет открываться каскадом), номер ее позиции в списке кнопок, формулы скрытия, открытия и свойства отображения рисунков на кнопке.

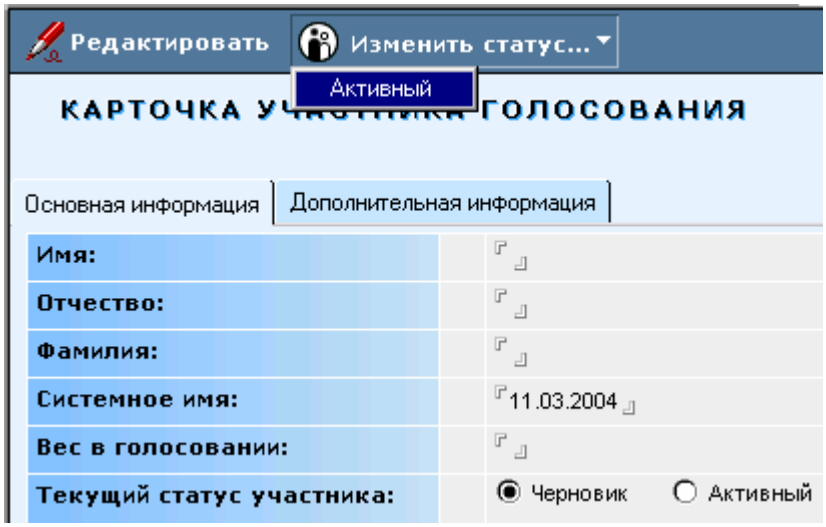
Набор стандартных иконок Lotus – хорош, но не стилен ☺, поэтому я внесу в библиотеку картинок иконки для кнопок из дизайна стандартного почтового ящика, и буду использовать на кнопках их. (Внести новые иконки можно, просто открыв ваш почтовый ящик в дизайнере и скопировав их оттуда)



Вторая закладка с формулами скрытия. Нам надо подумать – в каких ситуациях наша кнопка имеет смысл. Можем мы изменить поле документа в режиме чтения? Нет! Значит, скрываем на чтение и preview на чтение. Если у нас документ и так в состоянии «Активный» - нужна нам кнопка перехода в это состояние? Нет! Значит, нам в окно формул надо внести формулу, при выполнении которой кнопка будет скрыта status=”Активный” – то есть если в поле статус стоит такая строка, то кнопка скроется.



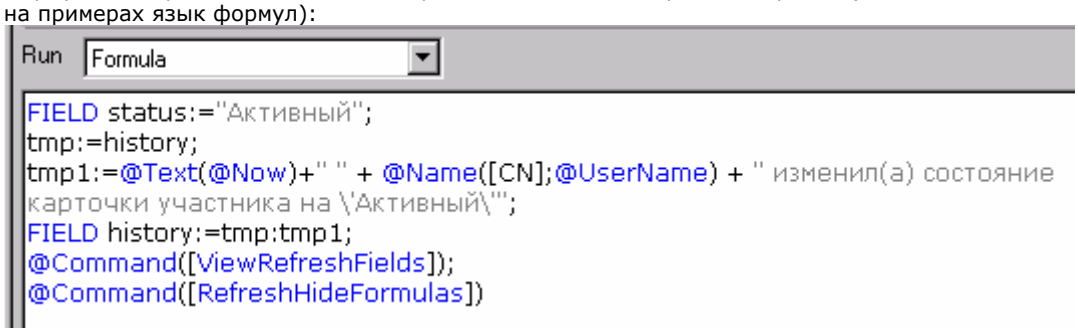
Надо опечатку поправить! ☺
Вот, что в итоге получилось:



Я забыла убрать в поле «системное имя» значение по умолчанию от старого поля, скопированного с подформы – поэтому получилась дата в этом поле. Надо убрать!

Теперь на кнопку надо «одеть» код. Он пишется в нижнем правом окне.
Что будет делать эта кнопка? Она будет изменять значение поля статус, добавлять новую запись в историю документа и обновлять документ на экране пользователя.

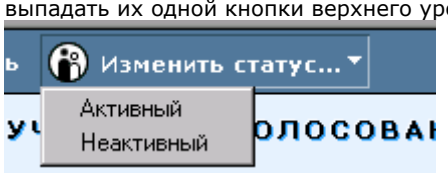
Формула, которая выполняет эти операции, выглядит следующим образом (кстати, мы с вами по ходу изучаем на примерах язык формул):



В первой строке полю status присваивается новое значение, для обозначения, что речь идет не о переменной, а о поле – при присвоении используется ключевое слово field. Присвоение значения делается оператором «:=».
Все операнды отделяются точкой с запятой.
В переменную tmp мы сохраняем текущее содержимое поля history.
В переменной tmp1 мы формируем строку для новой записи в историю. Строка состоит из сконвертированного в строку текущего дата\время, приведенного к приемлемому для восприятия виду имени текущего пользователя @Username и просто смысловой строки.
Далее в поле history (вспомните, что оно у нас многозначное – содержит списки) мы вносим список, состоящий из предыдущего значения этого поля и новой сформированной исторической записи.
Теперь мы просто обновляем документ, чтобы все изменения отобразились на экране.

Теперь сделайте Preview вашей формы с кнопкой и нажмите на нее – посмотрите на поле статус, на поле история и на видимость самой кнопки – которая скрылась в этом статусе.

Теперь по аналогии надо сделать кнопку для перевода в неактивный статус. Я думаю если все понятно по первой кнопке, то вторую вы сделаете элементарно – просто скопировав первую и изменив несколько строк в формулах скрытия и коде. Кстати – сохраните картинку – у вас кнопка каскадная – и обе кнопки будут выпадать их одной кнопки верхнего уровня:



Еще один момент! Когда запускается пересчет формы по факту нажатия кнопки – то происходит валидация формул полей и в написанном нами варианте формул валидации – при простом нажатии кнопки перехода из состояния в состояние – выскакивает окно о не введенном имени, фамилии (если они не введены). Такие предупреждения хорошо ограничивать только событием сохранения документа. Как это делается:

Есть ряд формул, которые возвращают true при происхождении некоторого события, например @Isdocbeingedited – знаменует, что документ редактируется, или @IsdocBeingSaved- заменует, что происходит сохранение документа. Это как раз то, что нам надо – в условии проверки заполнения полей в формуле валидации надо вставить также условие проверки того, что документ находится в процессе сохранения.
@IsDocBeingSaved & @trim(LastName)="";

6.2.1. Библиотека кнопок

Вообще на каждом документе должны присутствовать действия основной работы с документами – Сохранение, Отказ от сохранения, переход в режим редактирования.

Если проанализировать, то переход в режим редактирования могут делать роли, которые обладают правом на редактирование этого документа. Кнопку закрытия окна документа – могут видеть все, кнопку сохранения все, кто перевел документ в режим редактирования. Кнопку отказа от сохранения – аналогично сохранению.

Очевидно, что кнопки видны в разных режимах:
1. Редактировать – в режиме чтения
2. Сохранить – в режиме редактирования
3. Закрыть в режиме чтения
4. Отменить в режиме редактирования.

Кнопку редактировать – можно сделать отдельно для каждой формы – если в нее каждый раз прописывать новую формулу скрытия для разных ролей. Но, мы можем ввести стандарт для хранения информации о списке редакторов документа – в нашем случае пусть все редакторы документа хранятся в поле Editors. Поэтому мы можем не делать 10 кнопок редактирования для 10 форм, а сделать одну, в которой формула скрытия будет выглядеть следующим образом:
Tmp1:=@trim(@replace(editors;@UsernamesList;""));
@if(@elements(editors)>@elements(tmp1);@false;@true)

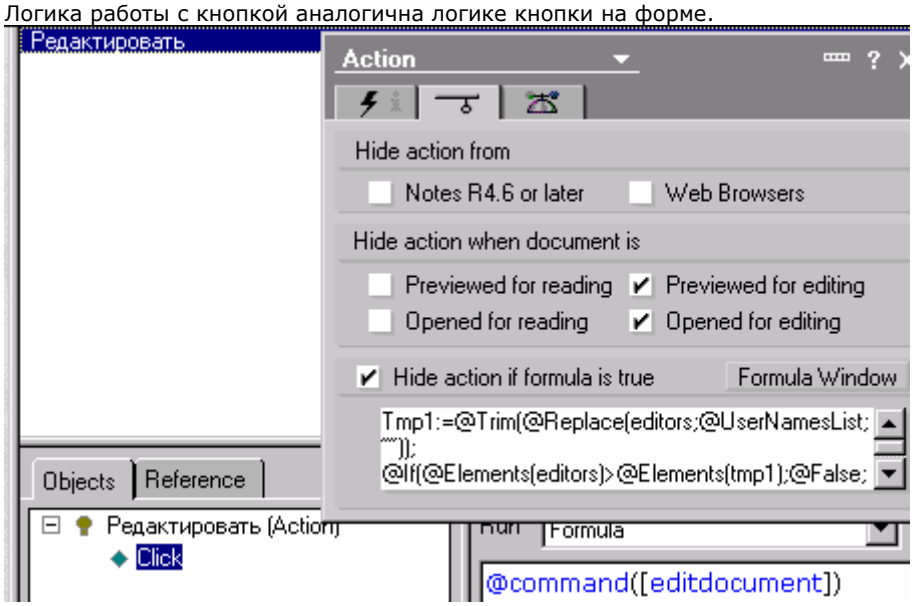
Сейчас я расскажу, что эта формула делает:

1. У нас есть функция, которая возвращает список всех ролей в текущей базе текущего пользователя и всех групп в адресной книге, в которые он входит и всех видов его имени - @userNamesList
2. В поле editors хранится список ролей, групп и имен, которые могут редактировать этот документ.
3. Нам надо понять – есть ли непустое пересечение этих списков
4. Для этого мы делаем Replace значений первого списка – вторым на пустые строки – то есть при нахождении в первом списке элемента, который есть также во втором – этот элемент заменится на «»
5. Далее на результате replace мы применяем @Trim, который на списковом параметре делает удаление всех пустых элементов списка.
6. Что мы имеем – если пересечение списков имело место, то кол-во элементов в исходном editors больше, чем количество элементов в результате @trim(@replace()). И тогда кнопка не скрывается, иначе мы ее скрываем

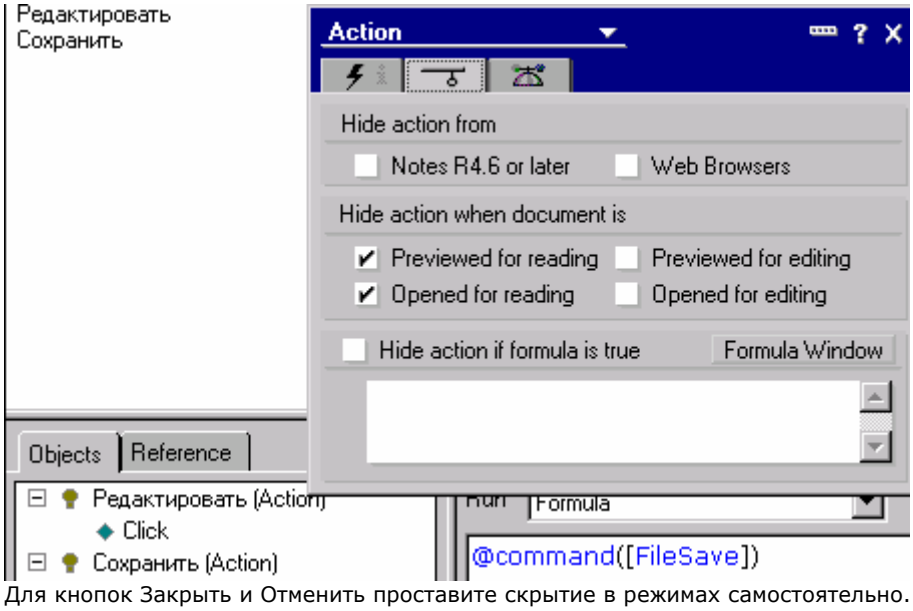
Всего 2 строчки и столько логики – весьма элегантно на мой взгляд ☺.

В общем, мы создаем 4 кнопки в библиотеке кнопок.
Зайдите в Shared actions – одна Untitled кнопка в библиотеке будет по умолчанию, остальные вы можете создать из меню create -> shared action.

Логика работы с кнопкой аналогична логике кнопки на форме.



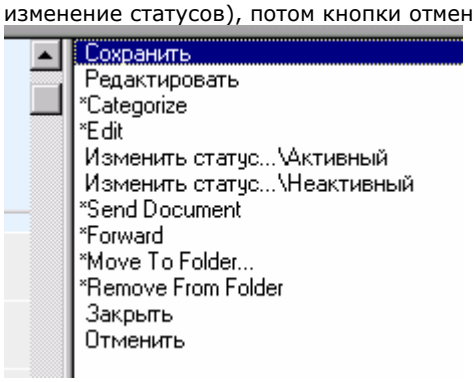
Аналогично я прописываю кнопки Сохранить, Закрыть, Отменить.



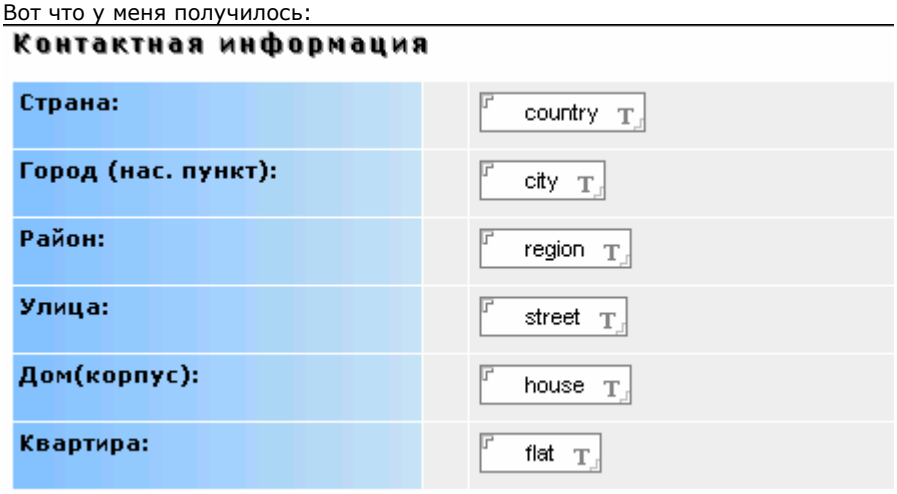
Для кнопок Закрыть и Отменить проставите скрытие в режимах самостоятельно. Команда для них одна - @command([FileCloseWindow]).

Теперь мы скроем нашу кнопку редактирования на форме и вставим наши кнопки из библиотеки.
Для этого перейдите в форму в окно с кнопками, поставьте в него курсор и из меню Create->Insert Shared action – добавьте все созданные нами кнопки. После этого укажите последовательность кнопок на панели.

Я поставлю первыми Сохранение и Редактирование, потом будут идти кнопки бизнес логики (в нашем случае изменение статусов), потом кнопки отмены и закрытия. Вот что у меня получилось:



Теперь откройте форму в клиенте, сохраните ее, потом откройте заново, перейдите в режим редактирования – поиграйтесь с тем, что получилось.
Далее, уже самостоятельно создайте подформу с контактной информацией и добавьте ее в 1ую закладку таблицы в форме участника. Я сделаю это в своей базе, которая прилагается к этому документу.
Вот что у меня получилось:



И теперь мы перейдем к дизайну представлений, который понадобиться нам далее.

6.3. Представления

На этом этапе дизайна у вас в базе есть только представление по умолчанию, в котором вы виде все свои документы просто пронумерованными в порядке создания в базе.

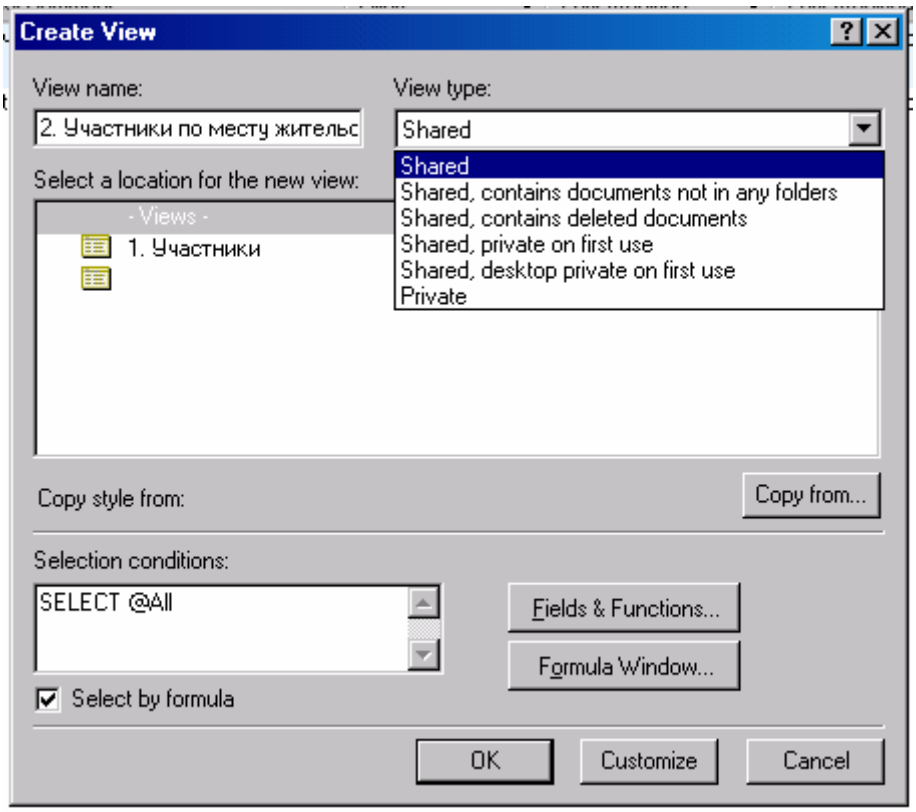
Сейчас моя цель сделать следующее – чтобы для полей страна и город – можно было выбрать одно из уже введенных значений (чтобы по 100 раз «Россия» не писать) или ввести новое.
А для этого мне потребуются представления уже существующей информации о введенных странах и городах.

В рамках этой задачи осветим построение представлений.

6.3.1. Линейные представления (вьюхи)

Что собой представляют вьюхи – сказано в самом начале документа, давайте вплотную займемся их конструированием.

В дизайнере перейдите в пункт views и там нажмите New View:



Это диалог создания вью, в котором задаются изначальные параметры – название, формула, по которой отбирать документы, здесь можно скопировать стиль уже существующей вью, а также задается тип вью. Какие бывают типы?

Типы вьюх

Итак:

1. Shared – это вью, которую видят все одинаково. Разумеется, в настройках доступа можно отрегулировать группы и роли, которые имеют и не имеют доступ к этой вью. Кроме того, в зависимости от прав доступа, разные люди будут видеть в этой вью разное количество и список документов. Однако если в такой вью, вы используете колонки для подсчета, например суммарных значений по различным категориям – то независимо от прав доступа человек будет видеть актуальные суммы всех документов по категории. То есть, даже если мне не доступна половина финансовых документов, в такой вью я увижу суммарные значения по всем документам. Этим эти вьюхи плохи.
2. Shared, contains documents not in any folder – такая вью на множестве документов, которые задаются формулой выбора задает подмножество тех, которые не лежат ни в одной из папок.
3. Shared, contains deleted documents – если в базе разрешено Soft deletion – это задается администратором на последней закладке свойств базы – то в такую вью будут попадать документы, которые были удалены и их оттуда можно восстановить.
4. Shared, private on first use – это общее представление, которое становится «приватным» - то есть видимым и принадлежащим только конкретному пользователю, после того как пользователь первый раз откроет это представлении. Что произойдет – создастся копия этого представления, принадлежащая пользователю. Такие представления удобно использовать для решения задач показа, например, документов созданных пользователем. Формула выбора будет Author=@UserName – для каждого пользователя, и чтобы представление каждый раз на сервере не пересчитывалось – его стоит сделать именно такого типа. В такое представление, например, не попадут финансовые документы, которые мне не доступны и я не увижу в суммах по категориям цифры, которые не должна видеть. У таких представлений есть и минусы – при изменении дизайна родительской вью на сервере, это никак не скажется на локальной копии, поэтому каждый раз при изменении дизайна вам придется оповещать всех пользователей вашей системы, чтобы они удалили старую вью и открыли ее заново. Место хранения приватного варианта вью определяется самим пользователем при ее создании
5. Shared, desktop private on first use – аналог предыдущего типа, только такая вью будет храниться на десктопе пользователя (desktop.dsk файл) и для ее удаления (чтобы получить новую версию дизайна) необходимо удалить с рабочей области иконку базы, подтвердив удаление приватных вью, и открыть базу заново.

6. Private – это представление доступное только самому пользователю. Он сам его создает, если ему не хватает стандартных представлений базы. При наличии прав в ACL на создание приватных вью – такое представление будет храниться в базе на сервере, если прав нет – то в desktop.dsk

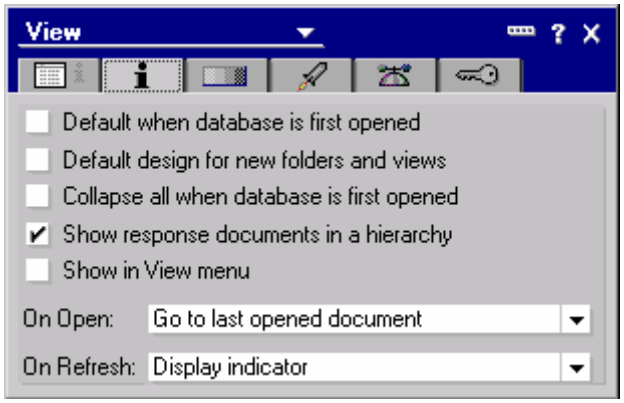
Какие бывают типы вьюх – пробежались.

Свойства вью

Вью создалась, теперь надо задать ее свойства. Для открытия окна свойств в меню Design->View Properties. На первой закладке указывается название вью и ее алиас, комментарии, а также основной вид вьюхи. Как выглядит стандартное представление, в котором линейно упорядочены документы, вы уже видели на картинке ранее. А есть еще календарные представления, когда у вас отображается календарь с датами и каждый документ привязан к какой-то дате, выглядит это следующим образом:

Можете просто зайти в свой почтовый ящик на Lotus, в календарную область (переключатель между областями – иконки слева внизу) и посмотреть, как это на практике работает.

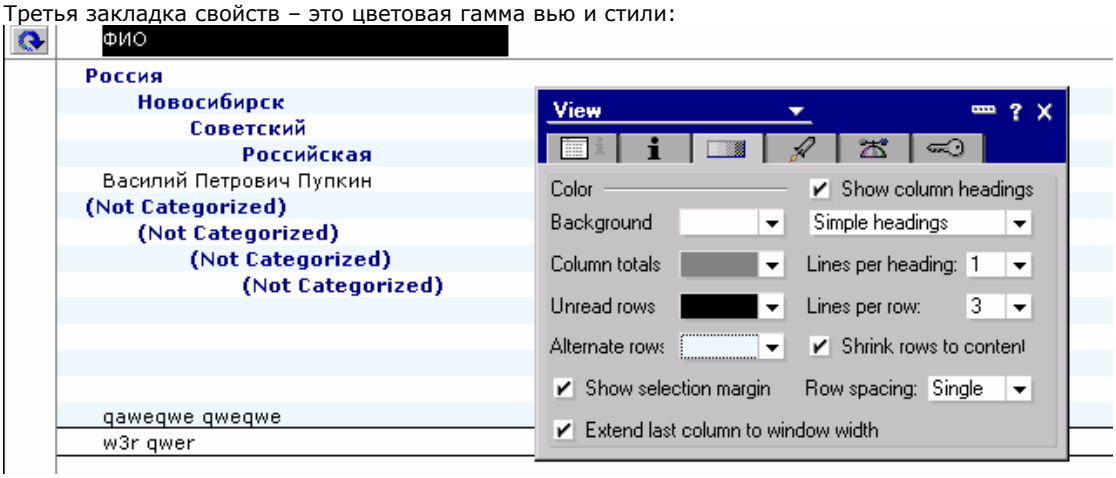
Далее, 2-ая закладка свойств:



1. Объявляет вью первой открываемой при открытии базы. Если не задан frameset, и на нем не указано открытие другой вьюхи, то будет открываться эта. В базе должна быть вью с таким атрибутом, в случае отсутствия оной бывают проблемы при внешнем обращении к базе.
2. Эта опция объявляет, что по умолчанию дизайн вновь создаваемых вью наследуется с этой вью.
3. Опция включает свертывание всех категорий при первом открытии базы. Эту опцию НЕЛЬЗЯ включать для вьюх, которые используются в диалогах выбора и являются категоризованными. Так как, если в диалоге выбора включить опцию выбора из конкретной категории, а категории при этом свернуты, то в окне выбора будет фига с маслом, вместо документов, причем начинающий разработчик может убить день и не понять в чем тут дело. Поэтому запомните этот факт пожалуйста.

4. Эта опция используется для вьюх, в которых отображаются не только документы, но и документы ответов (помните опцию свойств формы на 1 закладке, где указывалось, что это будет за документ – основной или документ ответа (например, для форумов))
5. Показ этой вьюхи в меню клиента View (очевидно только при открытии клиентом этой базы)
6. далее 2 опции – куда ставить курсор при открытии и что показывать для обновления

Третья закладка свойств – это цветовая гамма вью и стили:



Я буду описывать опции – сначала левую колонку сверху вниз, потом правую.
Итак, опции цвета:

1. Фон вью
2. Цвет тоталзов по колонкам (в самом низу вьюхи считаются общие тоталзы по всем колонкам, если для колонки включен подсчет по ней сумм)
3. Непрочитанные
4. Альтернативные – вот эта опция важна – так как видеть строки на абсолютно однородной по цвету простыне – плохо – информация плохо воспринимается. Для придания ей структуры – каждую вторую строку можно слегка подсветить другим цветом
5. Показ слева от всех колонок полосы, в которой документы можно пометить галочками и они становятся выбранными (с ними потом можно делать какие-то общие действия (да хотя бы копировать в буфер))
6. Можно включить, чтобы последняя колонка визуально растягивалась до конца экрана.

Вторая колонка – параметры отображения колонок и строк:

1. Показывать ли вообще заголовки колонок
2. Стиль этих заголовков – я вот включила «ПРОСТЫЕ заголовки», поэтому – вы можете прямо на картинке видеть – они стали прозрачными.
3. Количество строк в заголовке
4. Количество строк в строках вью, идет вкупе со следующим параметром, который включает сжатие до того количества строк, сколько актуально есть в отображаемом значении, ну пример, чтобы было понятно – у меня есть описание – в нем 35 слов, я его отображаю во вью. В 1 строку оно у меня не влазит, так как длинное, тогда я включаю в этой опции, что для одного документа, значение его поля во вьюхи отображать на «до 5 строк», так что у меня строчка моей вьюхи станет толстой – на 5 строк текста. Но если я 2-ую опцию не включу – то для всех документов будет по 5 строк. А если включу, то для документов, у которых в поле описание меньше текста – будет лишь требуемое для отображения кол-во строк.
5. Расстояние между строками

Четвертая закладка – показ вью в отдельном фрэймсете, пятая – параметры доступа через web, шестая – опции кому показывать вью. Для вьюхи, которую я создаю по участникам в опциях доступа должны быть выбраны роли: [MemberCreate]:[MemberRead]:[MemberEdit]:[Reader]:[Supervisor]

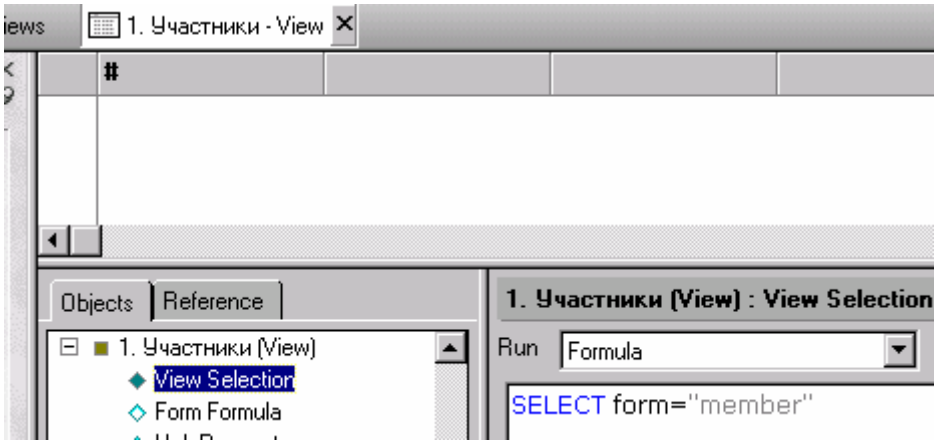
Если вы укажете стиль вью – календарь, то список свойств на закладках будет иным – там появятся настройки отображения календаря.

Формула выбора

В формуле выбора вы задаете множество документов, которые вы хотите видеть в этом представлении.
Формула выбора конструируется по следующей логике:
Ключевое слово SELECT + логический предикат, которому должны соответствовать документы.
Логический предикат задается на языке формул, если он верен для документа, то документ попадает во вью.

Например, чтобы отобрать все документ участников голосования я использую формулу select form=”member”, а вот если я захожу построить вьюху только по активным участникам, то select form=”member” & status=”Активный”

Вы можете задать формулу выбора при создании вью, и дальше ее изменить в свойствах вью View Selection в нижнем левом окне (открыв вью в дизайнере).
Моя вьюха сейчас выглядит следующем образом:



Колонки на вью и их свойства

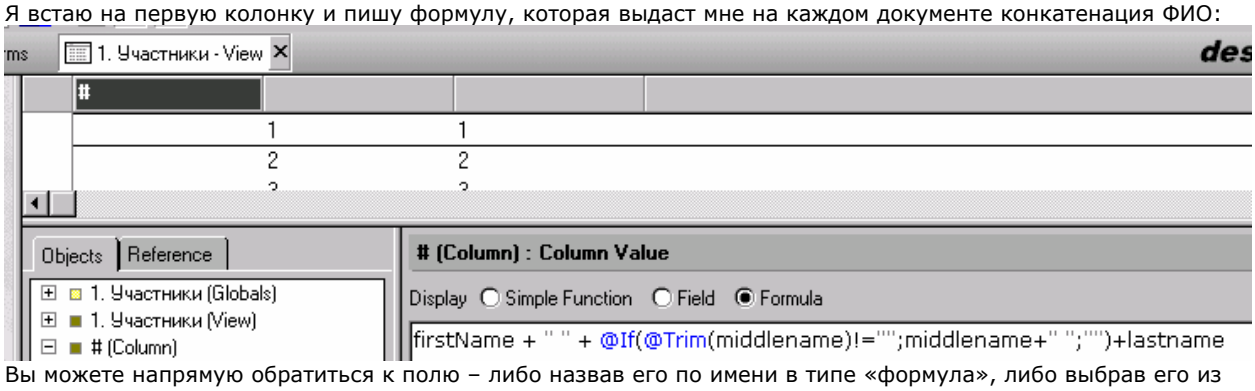
Если вы не задали стиля вью (то есть исходной вьюхи, откуда этот стиль копировать – скопируются колонки, их формулы, фонты и пр.), а сейчас вы не могли его задать, так как в базе нет уже отдизайненых вьюх, то у вас будет одна колонка. Чтобы отредактировать список колонок вьюхи - на той одной колонке, которая есть – кликните правой кнопкой мыши – там есть опции вставки колонки, добавления, удаления.

Создайте колонки, которые вам нужны. У меня во вьюхе будут колонки для:

1. ФИО
2. Веса в голосовании
3. Контактной информации

Теперь на созданных вами колонках надо указать – а что в каждой колонке будет отображаться.
Для этого поставьте курсор на колонку – в нижнем левом окне загрузится значение этой колонки – в значение пишется обычная формула, которая будет посчитана на каждом документе и будет выдавать некоторое значение. Значение может быть строкой, числом или датой. Есть еще колонки специального вида – они отображают иконки и картинки, об этом будет сказано позднее.

Я встаю на первую колонку и пишу формулу, которая выдаст мне на каждом документе конкатенация ФИО:



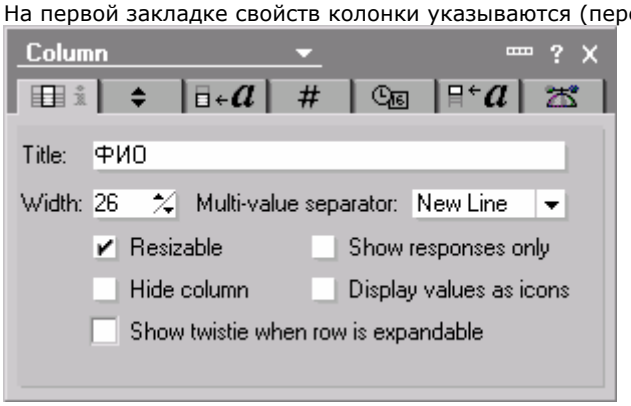
Вы можете напрямую обратиться к полю – либо назвав его по имени в типе «формула», либо выбрав его из списка в типе «поле». Есть еще набор Simple function, которые выдают на документе некоторые его атрибуты, такие как дата создания. Дата модификации, размер и т.д.

Теперь в следующей колонке я поставлю формулу, которая отобразит числовое значение поля weight в виде числа с процентами.
Для этого я в колонке выберу ее значением поле weight, а потом вызову свойства колонки (2 раза кликнув по ней мышкой) и на закладке со значком # чекну свойство Percentage (value*100)

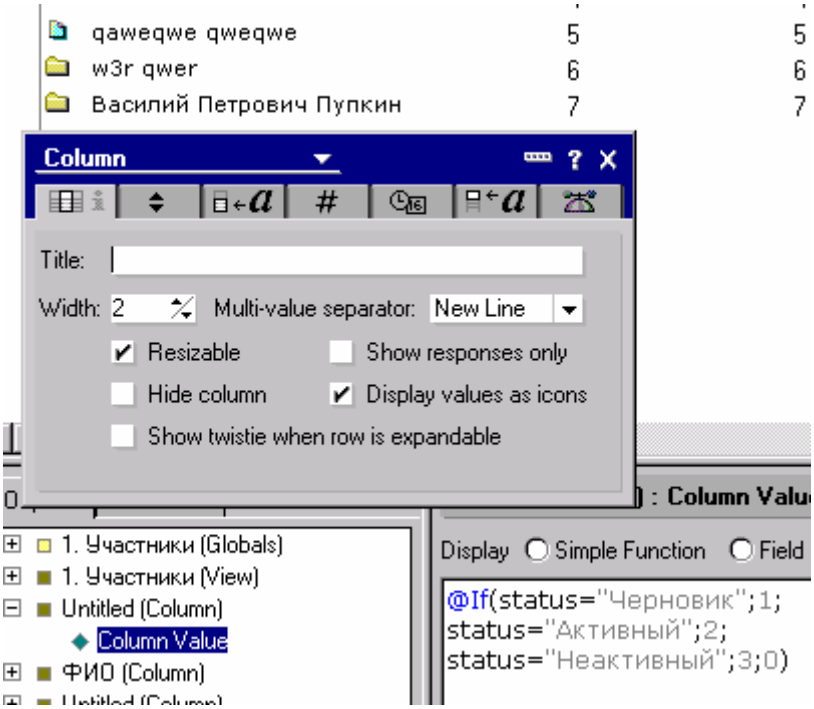
На свойствах колонки задаются свойства отображения данных в этой колонке для различных типов этих данных – на закладках 3-6, на 7 закладке хранится информация об уникальном имени этой колонки и параметры генерации ссылок на документы из значений этой колонки при просмотре через web.

Будьте внимательны с уникальным именем колонки!
При копировании и вставке колонки иногда Notes сам не уникализует имя и тогда, если у вас есть 2 колонки с одним именем, то вторая – даже если вы зададите для нее другую формулу - будет показывать то же, что и первая. Это называется ГЛЮК, о нем просто надо знать. Лечится он самостоятельным изменением этого имени.

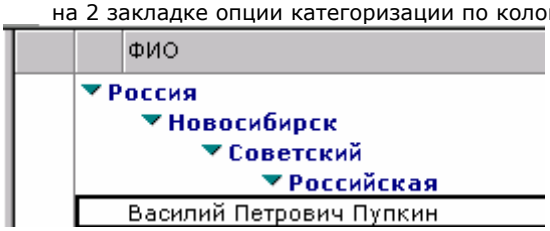
На первой закладке свойств колонки указываются (перечисление идет сверху вниз, слева направо):



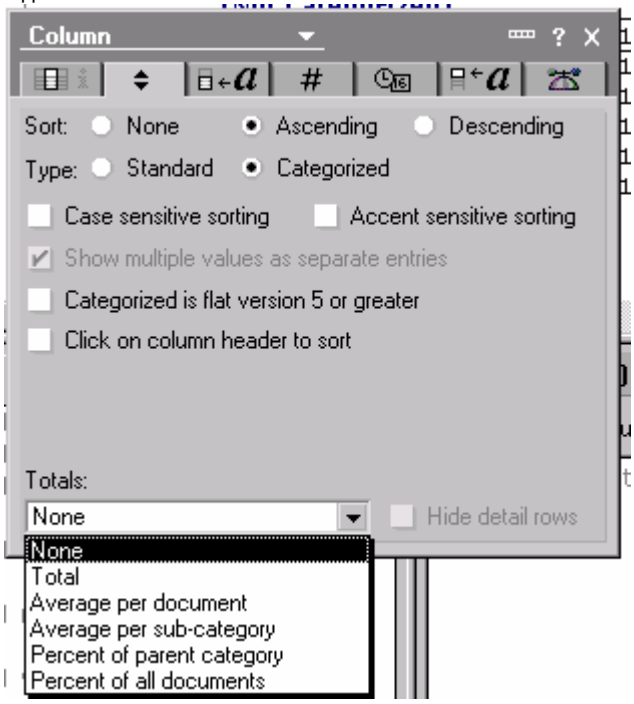
1. Заголовок колонки – та подпись, которая будет отображаться вверху
2. Ширина колонки
3. Если в колонке отображается поле со списком значений – то, что использовать при отображении в качестве разделителя элементов списка
4. Разрешение пользователю увеличивать и уменьшать размер колонки
5. Опция, используемая в иерархических вью, которая включает отображение в колонке именно документов-ответов.
6. Скрытие колонки – используется например, при введении сортировки по какому-то параметру документа, который не должен быть видим пользователю. Ну, например, у нас есть колонка с написанными прописью месяцами, и мы хотим их отсортировать последовательно. Строки названий при последовательной сортировке отсортируются не по логике времени, а по алфавиту. Как решить эту задачу? Надо перед видимой колонкой с месяцами-строками сделать скрытую колонку с номерами месяцев и поставить опцию сортировки по скрытой колонке.
7. Отображение значений иконками – эта опция позволяет отображать встроенные в Lotus иконки в строках вью. Например, у меня во вью отображаются участники в разных статусах – я могу пометить каждый статус иконкой. Иконки имеют номера и при подстановке в формулу номера иконки и включения этой опции, в колонке будет отображаться маленькая картинка. Есть возможность отображения произвольной графики в качестве иконок, это будет освящено далее.



8. Показ свертки/развертки для раскрывающейся категории – лучше показать наглядно, при включении на 2 закладке опции категоризации по колонке, эта опция включает «уголки»:

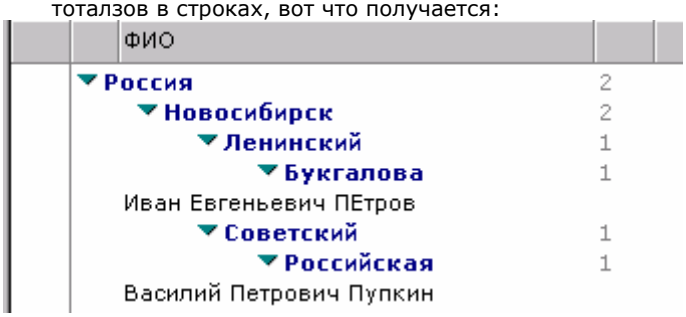


На второй закладке свойств задаются сортировки, категоризация, параметры суммирования и вычисления и т.д.



Итак, перечисляю сверху вниз и слева направо:

1. Параметр сортировки – ее может не быть, может сверху вниз или снизу вверх. Надо учитывать, что сортировка идет внутри того типа, которого эта колонка – то есть даты сортируются как даты, строки – по алфавиту, числовые значения – соответственно.
2. Тип – линейный список или категоризованный. Категоризованный - это значит, что все строки, которые имеют в этой колонке одинаковое значение – объединятся в группу, а группа будет свертываться и раскрываться как секция. На одной из предыдущих картинок, вью была категоризована по 1 колонке, в которую подставлялась информация об адресе.
3. Далее идут параметры сортировки – распознавать большие и маленькие буквы или accent sensitive – по-поводу 2-го параметра я не могу сказать конкретно, что это за сортировка – ни разу не пользовалась, при этом перевод «сортировка с учетом ударения» - звучит странно.
4. Отображение списковых значений как отдельных строк (если выключена эта опция, то они все отображаются в одной строчке), то есть один документ будет занимать не 1 строку, а столько, сколько значений в поле.
5. Далее параметр категоризации при версии клиента от 5 и выше
6. Включение опции сортировки для пользователя – если эта опция выбрана и далее указаны ее параметры, то на колонке вью появляются стрелочки, при щелчке мышью по которым эта колонка становится главной сортировочной колонкой вью и сортировка происходит по ней. Надо понимать, что у вью может быть N сортируемых колонок, тогда сортировка происходит сначала внутри первой, потом по второй и т.д.
7. В колонках с числовыми данными можно включить подсчет, например, сумм по категориям или других параметров (они перечислены). Можно также отключить показ данных в этой колонке для строк и оставить пока только для категорий. Например, если я хочу посчитать кол-во документов в категориях, я делаю колонку, у которой в значении колонки просто стоит 1 – то есть один документ – дает единицу, потом я задаю на колонке опцию отображения тоталзов и отключаю отображение тоталзов в строках, вот что получается:

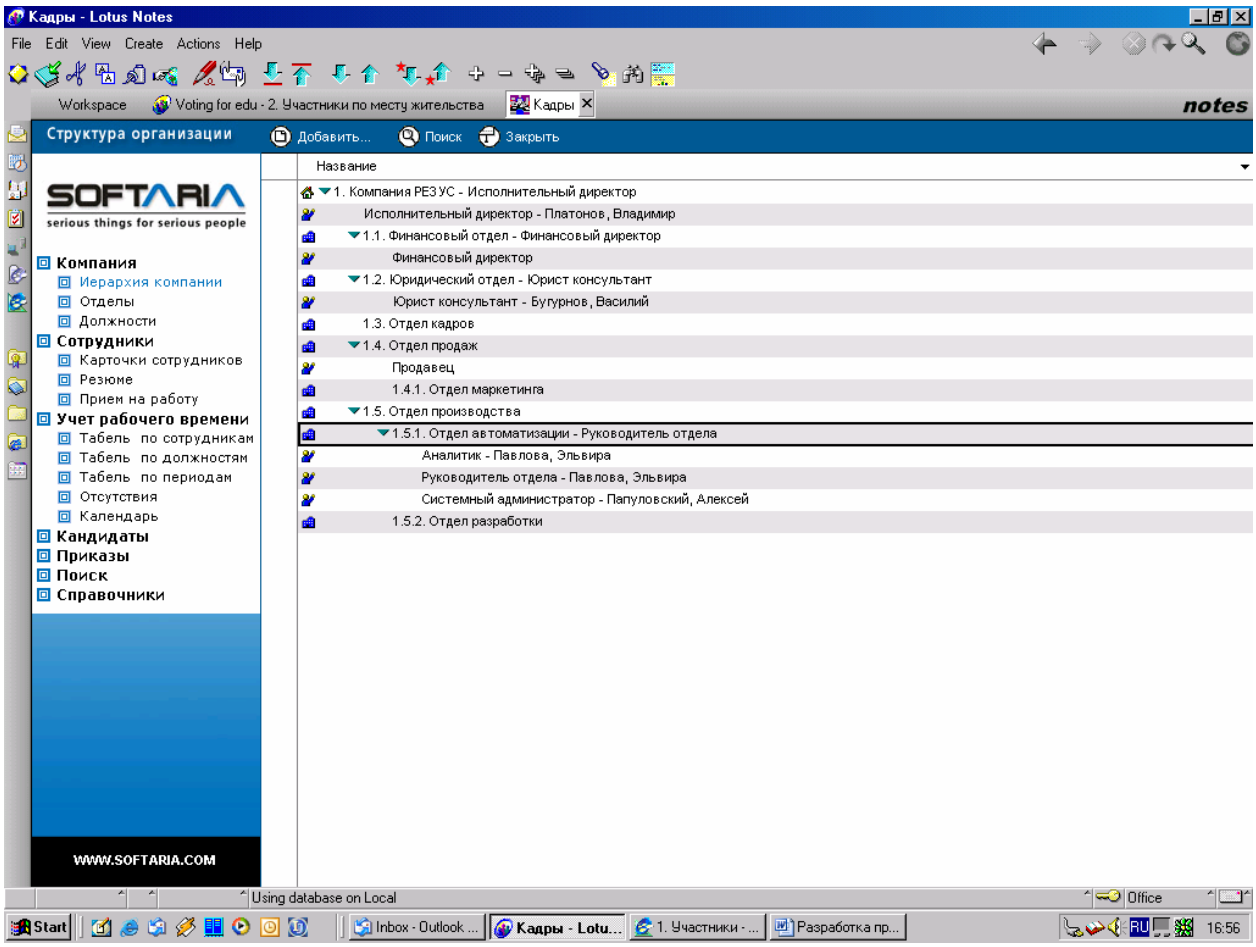


3-я закладка свойств – параметры фонтов в строках колонки
4-ая закладка – параметры отображения чисел
5-ая – дат
6-ая – отображение фонтов в заголовке колонки
7 –ая – уникальное имя колонки, параметры для генерации ссылок в колонке при просмотре вью через web.

6.3.2. *Иерархические вьюхи*

Иерархические вью, это вью, которые отображают иерархию документов. Для начала поймем – что такое иерархия и как она строится. В иерархии один документ является ответом на другой. У каждого документа (за исключением самых верхних) есть ровно один «родитель». Как в Lotus выстраивается отношение «родитель-ответ» между документами:
В документе ответа есть служебное поле $REF, в этом поле хранится UniversalID родителя. Причем это поле имеет не текстовый тип, а внутренний тип Lotus-а REFERENCE, то есть ссылка. Если вы строите вью, в которую выбираете документы родителей и ответов и указывается в свойствах вью, на 2-ой закладке Show response documents in hierarchy, то Lotus начинает отображать иерархию документов основываясь на связи, установленной через поле $REF. Когда вы устанавливаете тип формы Response или Response to Response – то при создании документа по этой форме, в него автоматически вносится это поле и в него прописывается UniversalID документа, на котором стоял курсор, при создании документа ответа.

Как выглядит иерархическая вью:



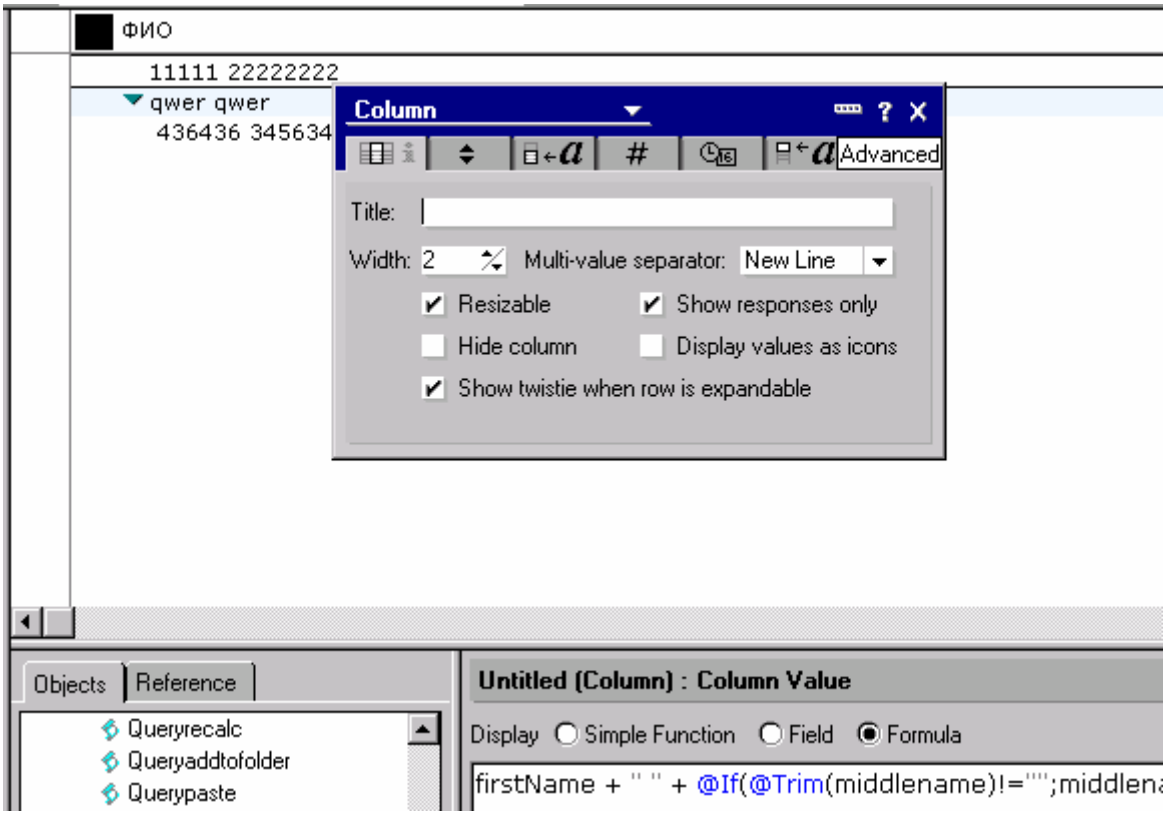
Рассмотрим пример построения иерархического представления:
Создайте новое представление, в нем у нас потом будут находиться документы голосований с бюллетенями в виде документов ответа к голосованиям.
Итак, проставьте свойства вью и на второй закладке галку отображения документов в виде иерархии.
Далее, я у себя в базе скопировала форму участника 2 раза. И переименовала 2 копии в Голосование и Бюллетень. На бюллетене я установила тип формы Response и включила опцию включения в меню Create.

Для примера я создала пару документов Голосование – открыв их методом action->preview in notes. Из дизайнера и сохранив.

В новой вью я для начала написала формулу выбора:
SELECT form="voting"

Мои документы попали в эту вью. Теперь стоя во вью на одном из документов я создала из меню create-> Бюллетень – пару документов бюллетеней. Однако, моя вью их не показала.
Почему? Потому что в формуле выбора не предусмотрен выбор этих документов.
Как мы можем включить документы ответов? Ну, например, сконструировав нашу формулу следующим способом: SELECT form="voting" | @isResponsedoc
Эта формула выберет документы по форме voting (это алиас формы Голосование) и документы ответов к выбранным.
Я могла бы также просто написать SELECT form*="voting":"bill". Эта формула означает - выбрать документы, у которых форма voting или bill. Вы можете использовать конструкцию «переменная*=список элементов», для упрощения записи выражения «переменная равна 1-ому элементу или переменная равна второму элементу или…»

Однако документов все равно не видно.
Теперь нам надо сделать колонку для их отображения. Эта колонка должна стоять перед колонкой, в которой отображается документ родитель и которая, вы хотите, чтобы открывалась как категория.
В ней должен быть выставлен флаг Show responses и в формуле для значения колонки – указано поле или формула для полей документов-ответов:



Здесь стоит такая формула для колонок, так как у меня сейчас голосование и бюллетень – точная копия формы участника, и я использую пока поля, которые есть в этих формах.

Далее на всех колонках, которые могут открываться и показывать документы ответов надо поставить Show twisty when row…
Если у нас иерархия 2-ух уровневая, как в примере, то это свойство на колонке с ответами – бессмысленно, но если иерархия может быть более 2-ух уровней, то эту опцию включить необходимо, чтобы видеть также документы ответов на ответы.

Включения категоризации нигде не требуется. В итоге мы получили вью, которой мы воспользуемся в дальнейшем, которая отображает иерархию документов.

Построение альтернативных иерархий

В реальной жизни один документ может являться участником разных иерархий – например, документ отдела может быть ответом на другой отдел и документом ответа на карточку руководителя отдела. Что тогда делать? Поле $REF. Хранит не более одной ссылки…

Все достаточно очевидно – строится иерархия по альтернативному полю. Например, в модели архитектуры объектов я указываю, что связь между объектами делается по полям HeadID & DeptID в документе отдела. По семантике именований полей понятно, что это ИД руководителя и вышестоящего отдела.

При создании документа отдела в нем указывается документ руководителя и документ вышестоящего отдела. В соответствующие поля вносятся UniversalID-ы этих документов. Как внести UniversalID не в виде строки, а в виде внутреннего типа Reference – это отдельная песня. Она будет спета в части этого документа, посвященной Lotus script. Сейчас мы просто представим, что мы получили на документах отделов 2 поля с UniversalID-ами соответствующих родителей.

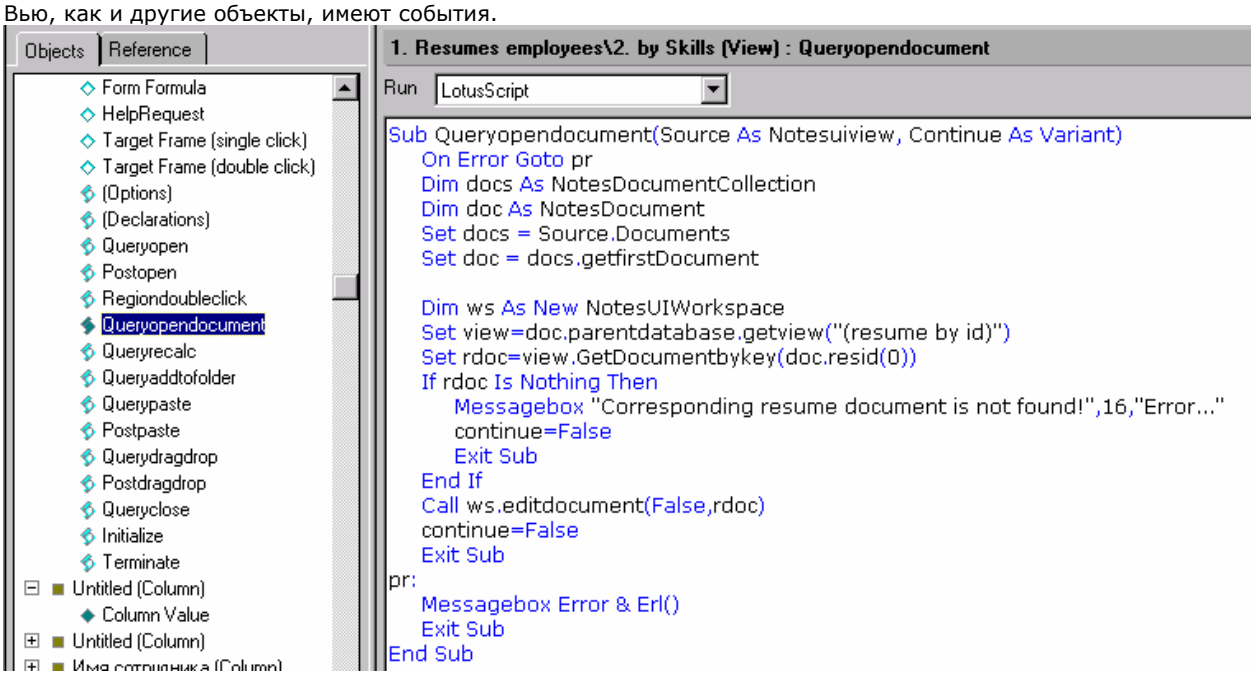
А теперь мы строим обычную иерархическую вью, но в формуле селекции мы делаем подмену поля $REF, на то, в котором хранятся наши новые иерархические связи, например:
DEFAULT $REF:=HeadID;SELECT (form*="department":"cposition") & breaked!="1"

6.3.3. Отображение произвольных картинок во вью

В свойствах колонки (первой закладке) уже обсуждалось использование стандартных иконок. Их, кажется, что-то около 175 штук (их таблица есть в Designer Help), но все они уже порядком надоели. Вы можете нарисовать свои иконки, положить их в Image resources и в формуле колонки использовать название этого ресурса в виде строки текста.

6.3.4. *События View*

Вью, как и другие объекты, имеют события.



1. QueryOpen – событие, которое происходит прямо перед открытием вью
2. PostOpen – после открытия вью на рабочем пространстве
3. RegionDoubleClick – событие календарной вью, которое происходит сразу после двойного щелчка мышью на элемент календаря
4. QueryOpenDocument – событие происходит прямо перед открытием документа из представления – на этом событии в картинке происходит выявление открываемого документа, у которого в этой базе нет формы, нахождение его родителя и открытие этого родителя, а событие открытия документа без формы возвращает continue=False, поэтому исходный открываемый документ не открывается.
5. QueryRecalc – событие, которое происходит при рефреше вью
6. QueryAddToFolder – происходит прямо перед операцией добавления документа в папку
7. QueryPaste – происходит прямо перед копированием документа во вью – здесь можно делать какие-то проверки и запрещать вставку документов
8. PostPaste – на этом событии можно поймать вставленный документ и что-то с ним сделать – ну например, удалить, или поставить ему флаг. Что он является копией
9. QueryDragDrop – прямо перед выполнением действия Drag & Drop – не знаю как это по-русски сказать
10. PostDragDrop после выполнения того же самого
11. QueryClose – прямо перед закрытием вью

6.3.5. *Form Formula*

Одно из свойств вью, наряду с методами это Form Formula. Есть вы указали здесь имя формы, то все документы в этой вью будут открываться и создаваться по этой форме, даже если в поле FORM самого документа стоит другое значение. Это не распространяется на документы, в которых форма хранится в самом документе (они были созданы по форме с флагом Store in document). То есть Form formula перекрывает форму в поле form.

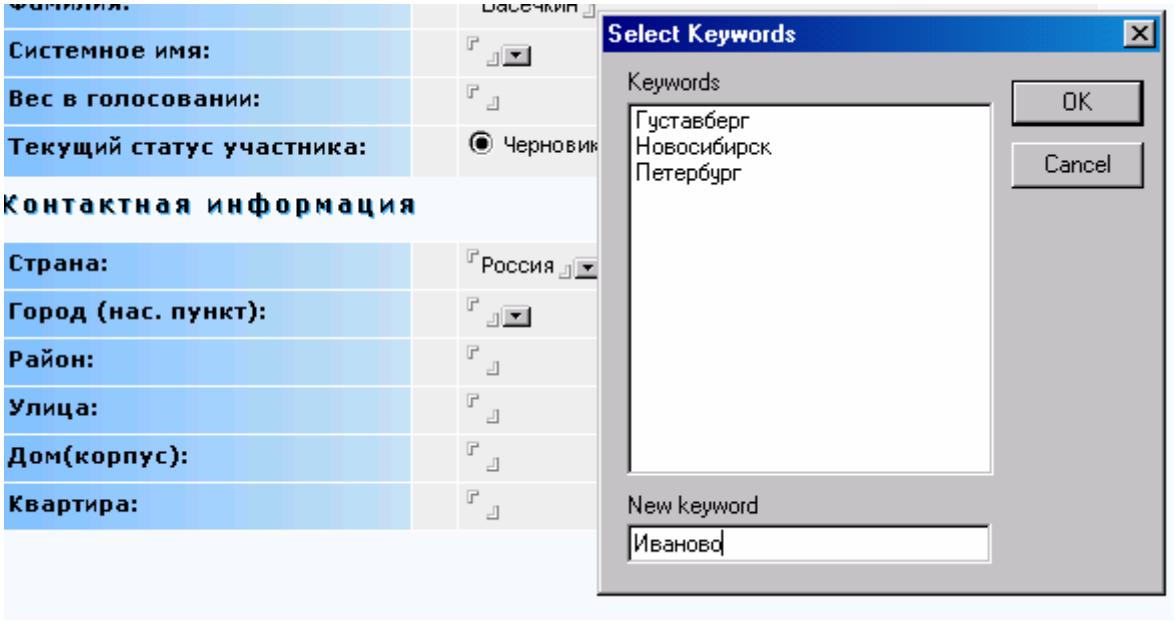
6.3.6. *Продолжение дизайна полей базы, которые используют вьюхи*

А теперь вернемся к дизайну нашей формы и подформы с контактной информацией. Мы хотели, чтобы поля со страной и городом позволяли выбор из уже введенных значений и добавление нового.

Поле, которое позволяет выбор это …. Какой это тип поля?
Посмотрите, какие типы нам подойдут. Я возьму поле типа dialog List, вы можете взять другой тип, который вам более нравится.

В этом поле я включу опцию на 2 закладке – Allow values not in list – она нам позволит вводить новые значения.

Опции списка на 2 закладке я выберу Use formula for choices и напишу в окно формулу, которая составит список изо всех, существующих значений этого поля в документах. Как такую формулу написать?
Я могу построить вью, в которой в первой колонке будет интересующее меня поле. Потом использовать формулу, которая получит список значений в этой колонке @dbcolumn("":"NoCache";@dbname;"имя вью";номер колонки), потом к результату я применю формулу, которая уникализует список – для удаления дубликатов @Unique(@dbcolumn(…))
Давайте назовем вьюху (by Countries) – для поля со страной и (by Cities) для поля с городами, город (страна) у нас будут в первой колонке, поэтому формула для поля будет выглядеть соответственно @unique(@dbcolumn("":"NoCache";@dbname;"(by Cities)";1)) и @unique(@dbcolumn("":"NoCache";@dbname;"(by Countries)";1)).
И создайте вью. Эти вью с именами в круглых скобках – скрыты от пользователя при просмотре вью из меню View->goto.
При создании вью, вам необходимо сделать обычную shared view, с формулой селекции select form="member" и одной единственной колонкой, содержащей соответствующее поле документа. Есть одна тонкость – ее надо запомнить:
Чтобы по вью можно было осуществлять поиск и брать из нее значения она должна иметь одну сортированную колонку – то есть колонка вью на 2 закладке должна иметь включенную опцию сортировки.
В наших вью колонка должна быть отсортирована.
Я сама создала в базе 2 вью и прописала в поля со страной и городом свойства и формулы селекции, вот что получилось:



6.4. Агенты

Как уже говорилось, агенты это логически выделенные куски кода, которые могут запускаться при нажатии кнопок, из всех событий, на которых используется Lotus script или формулы.

Агенты могут выполняться на сервере и на клиенте.
При выполнении на сервере код выполняется от имени пользователя, которым подписан агент.
При выполнении на клиенте – от имени текущего пользователя.

При создании агента (закладка Agents->New agent) указывается:

1. Имя | Алиас
2. Общедоступность или персональность агента (эта опция **не может** быть изменена в будущем при редактировании агента)
3. Каким образом запускается агент
4. На каком множестве документов он запускается
5. А также код агента
6. Есть ряд опций, которые отвечают за периодичность запуска, на каком сервере агент пускается, доступен ли он для публичных пользователей, сужает множество документов, на которых пускается агент и т.д. – они регулируются кнопками Options, Schedule, Add Search

Теперь разберем пункты с вариантами выбор детально:

6.4.1. Когда агент может быть запущен

1. Manually from action menu – это агенты, которые запускаются из меню Действия(Action) на указанном множестве документов. Чтобы в таком агенте это множество получить обычно используется код вида:

   Dim session as new NotesSession – рождение новой сессии, из которой в дальнейшем можно получить доступ к текущим объектам
   Set coll=session.currentDatabase.UnprocessedDocuments – получение текущей базы и коллекции необработанных документов

   Надо сказать, что множество документов, которые будут для агента рассматриваться, как необработанные, задается второй опцией – Which documents should it act on
2. Manually from agent list – такие агенты вызываются из кода кнопок, событий и пр. Они недоступны обычному пользователю из меню. Их можно получить по имени из объекта текущей базы, например
   Dim session as new NotesSession
   Set agent=session.currentDatabase.GetAgent("Ulialia")
   Call agent.run или call agent.runOnServer
   При запуске агенту можно передать документ, на котором он должен отработать, и делается это следующим образом:
   Передается в качестве строки NoteID. Этого документа в базе, то есть call agent.runOnServer(doc.NoteID)
   А в самом агенте переданный параметр и документ по параметру берутся так:
   Dim session as new NotesSessiom
   param=session.CurrentAgent.parameterDocID
   set paramDoc=session.CurrentDatabase.GetDocumentbyID(param)
3. Before(After) new mail arrives – прямо перед и сразу после прихода почты в базу
4. При создании нового или модификации существующего документа в базе
5. При вставке документа в базу
6. По расписанию – чаще раза в день, раз в день, раз в неделю, раз в месяц
7. По расписанию – никогда – это агенты, которые вызываются из других агентов

Вообще все это разграничение весьма условно. Вы всегда можете вызвать правильно написанный агент методом Run или RunOnServer. Чтобы не запутаться во всем множестве видов запуска для себя стоит понять и запомнить что:

1. Все агенты можно запустить из скрипта и формул на клиенте или сервере, но надо всегда понимать текущий контекст запуска и как они в этом контексте будут работать
2. Есть агенты по расписанию – их надо использовать для выполнения периодических задач
3. Есть агенты для работы с почтой
4. Есть агенты, которые запускаются на Веб.
5. Есть агенты, запускающиеся по событиям в базе

Агенты из 3 и 4 пункта получают документ, с которым они работают через set doc=session.DocumentContext

6.4.2. Множество документов, на которых запускается агент

Надо понимать, что можно это множество задать опциями в агенте – но это весьма и весьма негибко, а можно определить это множество в самом агенте – то есть в коде агента взять нужные документы, например, из

специально для этого множества заготовленной вью или, воспользовавшись поиском, который по сути создаст ту же вьюху только динамически, поэтому будет сильно тормозить. Можно в самом агенте взять документ, например, как переданный параметр.

ТО есть – есть множество документов, которые будут передаваться агенту как unprocessed либо через session.DocumentContext и которые определяются опцией агента «На каких документах его пускать». Далее агент может работать с этим множеством, а может сам обратиться, например, к вьюхе и пройтись по ней, наплевав на то множество, которое ему передано на обработку опцией.

Пример:

Я создаю агента, который запускается раз в день и работает на All documents in database. При этом в самом агенте в его коде, вместо того чтобы брать документы и что-то с ними делать из session.currentDatabase.UnprocessedDocuments я делаю так:
Set view = session.CurrentDatabase.GetView("MyView")
Set doc=view.GetFirstDocument
While not doc is nothing
……
Set doc=view.GetNextDocument(doc)
Wend
Тут правда надо быть аккуратным – по стандартной логике такой агент берет документ из вью, что-то с ним делает, выставляет флаг, что он этот документ уже обработал и переходит к следующему. Если во вью стоит в формуле выбор, что документы с флагом – не попадают во вью, то как только документу поставят флаг - он уйдет из вью и поэтому метод взятия следующего Set doc=view.GetNextDocument(doc) – выдаст ошибку. Чтобы этого не было – следующий берется ДО того как обрабатывается текущий и перед переходом на следующий виток цикла текущему присваивается следующий:
While not doc is nothing
Set doc1=view.GetNextDocument(doc)
……..
Set doc=doc1
Wend

Общую идеологию я рассказала, теперь более подробно о значениях опции (иногда не стоит изобретать велосипед и стоит ими воспользоваться). Для разных триггеров запуска список опций множества документов отличается.

1. Все документы в базе
2. Все новые и модифицированные с момента последнего запуска
3. Все непрочитанные в текущем вью
4. Все документы в текущем вью
5. Выбранные пользователем документы
6. Run Once – Это когда множество документов определяется самим агентом или для получения доступа к текущему документу (при запуске агента на каком-то открытом документе) – в UnprocessedDocuments будет только текущий
7. Есть ряд предопределенных множеств – Pasted document, Newly created и т.д., которые проставляются принудительно для агентов запускающихся по событиям базы или прихода почты.

Для агентов, запускающихся с Веб следующие опции недоступны:

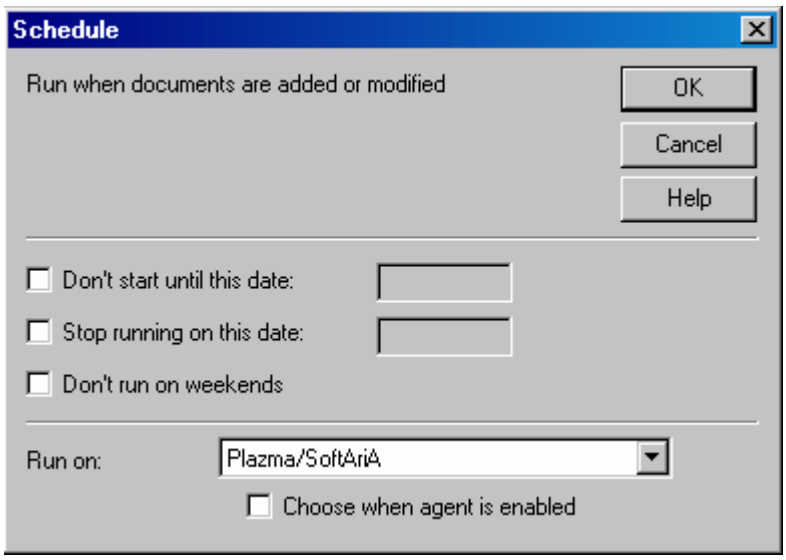
- All unread documents in view
- All documents in view
- Selected documents
- Pasted documents

То есть вы, конечно, можете их использовать в агенте и вызвать его на Веб, но множество будет пустым.

Есть еще такая опция для указания более точно множества документов, подаваемых на вход, как Add search. Она позволяет добавить условия на те документы, которые задаются в обычной опции. Например задать какие-то условия на поля и пр. Просто посмотрите сами виды условий в этой опции.

6.4.3. Подписание агента на сервер

А теперь вспомним параграф, посвященный правам доступа. Там говорилось, что чтобы агент пускался на сервере, у дизайнера, его подписавшего, должны быть права запуска агентов на сервере. А сервер указывается в опции Schedule:



Чтобы эта опция была хорошо понятна, проведем небольшой экскурс в архитектуру распределенных систем. Представим себе сложную систему, которая стоит в 15 филиалах и одном главном офисе, а также на ноутбуках у кучи, находящихся в разъездах сотрудников. И все это должно работать.

В таких больших системах время от времени возникают конфликты репликации – то есть конкурентное редактирование одного и того же документа в разных филиалах. Необходимо настроить системные активности и потоки данных таким образом, чтобы обеспечить всем доступ к необходимой им информации, не гонять лишнюю информацию (и не создавать лишний трафик), минимизировать количество конфликтов.

Как правило, необходимо нарисовать схему, входящие потоки данных, перемещение потоков и какие действия требуются от какого филиала и на каком множестве документов они осуществляются.

Необходимо достичь того, чтобы один и тот же агент не запускался на одном и том же документе (реплике документа) на разных серверах одновременно. То есть, мы можем либо ограничить агент одним сервером, на котором он пускается, а изменения внесенные агентом потом реплицировать на сервера, либо сделать так, чтобы агент пускался на всех серверах, но множество документов, поданных ему на вход – на всех серверах было разное.

Примеры:

Пример 1
Есть агент, бегает по резюме раз в день и на основе даты рождения пересчитывает поле с возрастом сотрудника.
Есть 2 сервера, множество резюме на обоих серверах идентично. В этой ситуации необходимо, чтобы агент запускался только на 1 из серверов и желательно ночью или в иное время минимальной активности пользователей. Тогда на сервере А посчитается возраст и эта информация уйдет на сервер Б.

В противном случае – подписания агента на оба сервера – информация будет посчитана и изменена одним и тем же образом, но одновременно на разных серверах, что в результате породит конфликт.

Пример 2
Есть база резюме сотрудников холдинга, и приказов по всему холдингу. В каждом филиале есть только множество документов сотрудников этого филиала, а приказы во всех филиалах все.
Есть агент, бегает на документах сотрудников и делает то же, что и в примере 1 – считает возраст. В данном случае множество документов, поданных на вход агенту – специфично для каждого филиала, поэтому мы можем включить агент на ВСЕ сервера. И на каждом сервере он обработает свое множество документов. Более того, если мы ограничим агент одним серверов, то часть документов будет вообще не обработана.

Часть агентов запускается локально и рассчитаны на работу удаленных пользователей, работающих на ноутбуках.

В опции schedule агента указывается – на каком сервере этот агент будет пускаться.

Писать агенты можно на Lotus script, Formula Language, Java или simple action – это набор простейших действий, которые можно просто выбрать из списка.

Чтобы написать агент надо владеть хотя бы одним из этих языков. Посему приступим к изучению Lotus специфичных языков – скрпта и формул.

7. Lotus Script

Я хочу сразу сказать, что здесь не будет детального и занудного описания различных конструкций, ограничений языка и прочего, что пишут в большинстве книг. Цель этой главы – дать скелет представления и базовые навыки, а почитать о методах конкретного объекта, которые решат конкретную задачу – можно и самостоятельно в официальном хэлпе, где приведена полная спецификация всех объектов, конструкций и пр.

Вообще Lotus script – это basic с набором классов для работы с объектами Lotus Notes и возможностями самому писать объектно-ориентировано.

Для отладки скрипта можете включить отладчик – в клиенте File->Tools->Debug lotus script.
Формулы, к сожалению, отладке в отладчике не подлежат.

7.1. Типы и прочие сущности

Сразу скажу – названия переменных, объектов, методов и вызовы синтаксических конструкций являются не case-sensitive - можете писать их, не зависимо от регистра. Переменные можно декларировать, присваивая им тип, а можно использовать без предварительной декларации, тогда такая переменная будет иметь тип Вариант, и содержать тот тип, который вы в нее положите. НО это дурной стиль. Первый год не рекомендуется им пользоваться, так как отсутствие декларации может доставить вам массу гемороя – вы получите кучу непонятных или скрытых ошибок в коде.

7.1.1. Простые типы

Есть различные простые типы – строка, числовые типы с различным количеством байт для хранения. Таблица простых типов выглядит так:

Тип	**Значения**	**Размер**
Integer	-32,768 до 32,767	2 bytes
Long	-2,147,483,648 до 2,147,483,647	4 bytes
Single	-3.402823E+38 до 3.402823E+38	4 bytes
Double	-1.7976931348623158+308 до 1.7976931348623158+308	8 bytes
Currency	-922,337,203,685,477.5807 до 922,337,203,685,477.5807	8 bytes
String	Ограничена доступной памятью (но помните при вставке значения в поле документа ограничения на поля текстового типа в 32K)	2 bytes/символ

Для декларации переменной какого-либо типа используется конструкция DIM имя переменной as имя типа, например Dim UserName_ as string присвоение значения делается так- UserName_=”Bebebe zuzuzu”

7.1.2. Сложные типы

Существуют более сложные конструкции:

Конструкция	**Описание**	**Размер**
Array	Массив элементов одного типа. Может быть многомерным до 8 измерений. Каждый из индексов может варьироваться от -32,768 до 32,767. Декларируется так: Dim имя переменной с массивом (нижняя граница индекса to верхняя граница индекса) as тип данных Например: Dim my_array(0 to 14) as string	Доступная память

	Кстати надо отметить, что массив может быть не только из простых типов, но из сложных, например массив объектов. Массив может быть динамическим, тогда он определяется изначально имеющим некоторое кол-во элементов и потом может быть переопределен, например: Redim brdr (0 to 0) as string – это я изначально задаю динамический массив, потом я его переопределяю: Redim brdr (0 to 8) as string – при переопределении сохраняется тип. Если я в массив перед его переопределением уже вбила значения элементов, например: Brdr(0)=”biaka” Brdr(1)=”kaka” И, переопределяя размерность, я хочу, чтобы заполненные значения у меня не потерлись, я пишу такое слово redim preserve brdr(0 to 10) as string	

List	Одномерный массив элементов одного типа, которые вызываются не по индексу, а по уникальным именам.
	Декларация выглядит так:
	Dim aaa List As String
	
	Создание нового элемента или присвоение значения существующему:
	aaa("jan")="Январь"
	
	Чтобы обратиться к элементу и получить его значение:
	aaa("jan")
	
	Чтобы проверить наличие элемента в списке можно пользоваться функцией IsElement,
	If IsElement(aaa(“jan”)) then
	…
	End if
	
	Чтобы по элементу получить его название в списке пользуйтесь функцией listTag, она правда работает только в цикле forall, который позволяет пробегать по всем элементам списка или массива:
	Forall a in aaa
	LastName=ListTag(a)
	End forall
	
	Для очищения списка или удаления конкретного элемента используется функция Erase aaa(), Erase aaa (“jan”)
		Вся доступная память

Variant	Это супер тип – мне он очень нравится. Но при освоении им лучше пользоваться по минимуму. В него можно запихать все, что угодно. Кроме того, если вы начинаете использовать переменную, НЕ ДЕКЛАРИРУЯ ее перед этим, то она по умолчанию имеет именно этот тип. То есть я могу сделать	16 bytes

	так: Dim aaa (0 to 10) as string Aaa(0)=”Kaka” Aaa(1)=”Biaka” Tmp=aaa После чего у меня появится переменная tmp типа вариант, которая будет содержать массив	

User-defined data type	Определяемый пользователем тип – может содержать набор элементов разных системных типов. Это аналог record в Pascal или struct в C.
	
	Декларация выглядит так:
	Type phoneRec
	  name As String
	  areaCode As Integer
	  phone As String * 8
	End Type
	
	Перед типом можно указать его вид
	[Public | Private]
	Вообще, эти конструкции применимы и для деклараций других типов при объектном стиле написания кода.
	Смысл их аналогичен смыслу в других языках – приватные переменные видны только в классе, публичные доступны отовсюду.
		64K bytes

User-defined class	Класс, написанный пользователем – смотрите дизайнер хэлп для описания всех конструкций.	
Object reference	Ссылка на объект, который может быть, например, OLE Объектом или объектом Lotus или объектом, написанным пользователем. Такие ссылки декларируются как и другие типы, просто вместо имени типа ставится имя объекта – например: Dim doc as NotesDocument А при установке значения ВСЕГДА используется ключевое слово SET: Set doc=new NotesDocument(db)	4 bytes

7.2. Классы lotus

Теперь займемся освоением классов Lotus на конкретных примерах, чтобы хорошо запомнилось.

Для того чтобы родить объект lotus как правило используется либо New либо функции получения этого объекта из существующего.

Давайте для примера напишем кнопку, которая на текущем открытом документе осуществит выбор каких-то данных из списка и заполнение соответствующих полей текущего документа.

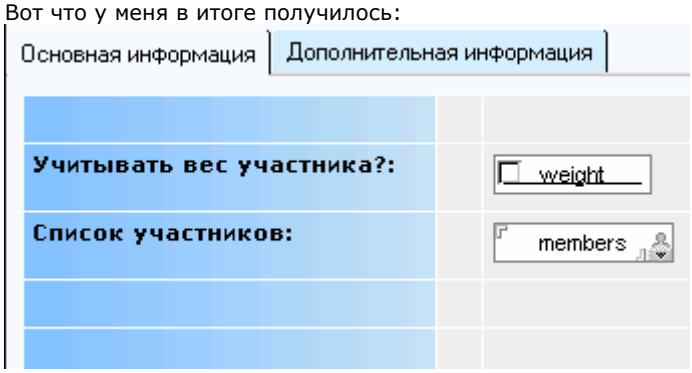
Чтобы пример был актуален – давайте откроем в нашем приложении форму голосования и начнем делать в ней кнопку, которая будет выбирать список голосующих.

Итак, открываем форму Голосование (мы создавали ее как копию формы участника), прописываем для этой формы роли в закладке безопасности свойств формы. Прописываем Window title, роли в поле editors и текст в заголовке формы. Удаляем на 1 закладке таблицы все поля – чтобы они нам не мешали и все подписи, а таблицу, вложенную в таблицу с закладками – оставляем. В общем, удаляем все лишнее и оставляем бланк дизайна, в который мы будем вносить нашу логику.

Теперь в строку таблицы вносим подпись «Список голосующих» и поле, в котором этот список будет отображаться, но не редактироваться (то есть поле вычисляемое само от себя), флаги многозначности и тип этого поля выберите сами – просто подумайте что будет в нем храниться.

А также вносим в строку таблицы «Учитывать вес» и поле в котором будет стоять пометка – учитывать вес или нет. Так как поле – по сути просто пометка, то наверное оптимально использовать редактируемый чекбокс с единственным значением.

Вот что у меня в итоге получилось:



Теперь, в колонке между списком участников и полем, я вложу имидж и натяну на него хотспот, который и запрограммирую, как мне хочется – на выбор участников.

Итак, пишу код:

```
Sub Click(Source As Button)
        Dim ws As New NotesUIWorkspace

        Dim uidoc As NotesUIDocument
        Set uidoc=ws.currentDocument

        Dim doc As NotesDocument
        Set doc= uidoc.document
End Sub
```

Для начала я рождаю объект, который представляет мою рабочую область и даст мне потом доступ к тому, что лежит в этой области – это notesUIWorkspace – я его порождаю стандартным способом.

У этого объекта кучка свойств и методов, которые возвращают текущие объекты в рабочей области – документ, вью, базу данных и пр, а также рождают различные окошки.

Я использую свойство CurrentDocument, которое возвращает документ, открытый в рабочей области и возвращает его именно как открытый на пространстве. Возвращенный объект является экземпляром класса NotesUIDocument. Вообще все классы с префиксом NotesUI – это классы, который предоставляют доступ к объекту в рабочей области, а не бакэнде.

Теперь, чтобы пользоваться информацией, указанной в документе и вносить в него информацию, я получу доступ к документу в бакэнде. Я декларирую переменную класса NotesDocument и беру документ из бакэнда текущего документа Set doc=uidoc.document

Теперь я могу черпать данные из документа просто обращаясь к имени поля, например doc.weight или doc.members

Теперь обратите внимание, что в поле может храниться как одно значение, так и массив. Поэтому **значение поля это** фактически всегда **МАССИВ.** Просто в нем может быть один элемент или более. И если вы положите значение поля в какую-либо переменную, то это должна быть переменная типа вариант, которая будет массивом той длины, что имеет поле, и к реальным значениям поля вы будете обращаться потом по индексу. Самый первый индекс по умолчанию 0.
Например,

```
mems=doc.members
dim FirstMember as string
Firstmember = Mem(0)
```

Можно сразу к полю обратиться doc.members(0), но при таком обращении вы должны быть уверены на 100%, что такое поле на документе точно есть, потому что иначе результат вызова doc.members даст просто пустой объект и обращение к элементу массива, которым этот объект не является – даст ошибку.

Для избежания этой ситуации вы можете просто проверять документ на наличие такого поля:

```
If doc.HasItem("members") then
        firstMember=doc.members(0)
End if
```

Итак, продолжим написание нашей кнопки:

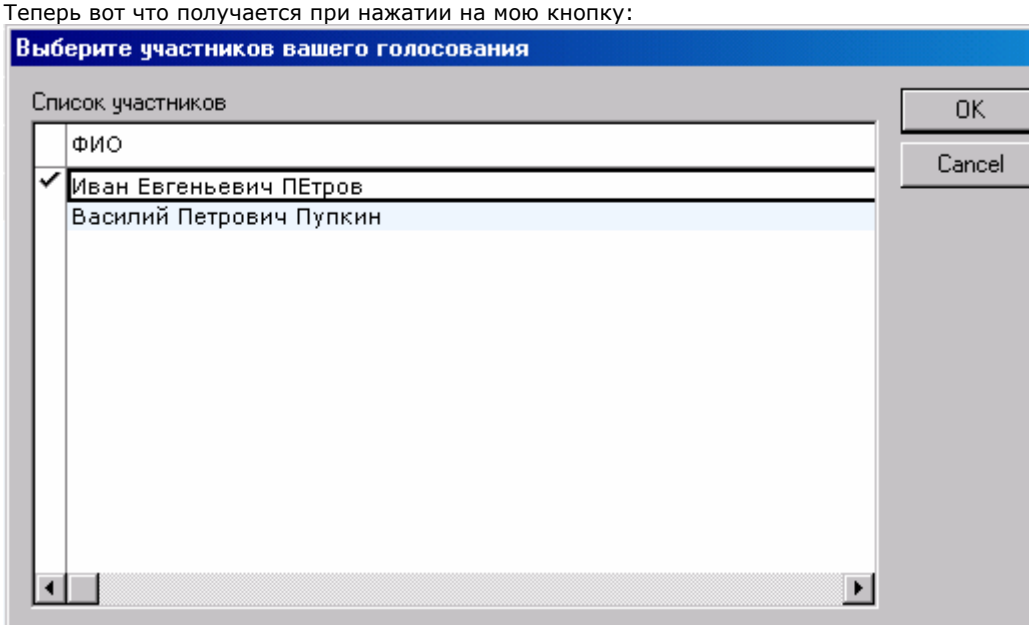
```
Sub Click(Source As Button)
        Dim ws As New NotesUIWorkspace

        Dim uidoc As NotesUIDocument
        Set uidoc=ws.currentDocument

        Dim doc As NotesDocument
        Set doc= uidoc.document

        Dim db As NotesDatabase
        Set db=doc.parentDatabase

        Set coll=ws.PickListCollection(PICKLIST_CUSTOM, True,db.server, db.filePath,"v11","Выберите участников вашего голосования", "Список участников","Активный")
End Sub
```

Мы породили объект – текущая база данных – взяв ее как свойство объекта NotesDocument.ParentDatabase – которое возвращает базу, в которой находится документ.

А далее мы вызываем на notesUIWorkspace метод pickListCollection, который откроет нам в диалоговом окне вью и позволит выбрать из нее один или несколько документов, в зависимости от флага, переданного вторым параметром.
2,3 и 4 параметры – это сервер, путь к базе и имя вью в этой базе, которые я открываю в окне. Далее 5 и 6 параметры – подписи к диалоговому окну. 7 –ой параметр – опционален – это имя категории. Я могу не все документы вью показать в окне, а только документы, попадающие в определенную категорию. В данном случае я создала вью участников голосования с алиасом «v11», категоризовав по статусам, и выбираю только активных участников.

В этой вью в свойствах обязательно надо снять на 2 закладке флаг – Collapse when…, иначе при первом открытии категории будут свернуты и в диалоговом окне будет пусто.

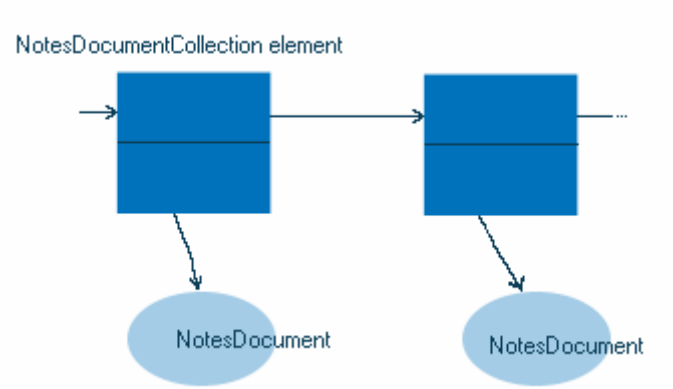
Теперь вот что получается при нажатии на мою кнопку:



Всего участников у меня штук 7, но в окне я вижу только тех, которые имеют активный статус.

Однако моя кнопка только вызывает окно и ничего пока не делает с тем, что это окно возвращает.
А возвращает это окно объект, который называется notesDocumentCollection – это коллекция документов. Она может быть пустой, если нажать на Cancel, или не выбрать ничего, а может содержать сколько-то документов.

Коллекция документов устроена следующим образом:



То есть, каждый элемент коллекции содержит ссылку на следующий (и предыдущий), а также собственно на документ, который входит в оклекцию и представлен этим элементом.

Из этой структуры сразу следуют некоторые выводы насчет оптимальности алгоритмов обхода коллекции: Есть 2 способа и в хэлпе в примерах приведен следующий:

```
For i=1 to coll.count
Set doc=coll.GetNthDocument(i)
End for
```

При использовании такого способа для подсчета каждого элемента, взятого по индексу, происходит обход коллекции от 1-го элемента до i-того. А это очень не оптимально, так как для коллекции из n элементов количество обходов будет (1 + 2 + 3 +… + n-1).

Есть человеческий способ обхода коллекции – когда используются ссылки между элементами друг на друга и коллекция обходится методом set doc=coll.GetNextDocument(doc) – то есть мы берем следующий документ от текущего. Работает такой метод намного быстрее.

Вот сейчас мы в нашей кнопке проверим возвращенную коллекцию на наличие в ней документов и пройдемся по ней, чтобы вытащить из документов участников голосования, интересующую нас информацию.

```
Dim ws As New NotesUIWorkspace

Dim uidoc As NotesUIDocument
Set uidoc=ws.currentDocument

Dim doc As NotesDocument
Set doc= uidoc.document

Dim db As NotesDatabase
Set db=doc.parentDatabase

Dim coll As NotesDocumentCollection
Set coll=ws.PickListCollection(PICKLIST_CUSTOM, True,db.server, db.filePath,"v11","Выберите участников вашего голосования", "Список участников","Активный")
If coll.count=0 Then Exit Sub ' мы проверяем, что в коллекции не 0 число документов и выходим из кода, если пользователь не выбрал ни одного документа.

Dim memberdoc As NotesDocument
Set memberdoc=coll.GetFirstDocument

While Not memberdoc Is Nothing ` мы используем цикл, в котором перебираем элементы коллекции пока не дойдем до конца, в конце коллекции, когда на последнем элементе будет вызван метод GetNextDocument – следующие элемент будет пуст и поэтому объект NotesDocument, на который смотрит несуществующий элемент коллекции – будет nothing. Nothing это когда ссылка на объект смотрит в никуда.

        Set memberdoc=coll.GetNextDocument(memberdoc)
Wend
```

Теперь нам надо из всех документов по которым мы прошли вытащить информацию о имени участника и о весе участника в голосовании.

У нас будет 2 соответствующих друг другу массива Names_ и Weight_

В момент декларации массивов мы уже знаем, сколько документов в коллекции, но сама коллекция считается динамически, и мы не знаем априори – сколько в ней будет элементов, поэтому мы объявим эти массивы динамическими:

```
Dim memberdoc As NotesDocument
Set memberdoc=coll.GetFirstDocument

Dim n As Integer
n=coll.count

Redim weights(1 To n) As Double
Redim Names_(1 To n) As String

Dim i As Integer
i=1
While Not memberdoc Is Nothing
        weights(i)=memberDoc.weight(0)
        Names_(i)=memberDoc.NotesName(0)
        Set memberdoc=coll.GetNextDocument(memberdoc)
        i=i+1
Wend
```

Теперь нам надо внести посчитанные массивы в соответствующие поля нашего документа открытого в рабочей области.

Поле документа по сути является массивом, причем динамическим, в него можно пихать массивы произвольной длинны. В данном случае оно ведет себя как переменная типа вариант. Единственное – поле не допускает присвоения массива состоящего из разнотипных элементов.

Вы можете создать поле на документе, не зависимо от того – есть ли оно на форме. Как уже говорилось в самом начале этого документа – реальный документ лежит в бакэнде, в форма – трафарет, через который мы этот документ смотрим. При этом с документом в бакэнде мы можем делать много разных вещей, а с документом, открытым в рабочей области – через форму, мы кое-что делать можем, обращаясь к полям, но наши действия ограничены логикой формы.

Я это пишу для тех, кто по ходу чтения этого документа смотрит спецификацию классов, упомянутых здесь, и увидел такие методы NotesUIDocuemnt-a, как FieldSetText.
Мы не сможем создать новое поле на NotesUIDocument этим методом.

Посему присвоение я делаю так:

```
Doc.members=names_
Doc.weights=weights
```

Мы заполнили поле, которое есть на форме и создали новое для весов. Этого поля на форме нет, а на документе оно будет.

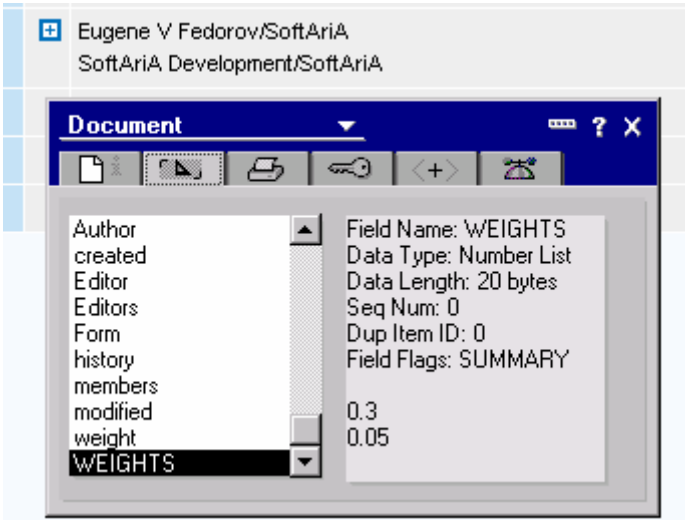
Чтобы открытый в рабочей области документ отобразил изменения, происшедшие в бакэнде его необходимо отрефрешить. Рефреш доступен только в режиме редактирования. Поэтому и кнопку нашу в режиме чтения мы просто скроем.

```
Call uidoc.Refresh
```

Вот конец нашего кода:

```
Dim i As Integer
i=1
While Not memberdoc Is Nothing
        weights(i)=memberDoc.weight(0)
        Names_(i)=memberDoc.NotesName(0)
        Set memberdoc=coll.GetNextDocument(memberdoc)
        i=i+1
Wend

doc.members=names_
doc.weights=weights

Call uidoc.refresh
```

Я запустила код и вот что у меня получилось в результате после сохранения документа:

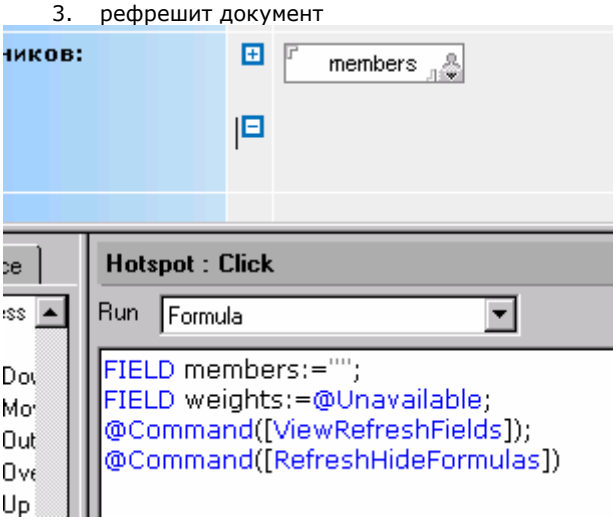


Теперь надо сделать кнопку, которая почистит эти поля:
Я делаю еще один хотспот, который скрыт в режиме чтения и если в поле members="", его я сделала просто на формулах, так как это проще:

```
field members:="";
field weights:=@unavailable;
@command([ViewRefreshFields]);
@command([RefreshHideFormulas])
```

Мы уже рассматривали такие кнопки – я повторюсь с описанием, что она делает:

1. делает пустым поле members
2. удаляет поле weights
3. рефрешит документ



Весьма вероятно, что попробовав самостоятельно сделать такие же хотспоты вы столкнетесь с одной проблемой:
Когда вы будете писать формулу скрытия для кнопки МИНУС, если вы потом перейдете на кнопку ПЛЮС, то увидите, что та же формула появилась на кнопке ПЛЮС, хотя ее-то нам как раз скрывать не надо, если у нас не выбраны участники. Дело в том, что формула скрытия распространяется на все элементы, стоящие в одной строке. Поэтому чтобы избежать распространения формулы – просто нужно отделить один хотспот от другого переводом строки. Это касается и полей и прочих элементов. **Формулы скрытия это атрибут строки, в которой расположен объект.**

Мы реализовали логику выбора и удаления участников голосования.

Это просто пример, для того чтобы «пощупать» классы Lotus и их использование.
Теперь пройдемся по часто используемым классам более подробно и опишем задачи, в решении которых они используются.

7.2.1. Классы родители всего ☺

Как уже рассматривалось в примере – доступ к рабочей области осуществляется через класс NotesUIWorkspace, есть ряд UI-классов, которые представляют объекты в рабочей области. Как правило, из

этих UI-классов, представляющих объект, открытый в рабочей области, можно получить доступ к этому же объекту в бакэнде. Например, NotesUIDocument.Document или notesUIView.View.

Для того чтобы получить доступ к окружению, например в агенте, который запускается на сервере и не имеет доступа к рабочей области, используется класс NotesSession.

ВАЖНО: В агентах, запускаемых на сервере нельзя использовать UI-классы, а также подключать библиотеки, использующие эти классы.

Из этого класса получаем доступ к текущей базе данных – объекту, представляющему ее в бакэнде, текущему агенту, пользователю и прочим объектам и переменным окружения.

Сейчас мы рассмотрим таблицу частоиспользуемых свойств и методов этих 2 классов:

NotesUIWorkspace	
Свойства	**Описание**
CurrentCalendarDateTime	Возвращает значение типа Date. Когда вы во вью календарного типа стоите в одной из ячеек, представляющей определенный день, и вызовете этот метод, то вы дату этого дня и получите.
CurrentDatabase	Текущая база данных, открытая в рабочей области. Возвращает объект типа notesUIDatabase
CurrentDocument	Открытый в рабочей области документ. Возвращает объект типа NotesUIDocument. Надо понимать, что если не открыт никакой документ, то вернет Nothing
CurrentView	Открытая в рабочей области вью. Тип объекта notesUIView. Если не открыто никакой вью, вернет nothing
Методы	**Описание**
AddDatabase(server$, filename$)	Открывает в рабочей области базу на указанном сервере и с указанным путем к базе. В рабочей области подсвечивается иконка базы.
CheckAlarms()	В почтовом файле пользователя находится информация о различных события – встречах, задачах и пр. Клиент запускает периодически задачу, которая называется alarm-daemon, которая выдает различные предупреждения о наступлении события, согласно настройкам. Этот метод запускает этот alarm-daemon, который проверяет почтовый ящик текущего пользователя на предмет необходимости выдачи окошка с предупреждениями.
ComposeDocument	Set notesUIDocument = notesUIWorkspace.ComposeDocument([server$ [, file$ [, form$ [, windowWidth# [, windowHeight#]]]]]) Этот метод позволяет программно открыть в рабочей области новый документ по форме, которая указывается в качестве параметра, и находится в соответствующей базе, на соответствующем сервере. Можно задать также размеры окна. Если база лежит локально, то всегда во всех методах, которые используют сервер, для обозначения локальности базы используется пустая строка “”. При создании документа этим методом выполняются все события формы – QueryOpen, PostOpen. Пример: set uidoc = ws.composeDocument(“”,”sf.nsf”,”SF”,640,480) Если вы используете метод в кнопке, которая лежит на вью, в которой в Form formula написана другая форма, то документ создастся по форме указанной в form formula, чтобы этого избежать можно модифицировать саму form formula-y: @if(@isnewdoc;return(form);”ViewPrefferedForm”)
DialogBox	flag = notesUIWorkspace.DialogBox(form$ [, autoHorzFit [, autoVertFit [, noCancel [, noNewFields [, noFieldUpdate [, readOnly [, title$ [, notesDocument [, sizeToTable [, noOkCancel]]]]]]]]]) Этот метод открывает переданный ему notesDocument по указанной форме. При этом задается масса параметров отображения этого окна. Если форма имеет поля ввода, и пользователь что-то вносит в этот документ, то вся внесенная информация доступна в notesDocument в полях, соответствующих именам полей формы. Я могу открыть существующий документ, который открыт в рабочей области, могу создать какой-то промежуточный документ, в котором сохраню результаты диалога, и потом использую их в коде, как мне требуется.
EditDocument	Set notesUIDocument = notesUIWorkspace.EditDocument([editMode [, notesDocument [, notesDocumentReadOnly [, documentanchor$]]]]) Этот метод открывает в рабочей области переданный ему notesDocument. Указывается режим – чтение или редактирование. Опциональные параметры – это запрещение пользователю потом переключить документ из чтения в редактирование. Последним параметром автор не пользовался, поэтому описать как он действует в реальности не может.

EditProfile	Set notesUIDocument = notesUIWorkspace.EditProfile(profileName$ [, userName$]) Этот метод открывает на редактирование существующий или создает новый профильный документ с этим именем или именем формы для текущего пользователя. Надо сказать, что профиль может быть один для всех пользователей.
EnableAlarms	flag = notesUIWorkspace.EnableAlarms(True/False) – включает/выключает alarm-daemon
Folder	Отображает диалог перемещения в папку текущего документа
New	Рождает новый экземпляр класса dim ws as new NotesUIWorkspace
OpenDatabase	Call notesUIWorkspace.OpenDatabase(server$, file$, view$, key$, newInstance, temp) Открывает базу данных, указанную вью, в рабочей области, может открыть в новом окне, может не добавлять иконку на рабочий стол.
OpenFileDialog	Открывает диалог выбора файла, можно выбрать несколько файлов, возвращает список имен выбранных файлов.
OpenFrameSet	Call notesUIWorkspace.OpenFrameSet(frameset$) Открывает указанный по имени фрэймсет
OpenPage	Call notesUIWorkspace.OpenPage(pagename$) Открывает указанную страницу в рабочей области
PickListCollection PickListStrings	Эти методы выдают окно диалога, для выбора значений из вью. Возможен выбор из определенной категории. Первый метод возвращает объект NotesDocumentCollection – коллекцию документов, второй – массив строк из указанной колонки вью.
Prompt	variant = notesUIWorkspace.Prompt(type%, title$, prompt$, [default] [, values]) Выдает диалоговое окно, тип которого задается первым параметром и позволяет вводить, выбирать значения из списка и прочий интерактив.
ReloadWindow	Заново загружает содержимое текущего окна
RefreshView	Обновляет текущее вью

NotesSession	
Свойство	**Описание**
AddressBooks	Все известные адресные книги
CommonUserName	Имя текущего пользователя в формате First Middle Last
CurrentAgent	Текущий агент, объект типа NotesAgent
CurrentDatabase	Текущая база данных, объект типа NotesDatabase
DocumentContext	Текущий документ, находящийся в памяти в момент старта агента, тип notesDocument
EffectiveUserName	Имя пользователя от имени которого запущен агент