



Попробуйте потрясающие инструменты Тора

Тамар Э. Гранор
Tomorrow's Solutions, LLC
Телефон: 215-635-1958
Электронная почта:
tamar@tomorrowssolutionsllc.com

Проект VFPX, Thor, включает в себя десятки инструментов для помощи в разработке. В этом уроке мы рассмотрим некоторые из того, что может предложить Thor. В уроке будет рассмотрен ряд инструментов Thor, включая Document View, Create Locals, Compare Objects и многое другое. Мы также увидим, как сделать любой инструмент Thor доступным с помощью сочетания клавиш. Мы также рассмотрим, как добавлять собственные инструменты в Thor и, если позволит время, как настраивать пользовательские настройки для инструмента.

Одна из вещей, которая делает Visual FoxPro таким замечательным инструментом для разработки программного обеспечения, – это открытая архитектура, которая упрощает создание инструментов разработчика. Редко можно найти опытного разработчика VFP, который не написал хотя бы один инструмент для автоматизации какой-либо задачи в IDE. У некоторых людей есть целый набор инструментов разработчика (и, по сути, возможность добавить панель меню – один из примеров открытой архитектуры VFP).

Сайт VFPX был создан, чтобы позволить разработчикам VFP обмениваться инструментами, и теперь на нем размещено довольно много инструментов разработчика (вместе с кучей компонентов, предназначенных для использования в приложениях VFP). Но это не лучший способ делиться небольшими инструментами. Что такое небольшой инструмент? Что-то, что занимает всего несколько строк кода,

возможно, без необходимости взаимодействия с пользователем. Что-то, для чего создание целого проекта VFPX было бы излишеством.

По мере того, как инструмент VFPX PEM Editor достигал зрелости, Джим Нельсон и Мэтт Слэй, его основные авторы и дизайнеры, обнаружили, что им нужно много маленьких инструментов, и начали добавлять их в PEM Editor. Но многие из этих маленьких инструментов не были действительно актуальны для управления свойствами, событиями и методами форм и классов. PEM Editor просто оказался удобным способом их распространения.

В конце концов, они поняли, что на самом деле нужен инструмент для управления инструментами, и так родился Thor. Thor – это инструмент, разработанный для управления инструментами разработчика; он поставляется с целым набором инструментов, но его можно расширять, чтобы вы могли добавлять свои собственные инструменты разработчика, а также те, которые вы получаете от других. Thor позволяет назначать горячие клавиши любому установленному инструменту, а также создавать пользовательские панели в меню VFP и пользовательские всплывающие меню, вызываемые с помощью горячих клавиш.

В этом сеансе мы рассмотрим ряд инструментов, которые идут с Thor, чтобы показать вам, почему вы хотите потрудиться изменить свой способ работы. Поскольку большинство инструментов работают с кодом в IDE, нам нужно будет продемонстрировать некоторые программы, формы и классы. Насколько это возможно, я буду использовать код, который идет с VFP, например классы из FFC (FoxPro Foundation Classes).

Начало работы с Тором

Чтобы использовать Thor, вам нужно всего лишь загрузить его с VFPX и установить. На сайте VFPX есть инструкции по установке Thor <http://vfpX.codeplex.com/wikipage?title=Thor%20Install&referringTitle=Thor%20Help>. После этого вы увидите две новые панели в меню VFP (см. рисунок 1) и откроется форма конфигурации Thor. Панель Thor содержит элементы для управления самим Thor. Панель Thor Tools содержит инструменты, которые поставляются с Thor; вы можете добавлять инструменты, а также изменять и удалять встроенные инструменты. Вы также можете указать, что любой инструмент должен запускаться при запуске Thor, и назначить горячую клавишу любому инструменту. Все эти действия выполняются с помощью формы конфигурации Thor, доступной из панели меню Thor.



Рисунок 1. Thor добавляет две панели в системное меню VFP.

Чтобы гарантировать, что Thor открывается каждый раз при открытии VFP, добавьте одну строку кода, показанную в листинге 1, в вашу программу запуска VFP. Число, которое вы передаете в качестве параметра, определяет частоту запуска процесса обновления Thor; например, указанное количество дней. Обновление Thor помогает вам поддерживать Thor и другие инструменты VFPX в актуальном состоянии.

Листинг 1. Эта строка кода в вашей программе запуска VFP еженедельно проверяет наличие обновлений VFPX.

```
DO D:\Fox\VFPX\Thor\Thor\RunThor.PRG WITH 7
```

После установки и настройки Thor для запуска при каждом запуске VFP вы готовы начать изучать предоставляемые им инструменты. Большая часть этого документа посвящена подмножеству этих инструментов, которые я считаю наиболее убедительными. Вы можете обнаружить, что другие инструменты вам больше всего нравятся. Я также покажу вам, как сделать любые нужные вам инструменты Thor легкодоступными.

Выделите структуру управления

Меню Code | Control Structures | Highlight Control Structure

Я много работаю с кодом других людей, то есть с кодом, изначально написанным другим разработчиком. Иногда даже приукрашивания кода недостаточно, чтобы помочь мне понять его структуру. Инструмент подсветки структуры управления Thor удобен, когда я смотрю на определенный блок кода и пытаюсь его понять. Он подсвечивает всю структуру, в которой находится курсор, будь то IF-ENDIF, FOR-ENDFOR, DO CASE, SCAN-ENDSCAN, DO WHILE, TEXT-ENDTEXT или TRY-CATCH. Если вы запустите инструмент во второй раз, он подсвечивает структуру, содержащую ту, которую вы уже подсветили. Последующие использования этого инструмента многократно повторяют это действие.

Например, на рисунке 2 показан блок кода из VFPXTAB.PRG. Курсор установлен на операторе присваивания, который находится внутри блока IF (между «ISNULL» и его открывающей скобкой). Блок IF находится внутри оператора CASE, который содержится в цикле FOR. Цикл FOR находится внутри цикла SCAN.

```
SCAN
  * Sum the relevant fields m.gtotal = .NULL.
  FOR i = 2 TO FCOUNT() - 1
    IF ISNULL(EVAL(FIELD(m.i)))
      LOOP
    ENDIF
    IF ISNULL(m.gtotal) AND !ISNULL(EVAL(FIELD(rn.i)))
      gtotal = 0
    ENDIF
    DO CASE
      CASE THIS.totaltype = COUNT_FIELDS
        * Count values
        IF THIS.shownulls
          gtotal = m.gtotal + IIF(ISNULL(EVAL(FIELD(m.i))), 0, 1)
        ELSE
          cTmpField = field(m.i)
          gtotal = m.gtotal + IIF(ISBLANK(&cTmpField), 0, 1)
        ENDIF
      OTHERWISE
        * SUM values
        gtotal = m.gtotal + EVAL(FIELD(m.i))
    ENDCASE
  ENDFOR
  IF THIS.totaltype = PERCENT_FIELDS
    gtotal = IIF(m.sumallflds=0 OR ISNULL(m.gtotal) OR m.gtotal=0, ;
```

```

0,ROUND(m.gtotal/m.sumallflds*100,THIS.nPercentFldDec))
ENDIF
REPLACE (m.totfldname) WITH m.gtotal
ENDSCAN

```

Рисунок 2. Этот блок кода, взятый из VFPXTAB.PRG, содержит IF внутри CASE, внутри цикла FOR, внутри цикла SCAN.

Рисунок 3 показывает результат использования Highlight Control Structure, а Рисунок 4 показывает код после второго применения Highlight Control Structure. Использование инструмента в третий раз выделит весь цикл FOR, а четвертое использование выделит весь SCAN. Фактически, весь этот блок кода находится внутри другого оператора IF, и пятое использование структуры управления выделением подчеркивает этот IF.

```

SCAN
* Sum the relevant fields m.gtotal = .NULL.
FOR i = 2 TO FCOUNT() - 1
  IF ISNULL(EVAL(FIELD(m.i)))
    LOOP
  ENDIF
  IF ISNULL(m.gtotal) AND !ISNULL(EVAL(FIELD(rn.i)))
    gtotal = 0
  ENDIF
  DO CASE
    CASE THIS.totaltype = COUNT_FIELDS
      * Count values
      IF THIS.shownulls
        gtotal = m.gtotal + IIF(ISNULL(EVAL(FIELD(m.i))),0,1)
      ELSE
        cTmpField = field(m.i)
        gtotal = m.gtotal + IIF(ISBLANK(&cTmpField),0,1)
      ENDIF
    OTHERWISE
      * SUM values
      gtotal = m.gtotal + EVAL(FIELD(m.i))
    ENDCASE
  ENDFOR
  IF THIS.totaltype = PERCENT_FIELDS
    gtotal = IIF(m.sumallflds=0 OR ISNULL(m.gtotal) OR m.gtotal=0, ;
      0,ROUND(m.gtotal/m.sumallflds*100,THIS.nPercentFldDec))
  ENDIF
  REPLACE (m.totfldname) WITH m.gtotal
ENDSCAN

```

Рисунок 3. Использование структуры управления подсветкой в коде на рисунке 2 выделяет только оператор IF, на котором находился курсор.

```

SCAN
* Sum the relevant fields m.gtotal = .NULL.
FOR i = 2 TO FCOUNT() - 1
  IF ISNULL(EVAL(FIELD(m.i)))
    LOOP
  ENDIF
  IF ISNULL(m.gtotal) AND !ISNULL(EVAL(FIELD(rn.i)))
    gtotal = 0
  ENDIF
  DO CASE
    CASE THIS.totaltype = COUNT_FIELDS
      * Count values
      IF THIS.shownulls

```

```

        gtotal = m.gtotal + IIF(ISNULL)(EVAL(FIELD(m.i))) ,0,1)
    ELSE
        cTmpField = field(m.i)
        gtotal = m.gtotal + IIF(ISBLANK(&cTmpField),0,1)
    ENDIF
    OTHERWISE
        * SUM values
        gtotal = m.gtotal + EVAL(FIELD(m.i))
    ENDCASE
ENDFOR
IF THIS.totaltype = PERCENT_FIELDS
    gtotal = IIF(m.sumallflds=0 OR ISNULL(m.gtotal) OR m.gtotal=0, ;
        0,ROUND(m.gtotal/m.sumallflds*100,THIS.nPercentFldDec))
ENDIF
REPLACE (m.totfldname) WITH m.gtotal
ENDSCAN

```

Рисунок 4. Второе использование структуры управления выделением выделяет весь оператор CASE.

Конечно, если вам нужно будет продираться через три слоя меню, чтобы использовать этот инструмент, он, вероятно, не покажется вам таким уж удобным. Однако одной из особенностей Thor является то, что вы можете назначить сочетание клавиш любому инструменту. Прежде чем перейти к рассмотрению других инструментов, давайте посмотрим, как это можно сделать.

Инструменты всегда под рукой

Thor предлагает несколько механизмов, упрощающих использование его инструментов. Самый простой – назначить комбинацию клавиш инструменту, чтобы вы могли использовать его без навигации по меню инструментов Thor. Для этого откройте форму конфигурации Thor, выбрав Thor | Configure в меню. Щелкните вкладку Определения инструментов (Tool Definitions), чтобы открыть страницу Определения инструментов, и перейдите в Treeview на левой панели, пока не найдете инструмент, к которому вы хотите добавить горячую клавишу. На рисунке 5 показана страница определений инструментов с выбранным инструментом Highlight Control Structure.

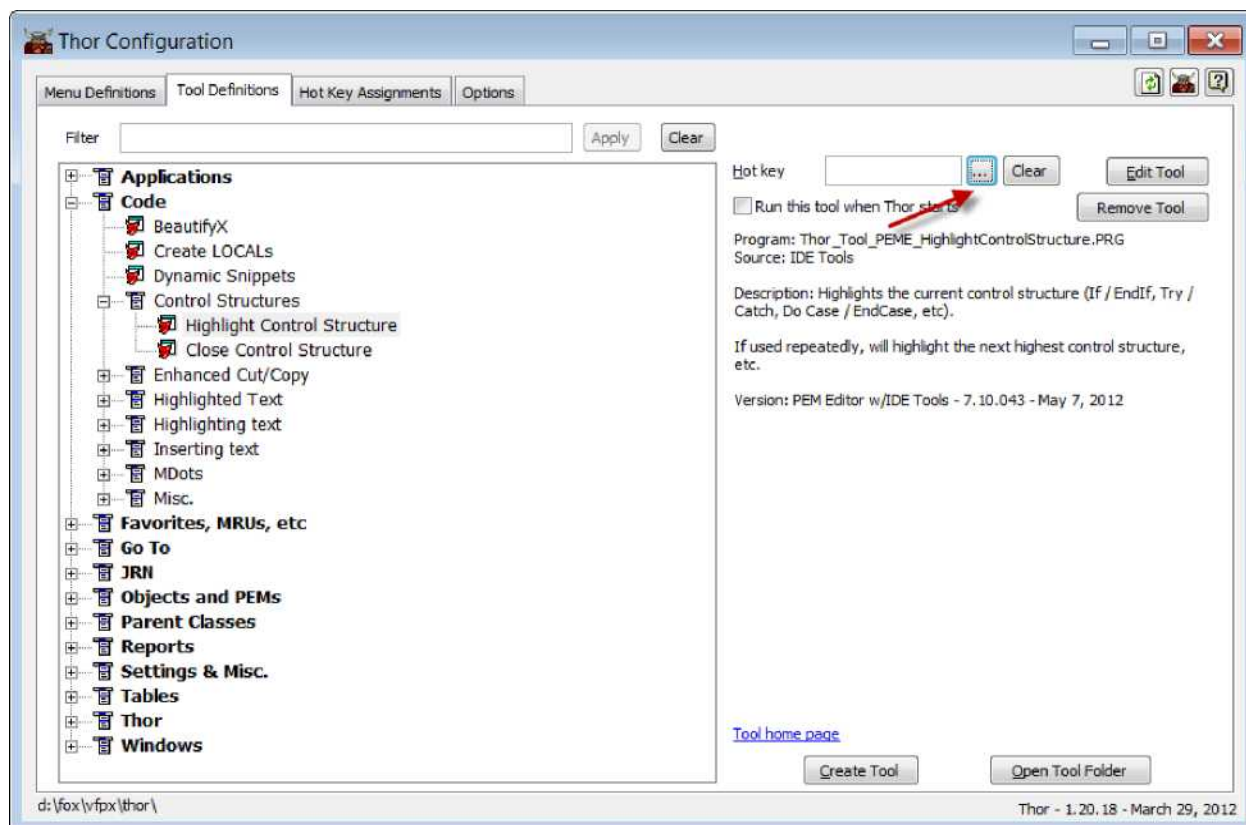


Рисунок 5. Страница «Определения инструментов» формы конфигурации Thor позволяет указать комбинацию клавиш для запуска инструмента.

Нажмите кнопку с многоточием (указано на рисунке), а затем, как указано в появившемся сообщении (рисунок 6), нажмите комбинацию клавиш, которую вы хотите использовать. После этого сообщение исчезнет, а в текстовом поле отобразится указанная горячая клавиша. На рисунке 7 вы можете видеть, что я указал Shift+Ctrl+C в качестве горячей клавиши для выделения структуры управления.



Рисунок 6. После нажатия кнопки с многоточием для горячей клавиши появляется это сообщение. Сделайте так, как там написано, чтобы указать горячую клавишу.

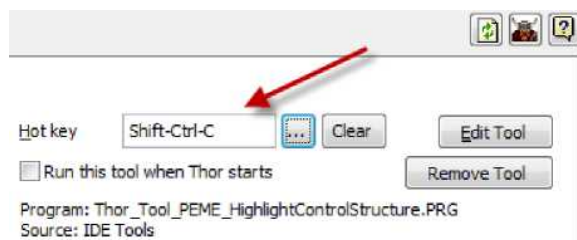


Рисунок 7. После ввода нужной комбинации клавиш она появится в текстовом поле горячих клавиш.

Указанная вами комбинация клавиш не вступит в силу, пока вы не закроете форму конфигурации Thor или не нажмете кнопку Thor в правом верхнем углу формы, чтобы обновить меню и горячие клавиши.

Форма конфигурации Thor предлагает несколько других способов сделать отдельные инструменты более доступными. Вы можете изменить системное меню VFP, добавив целые заголовки меню, добавив подменю к существующим заголовкам меню или добавив инструменты непосредственно к существующим заголовкам меню. Так что если вам нравится использовать меню, но вы обнаружили, что нужные вам инструменты слишком глубоко зарыты в меню инструментов Thor, вы можете поместить их туда, где вам нужно.

Кроме того, вы можете создавать всплывающие меню, которые появляются при нажатии определенной комбинации клавиш. Они похожи на меню, вызываемые правой кнопкой мыши, за исключением того, что они вызываются указанной вами комбинацией клавиш. Эти всплывающие меню могут включать любые выбранные вами инструменты и могут иметь подменю, если вы этого хотите.

Чтобы изменить системное меню VFP или создать всплывающее меню, используйте страницу Menu Definitions формы Thor Configuration. На рисунке 8 показана эта страница после нажатия кнопки Add Menu с выделенным элементом Popup Menus. Чтобы определить всплывающее меню, укажите подсказку (которая появляется только в форме Thor Configuration) и горячую клавишу для всплывающего меню. Затем используйте кнопку Add Tool, чтобы добавить один или несколько инструментов во всплывающее меню.

На рисунке 9 определено новое всплывающее меню и добавлены два инструмента Thor, которые работают с управляющими структурами. На рисунке 10 показано новое всплывающее меню над окном кода.

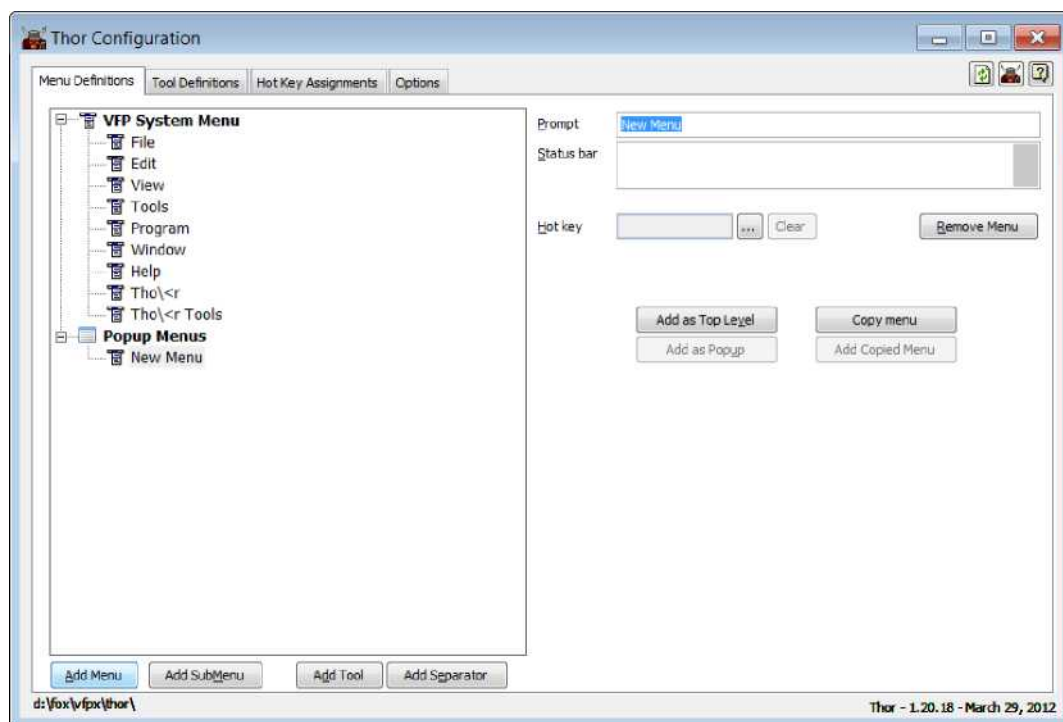


Рисунок 8. На странице «Определения меню» формы конфигурации Thor вы можете добавлять элементы в системное меню VFP и создавать собственные всплывающие меню.

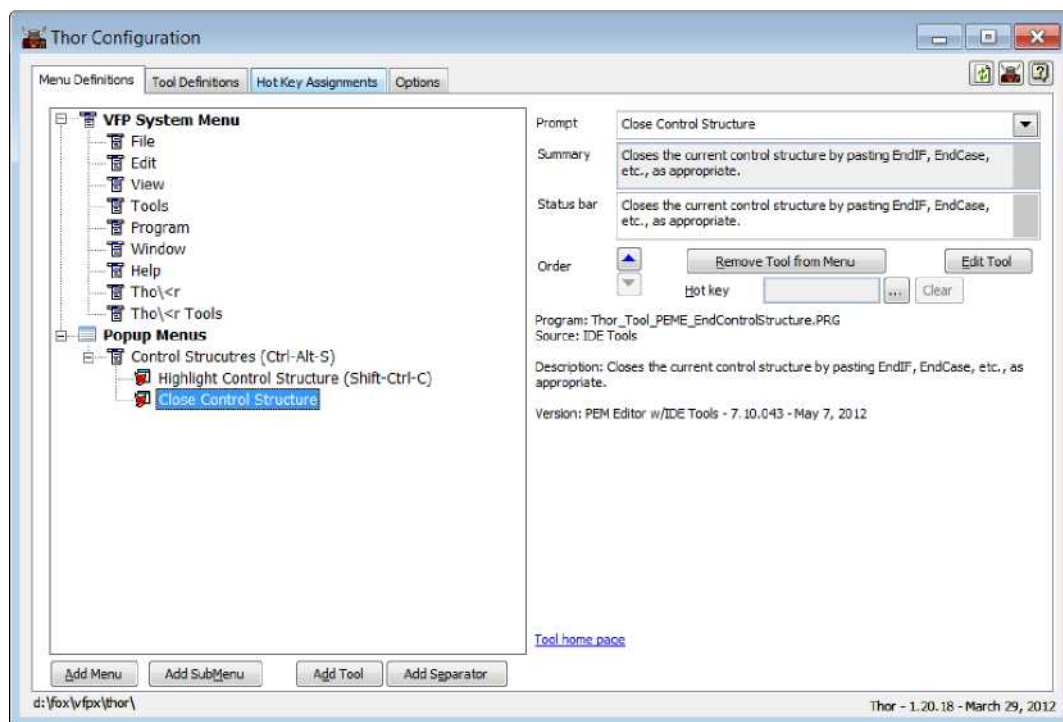


Рисунок 9. Определено всплывающее меню, содержащее два инструмента, связанных с управляющими структурами.

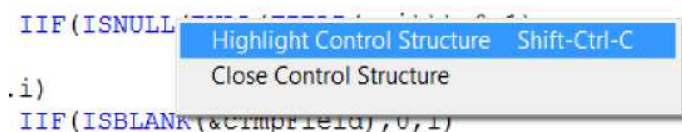


Рисунок 10. При использовании сочетания клавиш для всплывающего меню оно появляется в месте наведения мыши.

Возможность добавлять горячие клавиши, изменять системное меню VFP и определять всплывающие меню позволяет вам легко решить, какие инструменты Thor вы, скорее всего, будете использовать, а затем сделать их легкодоступными.

Редактировать родительские и содержащие классы

Меню: Parent Classes | Edit Parent and Containing Classes

Этот инструмент, скорее всего, заставит людей использовать Thor. Одной из слабостей VFP является то, что при редактировании формы или класса вы не можете открыть ни один класс в иерархии наследования любого члена редактируемого класса. Например, если вы работаете над списком в форме и понимаете, что вам нужно изменить какой-то код или настройку в классе, на котором основан список, вам придется закрыть форму, а затем открыть класс списка. Когда вы закончите вносить изменения, вам придется закрыть класс списка и снова открыть форму.

Хотя Thor не может изменить это правило, он может облегчить работу с ним. Именно для этого и предназначен этот инструмент. Он открывает небольшую форму, показывающую классы в наследии выбранного объекта, и позволяет вам открыть любой из них (после закрытия текущего класса или формы). Когда вы это делаете, форма остается открытой, позволяя вам легко перейти к другим перечисленным

классам; она также содержит кнопку для повторного открытия формы или класса, которые вы изначально редактировали.

Например, Object Inspector, который я построил, основан на наборе классов, опубликованных Дагом Хеннигом. Основная форма, показанная на рисунке 11, включает класс-контейнер `sfTreeViewExplorer`, который включает в себя `treeview` и несколько других элементов управления. Когда я запускаю инструмент Edit Parent and Containing Classes с выбранным объектом, открывается форма, показанная на рисунке 12. Она показывает, что элемент управления включен в класс `Form` с именем `sfExplorerFormTreeView` и что элемент управления основан на классе `sfTreeViewExplorer`, который наследует от `sfTreeViewCursor`, который наследует от `sfTreeViewContainer`, который наследует от `sfContainer`.

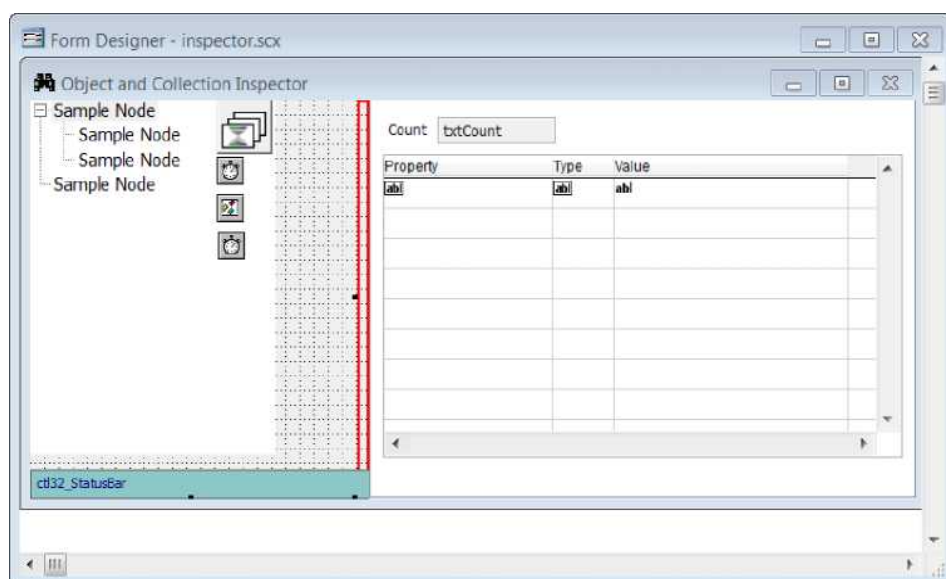


Рисунок 11. Инспектор объектов основан на формах проводника Дуга Хеннига. Здесь выбран объект-контейнер `sfTreeViewExplorer`.

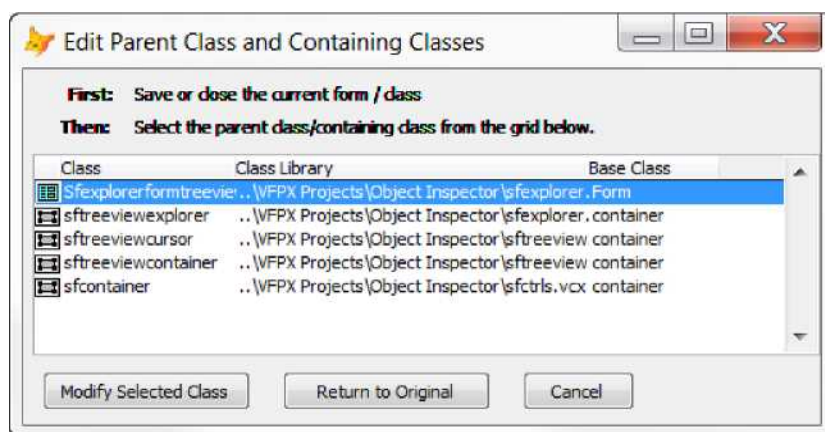


Рисунок 12. Инструмент «Изменить родительский класс и содержащие его классы» открывает эту форму, позволяя вам перемещаться между классами в наследии объекта.

Форма на самом деле не указывает, какие классы находятся в иерархии наследования, а какие в иерархии контейнеров. Оба включены в иерархию контейнеров, показанную первой. На рисунке 13 показан другой пример из той же формы. Перед тем, как открыть инструмент на этот раз, был выбран элемент управления таймером внутри контейнера `treeview`. Форма показывает, что таймер содержится в форме класса `sfExplorerFormTreeView`, а также содержится в контейнере класса `sfTreeViewExplorer`,

который наследует от `sfTreeViewCursor` и `sfTreeViewContainer`. Наконец, таймер основан на классе `sfTimer`.

Одна вещь на рисунке 13 может немного сбивать с толку. При взгляде на контейнер для таймера, почему список останавливается на `sfTreeViewContainer`? Почему он не идет обратно к `sfContainer`, как на рисунке 12. Ответ в том, что инструмент не отслеживает иерархию наследования для содержащихся классов. Он показывает вам каждый класс в этой иерархии, который содержит указанный объект. Итак, в этом примере `sfContainer` не включает таймер; он был добавлен в `sfTreeViewContainer`, а затем унаследован подклассами этого класса, `sfTreeViewCursor` и `sfTreeViewExplorer`.

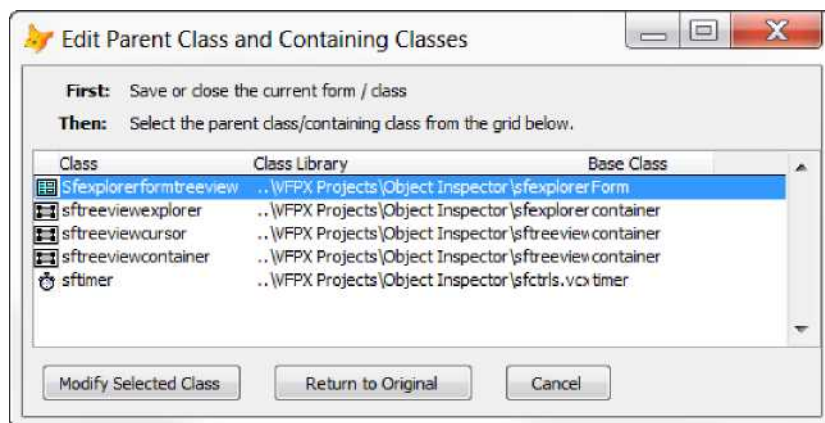


Рисунок 13. На этот раз при запуске этого инструмента был выбран один из таймеров внутри `sfTreeViewExplorer`.

Как показывают инструкции в верхней части формы, прежде чем я смогу открыть любой из этих классов, мне нужно закрыть форму Inspector. Но затем я могу выбрать класс и нажать `Modify Selected Class`, чтобы открыть его. После того, как я закончу редактирование и закрою `Class Designer`, я могу нажать `Return to Original`, чтобы снова открыть форму Inspector в том же состоянии, в котором я ее оставил.

Копирование и вставка РЕМ

Хотя вы можете скопировать объект в VFP и вставить его в другую форму или класс, нет собственного способа придать одному объекту те же значения свойств и код метода, что и другому объекту. Пара инструментов Thor предоставляет такую возможность.

Вы когда-нибудь настраивали элемент управления на форме, устанавливали свойства и добавляли код, а затем осознавали, что на самом деле хотите использовать другой тип элемента управления? Например, вы могли использовать текстовое поле и понять, что вам нужно поле редактирования, или начать с выпадающего списка и понять, что вам действительно нужно поле со списком, или вы можете захотеть переключиться между двумя классами, основанными на одном и том же базовом классе. Настройка нового элемента управления для соответствия старому – это своего рода боль, поскольку вам нужно пройти по одному измененному РЕМ (свойства, события и методы) в `Property Sheet` для исходного объекта и для каждого переключаться на новый объект, чтобы задать его соответствующий РЕМ.

Thor избавляет от этой боли. Он также позволяет легко настроить элемент управления на одной форме для соответствия или частичного соответствия элементу управления на другой форме, а также вставить родительский класс в иерархию наследования.

Копировать (для сравнения и вставки)

Меню: Objects and PEMs | Copy / Paste | Copy (for comparing and pasting)

Этот инструмент позволяет вам подобрать все измененные PEM для объекта и сохранить их в специальном буфере обмена. Чтобы использовать его, вы просто выбираете объект и выбираете этот инструмент.

Сам по себе этот инструмент не очень полезен. Он предназначен для вставки свойств и кода метода (описано в следующем разделе) или сравнения со скопированным объектом (описано далее в этом документе).

Вставить свойства и код метода

Меню: Objects and PEMs | Copy / Paste | Paste properties and method code

Этот инструмент позволяет вам устанавливать свойства и методы для значений, которые вы ранее скопировали из другого объекта. Вы можете выбрать, какие PEM копировать и следует ли добавлять свойства и методы, которых нет в целевом объекте.

Целевой объект не обязательно должен быть основан на том же базовом классе, что и копируемый объект. Это делает этот инструмент отличным для тех ситуаций, когда вы хотите переключаться между двумя похожими классами (хотя инструмент Re-Define Parent Class, описанный далее в этом документе, на самом деле предоставляет более прямой способ выполнить эту задачу).

На рисунке 14 показана форма, которая позволяет пользователю выбрать продукт (из таблицы Northwind Products) с помощью комбо. После тестирования я могу обнаружить, что список обеспечивает лучший пользовательский интерфейс. С помощью этой пары инструментов я могу перейти от одного к другому всего за несколько шагов.

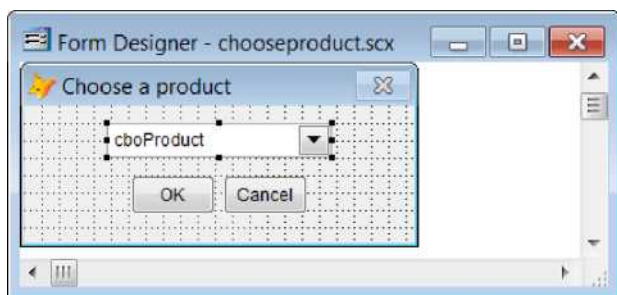


Рисунок 14. В этой форме используется выпадающий список, позволяющий пользователю выбрать продукт.

После выбора комбинации выберите инструмент Копировать (для сравнения и вставки), чтобы забрать его свойства и код метода. Затем перетащите список на форму (предположительно, изменив размер формы). Затем используйте инструмент Вставить свойства и код метода; появится форма, показанная на рисунке 15.

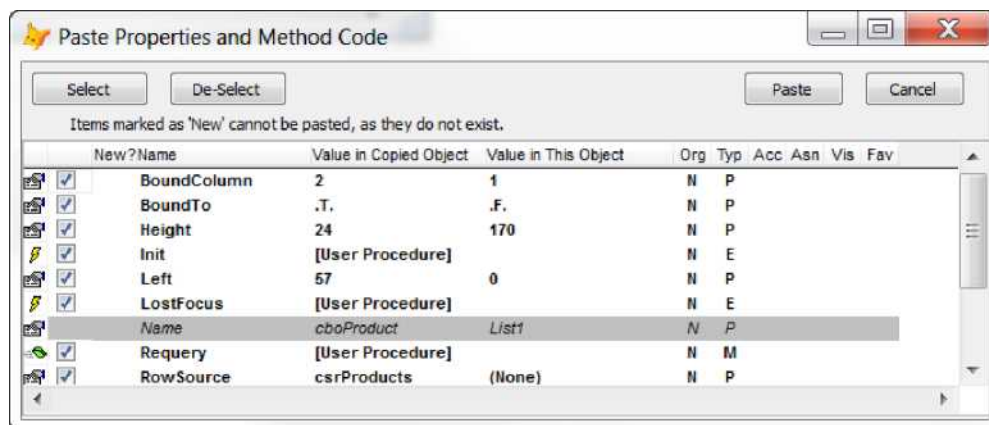


Рисунок 15. Инструмент «Вставить свойства и код метода» отображает эту форму, которая позволяет выбрать, какие РЕМ-файлы копируются в целевой объект.

Вы можете выбрать, какие РЕМ копировать в цель. В этом примере вы, вероятно, не захотите копировать свойство Height, поскольку весь смысл использования списка – показать больше продуктов одновременно. После того, как вы выбрали нужные, нажмите кнопку Paste.

Когда цель не является формой или классом (то есть классом, который редактируется в Class Designer), свойства, которые существуют в источнике, но не в цели, помечаются как New и не могут быть скопированы. Однако, когда цель является формой или классом, флажок спрашивает, следует ли создавать РЕМ, которые не существуют.

Предположим, мы решили создать комбинированный класс для выбора продуктов. После использования инструмента Copy (for comparing and pasting) мы можем создать новый комбинированный класс, а затем использовать Paste properties and method code. Форма, которая появляется в этом случае, показана на рисунке 16. (Конечно, если вы просто хотите превратить элемент управления на форме в класс как есть, вы можете использовать собственную опцию VFP «Сохранить как класс». Этот инструмент лучше подходит для случаев, когда класс уже существует, и вы хотите внести целый набор изменений, чтобы он соответствовал существующему элементу управления.)

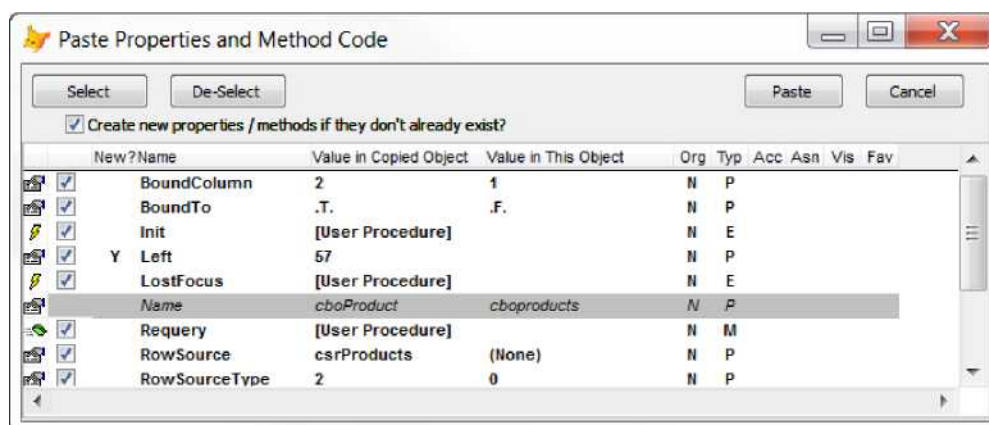


Рисунок 16. При вставке РЕМ-файлов в класс или форму вы можете добавлять свойства и методы «на лету».

Сравнение объектов

Один из моих любимых инструментов, не относящихся к VFP, – Beyond Compare, который позволяет мне сравнивать папки и файлы, чтобы находить различия между ними. С дополнением VFP Фрэнка Переса (<http://pfsolutions-mi.com/Product/VFP2Text>).

Я даже могу сравнивать классы и формы VFP. Но это неудобный способ делать это, когда я нахожусь в процессе их редактирования, поскольку для просмотра файлов с помощью Beyond Compare нужно закрывать их.

Thor включает в себя два разных инструмента для сравнения объектов; оба довольно полезны. Compare with Parent Class позволяет вам увидеть, какие свойства класса имеют значения, отличные от значений по умолчанию, и показывает значения родительского класса для тех же свойств. Compare with Copy Object позволяет вам сравнить два несвязанных объекта.

Сравнить с родительским классом

Меню: Parent Classes | Compare with Parent Class

Хотя таблица свойств VFP позволяет увидеть, какие РЕМ объекта были изменены по сравнению со значениями по умолчанию (и даже позволяет увидеть только нестандартные РЕМ), она не предоставляет простого способа узнать, каковы эти значения в родительском классе.

Инструмент Compare with Parent Class открывает отдельную форму, которая показывает каждое свойство, событие и метод, отличные от значений по умолчанию, в выбранном объекте, а также их значение в выбранном объекте и в родительском классе. Он также указывает те свойства, которые имеют одинаковое значение в обоих местах. Наконец, он позволяет сбросить любой из отображаемых РЕМ выбранного объекта к их значениям по умолчанию.

На рисунке 17 показана форма DataNav.SCX из Solution Samples, которые поставляются с VFP. Выбран объект _tablenav. На рисунке 18 показана форма, открытая инструментом Compare with Parent Class (который, очевидно, является вариацией того, который используется для инструмента Paste properties and method code). Он указывает, что в Property Sheet формы заданы три свойства и пять методов. Один из них, Name, такой же на форме, как и в классе.

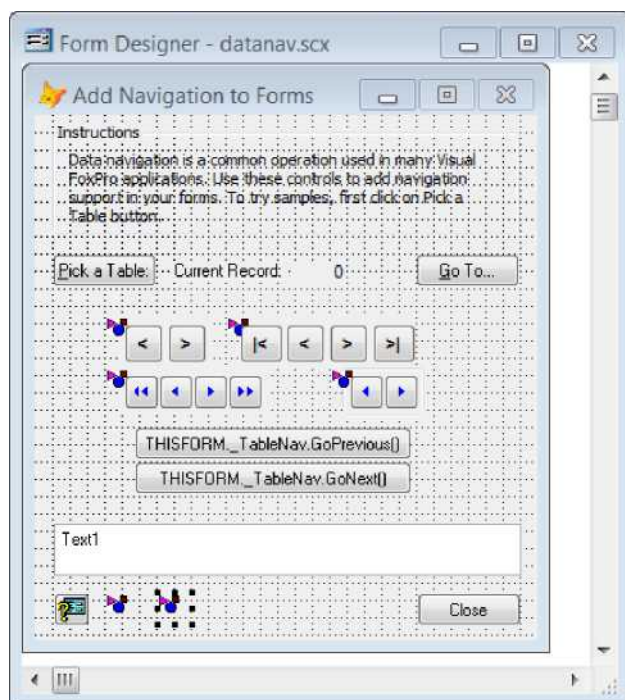


Рисунок 17. В этой форме из примеров решений выбран объект на основе класса `_tablenav` (из базовых классов FoxPro).

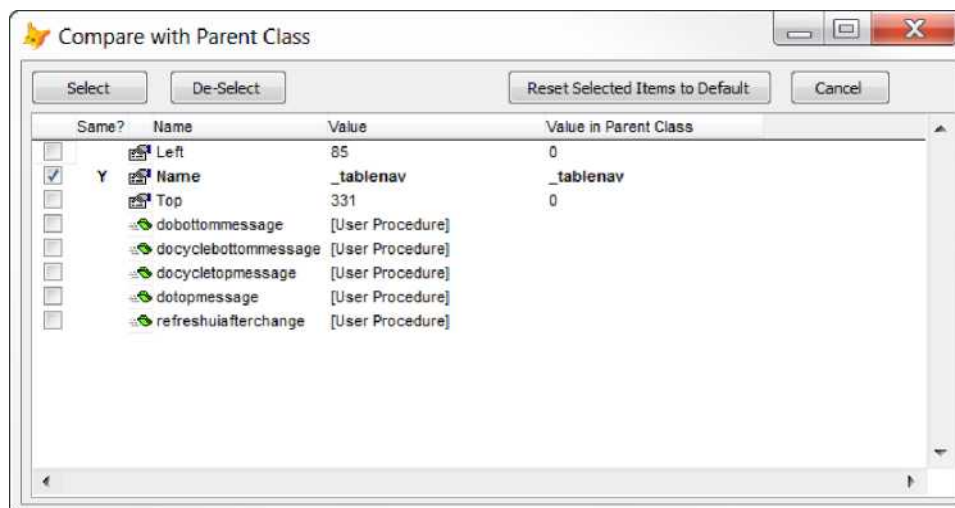


Рисунок 18. Инструмент «Сравнить с родительским классом» показывает только те свойства, значения которых не являются значениями по умолчанию.

Вы можете использовать флажки, чтобы выбрать некоторые или все показанные РЕМ, а затем нажать **Reset Selected Items to Default**, чтобы очистить эти РЕМ в выбранном объекте. Свойства, которые имеют то же значение, что и в классе, автоматически отмечаются.

Кнопки «Выбрать» и «Отменить выбор» открывают раскрывающееся меню (показано на рисунке 19), в котором можно указать тип элемента, который следует отметить или снять отметку.

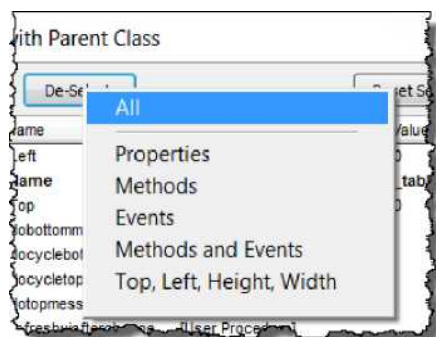


Рисунок 19. Кнопки «Выбрать» и «Отменить выбор» инструмента «Compare with Parent Class» позволяют выбрать тип элемента, который следует отметить или снять отметку.

Есть причина, по которой этот инструмент указывает свойства, которые имеют то же значение, что и в их родительском классе. Каждое свойство, заданное в Property Sheet, должно быть оценено и назначено во время инициализации формы или другого класса. Поэтому наличие свойств, которые явно заданы как значение по умолчанию из родительского класса, может замедлить выполнение (хотя вы вряд ли заметите это, если только нет большого количества объектов с большим количеством таких свойств).

Сравнить со скопированным объектом

Меню: Objects and PEMS | Copy / Paste | Compare with copied object

Второй инструмент Thor для сравнения кода требует некоторых пояснений. Он является частью набора инструментов копирования и вставки для классов, описанных в разделе «Копирование и вставка PEM» ранее в этом документе.

Чтобы использовать этот инструмент, вы должны сначала выбрать объект и использовать инструмент Copy (for comparing and pasting), который находится в той же ветке меню Thor. Затем выберите объект, с которым вы хотите сравнить первый объект, и выберите этот инструмент.

Для демонстрации мы рассмотрим тот же объект, что и для Compare with Parent Class, хотя этот инструмент в данном случае является лучшим выбором. Начиная с формы DataNav, щелкните объект _tablenav и выберите Thor Tools | Objects and PEMs | Copy / Paste | Copy (for comparing and pasting). Теперь закройте форму и откройте класс _tablenav из _table.vcx в папке FFC (или, что еще лучше, используйте инструмент Edit Parent and Containing Classes, чтобы открыть его). Когда вы выбираете этот инструмент из меню, появляется форма, показанная на рисунке 20.

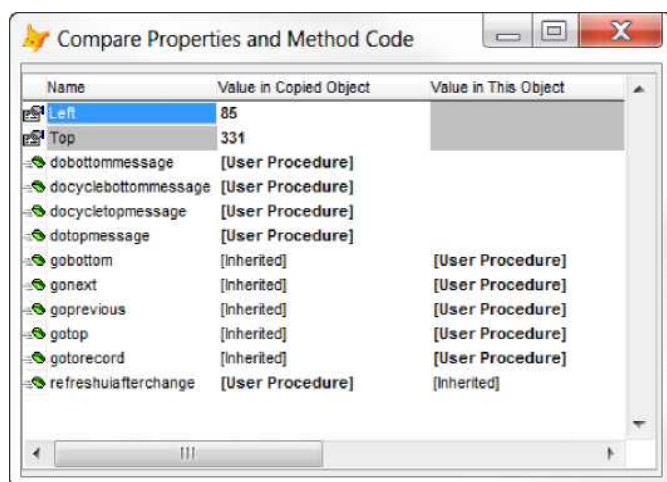


Рисунок 20. Эта форма появляется при использовании инструмента «Сравнить со скопированным объектом». Она показывает PEM, которые различаются в двух объектах.

PEMS с серым фоном не существуют в объекте, где столбец значений также серый. В примере класс _tablenav не имеет свойств Left и Top, но при перетаскивании на форму они добавляются. Методы, где столбец пустой (например, метод dobottommessage в примере), существуют, но не имеют кода нигде в иерархии наследования до этого класса.

Этот инструмент кажется особенно полезным в ситуациях, когда один экземпляр класса не работает, а другие экземпляры работают. Вы можете сравнить PEM работающего экземпляра с PEM неработающего, чтобы увидеть, что вы сделали по-другому.

Также представляется полезным выяснить, могут ли два существующих класса продуктивно использовать общий родительский класс. (Когда это возможно, инструмент Paste properties and method code предоставляет вам простой способ быстро приступить к созданию родительского класса.)

Переопределить родительский класс

Меню: Parent Class | Re-Define Parent Class

Изменение класса, на котором основан элемент управления, всегда было немного мучительным. Class Browser позволяет это сделать, и существуют различные сторонние инструменты, такие как HackCX, которые также справляются с этой задачей. Но все эти подходы требуют, чтобы вы закрыли класс или форму, над которыми вы работаете, чтобы инструмент мог открыть их. Этот инструмент позволяет вам вносить изменения, не закрывая форму или класс.

Чтобы использовать его, выберите объект, родительский класс которого вы хотите изменить, и выберите этот инструмент из меню. Появится форма, показанная на рисунке 21 ; вы используете ее для поиска нового родительского класса. Форма позволяет вам указать тип класса, который вы ищете (указан стрелкой), и где его искать. Типом может быть определенный базовый класс или «<All>». Для области действия вы можете указать папку (как на рисунке) или выбрать любой из проектов в списке MRU. Вы также можете указать все или часть имени нужного вам класса; в отличие от обычного сравнения строк VFP, указанная вами часть может быть в любом месте имени класса. Аналогично, вы можете указать часть имени файла, чтобы ограничить поиск.

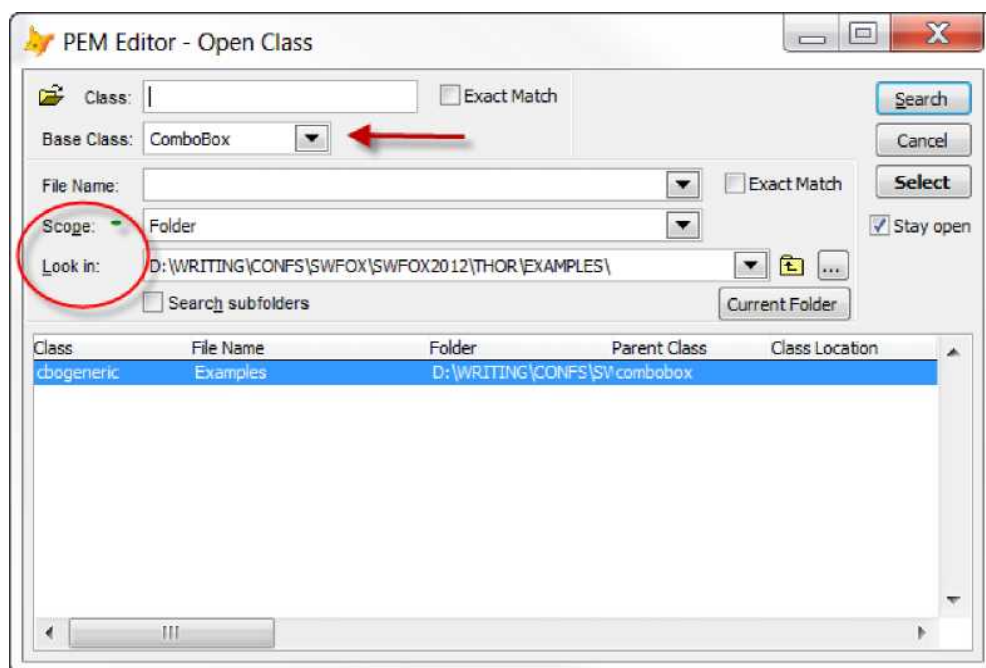


Рисунок 21. Найдите и выберите новый родительский класс для объекта, используя эту форму.

Когда вы настроили тип класса и где искать, нажмите кнопку Search, и сетка в нижней части формы покажет все классы, которые соответствуют вашим спецификациям. Выберите нужный и нажмите кнопку Select, чтобы изменить родительский объект выбранного объекта на этот класс.

Откроется тот же диалог, который используется для инструментов Paste properties and method code и Compare with parent class, как на рисунке 22, показывая вам свойства и методы, которые отличаются в новом родительском и существующем объектах. Решите, какие из измененных значений вы хотите сохранить, и нажмите Paste.

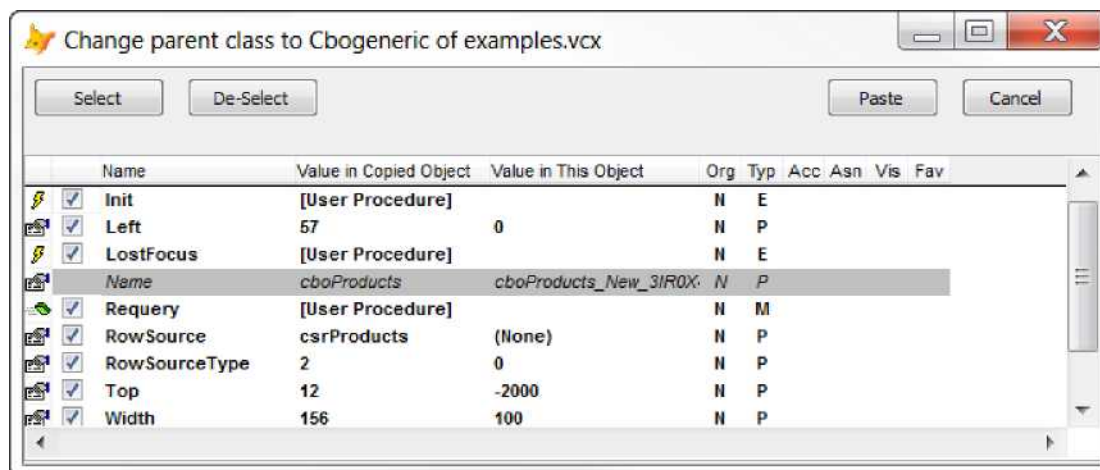


Рисунок 22. После выбора нового родительского класса у вас есть возможность решить, какие значения свойств и код метода следует сохранить.

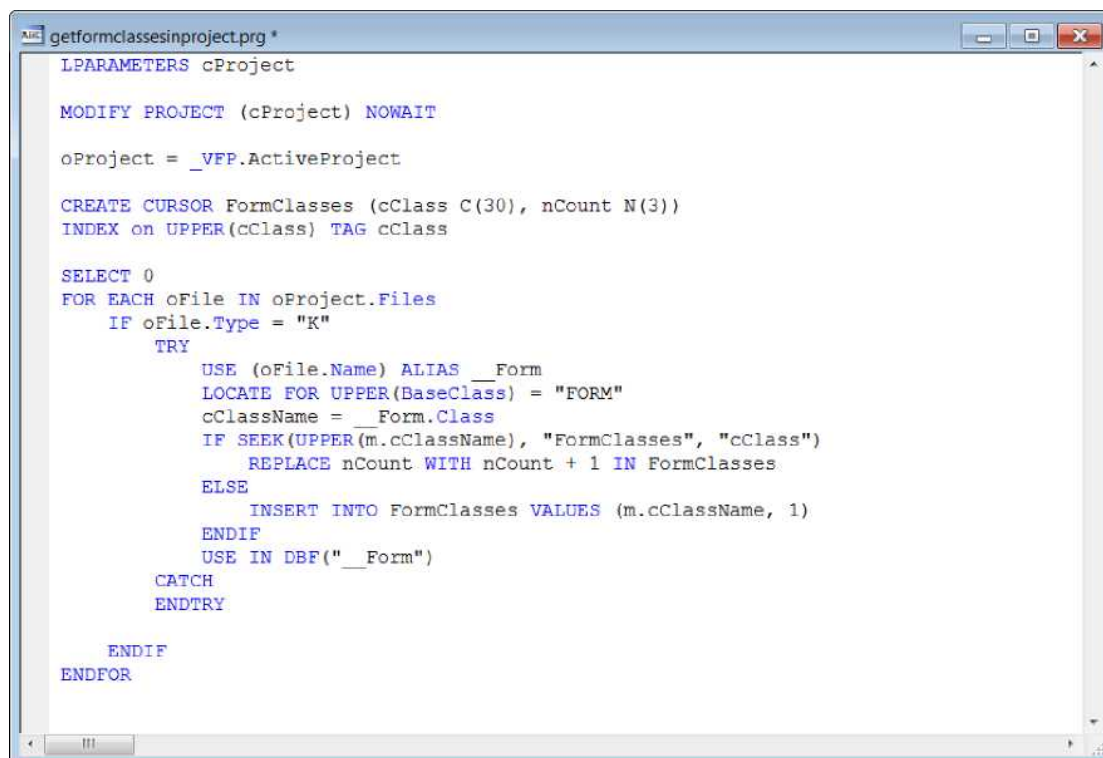
Создать локальные переменные

Меню: Code | Create LOCALs

Я всегда знал, что объявление всех переменных, используемых в процедуре, является лучшей практикой, и с тех пор, как FoxPro превратился в VFP, объявление всех переменных локальными является лучшим выбором. Но важность объявлений действительно была доведена до меня одним проектом. Он включал приложение VFP, которое предоставляло пользовательский интерфейс, но также создавало экземпляр объекта VFP COM. У объекта COM был таймер, и когда таймер срабатывал, объект COM мог вызывать методы объекта приложения основного приложения. Конечно, эти вызовы прерывали все, что происходило в основном приложении.

При тестировании этого кода мы столкнулись с некоторыми очень странными ошибками, при этом код большую часть времени работал, но время от времени вел себя довольно странно. В конце концов, мы поняли, что многие проблемы были вызваны наличием необъявленных переменных (которые по умолчанию являются приватными, а не локальными). Процедуры, вызываемые кодом таймера, использовали некоторые из тех же имен переменных, и когда переменные не были локальными, фактически изменяли значения переменных в процедуре, которая была прервана таймером. Как только мы гарантировали, что каждая переменная в каждом методе была объявлена локальной, многие проблемы исчезли.

Если бы у меня тогда был Thor! Инструмент Create Locals от Thor берет любое окно редактирования кода и добавляет необходимые локальные объявления. На рисунке 23 показан небольшой блок кода (который выгружает список форм в проекте в курсор); он использует несколько переменных, но ни одна из них не объявлена.



```
getformclassesinproject.prg *
LPARAMETERS cProject

MODIFY PROJECT (cProject) NOWAIT

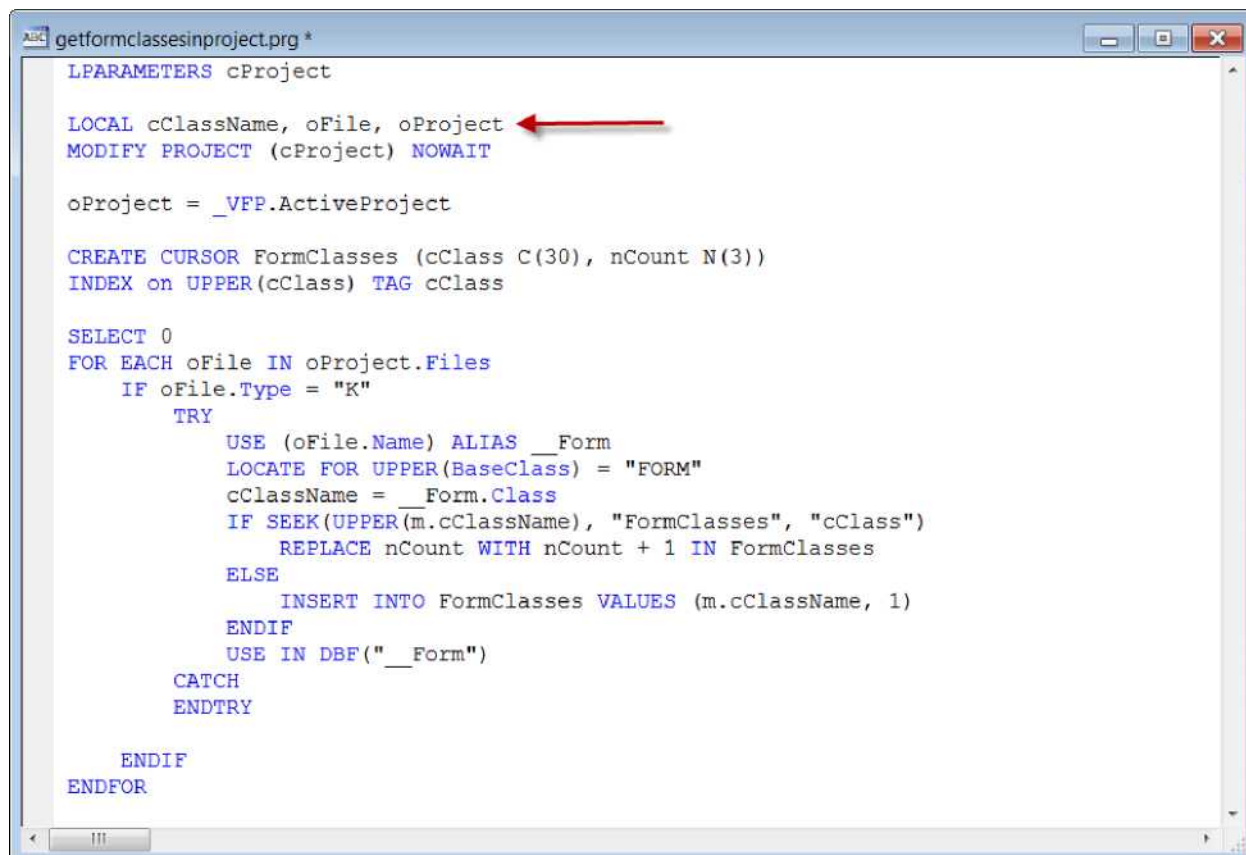
oProject = _VFP.ActiveProject

CREATE CURSOR FormClasses (cClass C(30), nCount N(3))
INDEX on UPPER(cClass) TAG cClass

SELECT 0
FOR EACH oFile IN oProject.Files
    IF oFile.Type = "K"
        TRY
            USE (oFile.Name) ALIAS __Form
            LOCATE FOR UPPER(BaseClass) = "FORM"
            cClassName = __Form.Class
            IF SEEK(UPPER(m.cClassName), "FormClasses", "cClass")
                REPLACE nCount WITH nCount + 1 IN FormClasses
            ELSE
                INSERT INTO FormClasses VALUES (m.cClassName, 1)
            ENDIF
            USE IN DBF("__Form")
        CATCH
        ENDTRY
    ENDIF
ENDFOR
```

Рисунок 23. Этот блок кода использует несколько необъявленных переменных.

На рисунке 24 показан тот же блок кода после запуска инструмента Create Locals; стрелка указывает на локальные объявления.



```
getformclassesinproject.prg *
LPARAMETERS cProject

LOCAL cClassName, oFile, oProject
MODIFY PROJECT (cProject) NOWAIT

oProject = _VFP.ActiveProject

CREATE CURSOR FormClasses (cClass C(30), nCount N(3))
INDEX on UPPER(cClass) TAG cClass

SELECT 0
FOR EACH oFile IN oProject.Files
    IF oFile.Type = "K"
        TRY
            USE (oFile.Name) ALIAS __Form
            LOCATE FOR UPPER(BaseClass) = "FORM"
            cClassName = __Form.Class
            IF SEEK(UPPER(m.cClassName), "FormClasses", "cClass")
                REPLACE nCount WITH nCount + 1 IN FormClasses
            ELSE
                INSERT INTO FormClasses VALUES (m.cClassName, 1)
            ENDIF
            USE IN DBF("__Form")
        CATCH
        ENDTRY
    ENDIF
ENDFOR
```

Рисунок 24. После запуска инструмента Create Locals тот же блок имеет объявления переменных.

Вы можете управлять некоторыми аспектами поведения этого инструмента с помощью вкладки «Параметры» инструмента конфигурации Thor.

На рисунке 25 показана вкладка «Параметры» с настройками для Create Locals, а на рисунке 26 показаны варианты в раскрывающемся списке Selection of variables. Этот раскрывающийся список позволяет указать, следует ли объявлять все переменные или только те, чьи имена начинаются со строчной буквы «l», префикса, используемого для локальных переменных в венгерском соглашении об именовании. Поскольку я использую другое соглашение об именовании (добавляя переменным префикс с указанием их типа, но не «l»), я предпочитаю вариант «Все переменные, объединенные».

Большинство других вариантов должны быть очевидными или легко понятными после небольшого тестирования. Последний флажок, Create LOCALs as part of BeautifyX, определяет, добавляются ли локальные объявления при использовании инструмента BeautifyX, который является заменой родного Beautify VFP.

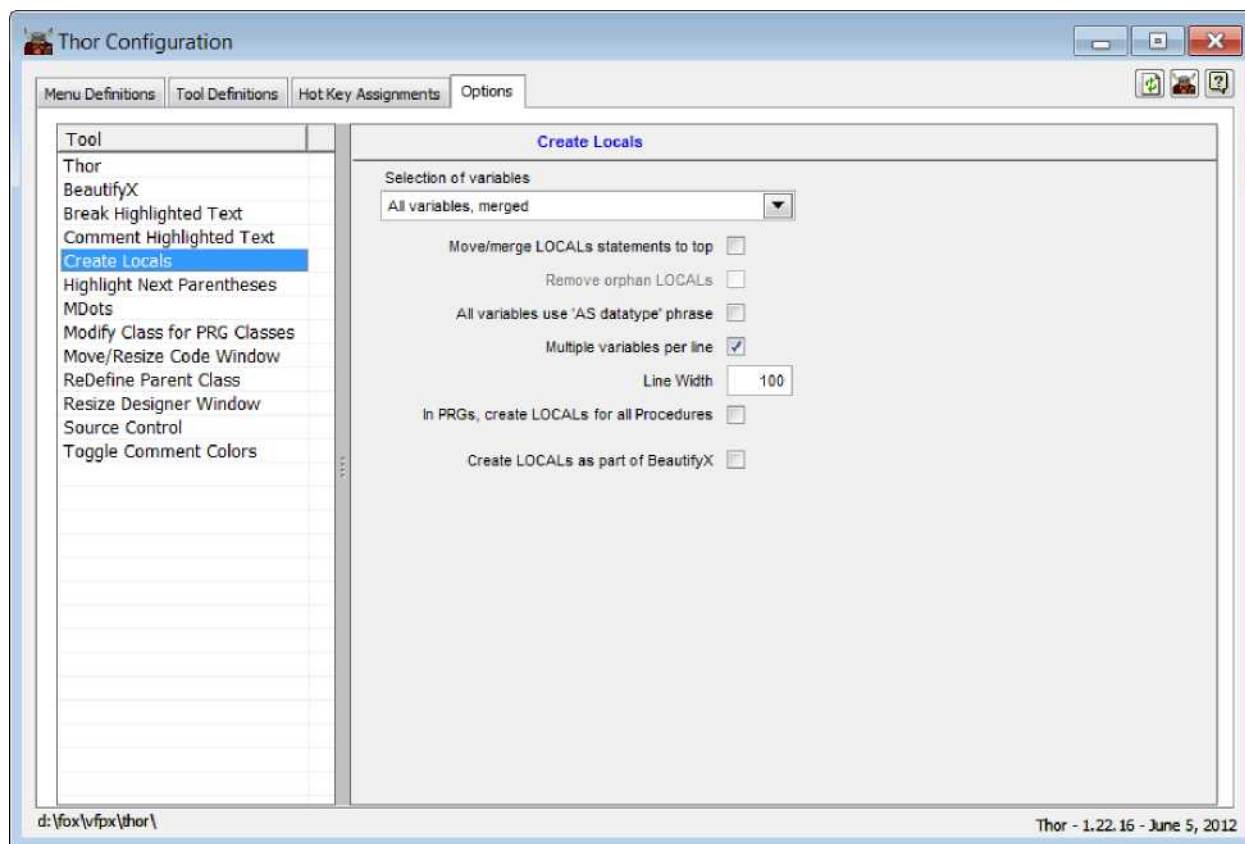


Рисунок 25. На этой странице можно определить, как будет работать инструмент «Создать локальные объекты».

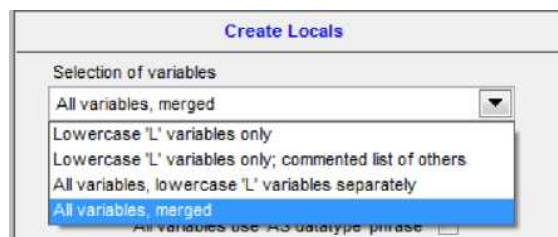


Рисунок 26. Create Locals можно применять только к подмножеству переменных на основе их имен.

Если даже после попытки различных настроек вам не нравится способ создания локальных объявлений, Thor предлагает вам максимальную гибкость. Вы можете создать собственную версию кода для инструмента. Для этого выберите More | Manage Plug-Ins в меню Thor. В появившейся форме найдите CreateLocalsStatements и нажмите кнопку Create рядом с ним. Это откроет программу, содержащую текущий код,

используемый для фактического создания локальных объявлений. Вы можете изменить его и сохранить (он автоматически находится в нужном месте), и с этого момента инструмент будет использовать вашу измененную версию.

Добавить MDots к именам переменных

Меню: Code | MDots | Add MDots to variable names

Как и в случае с объявлением всех переменных локальными, я всегда знал, что все обращения к переменной должны предваряться «m.» (часто пишется как «MDot»), чтобы гарантировать, что VFP смотрит на переменную, а не на поле с тем же именем. На самом деле, я все чаще вспоминаю, что нужно вставлять их в свой код, но иногда все еще забываю. Этот инструмент Thor отслеживает все места, которые я пропустил.

На рисунке 27 показан блок кода, который не использует нотацию MDot. На рисунке 28 показан тот же код после использования этого инструмента. Несколько изменений обведены кружками.



Рисунок 27. В этом коде не используется нотация MDOT, чтобы исключить конфликты между переменными и именами полей.

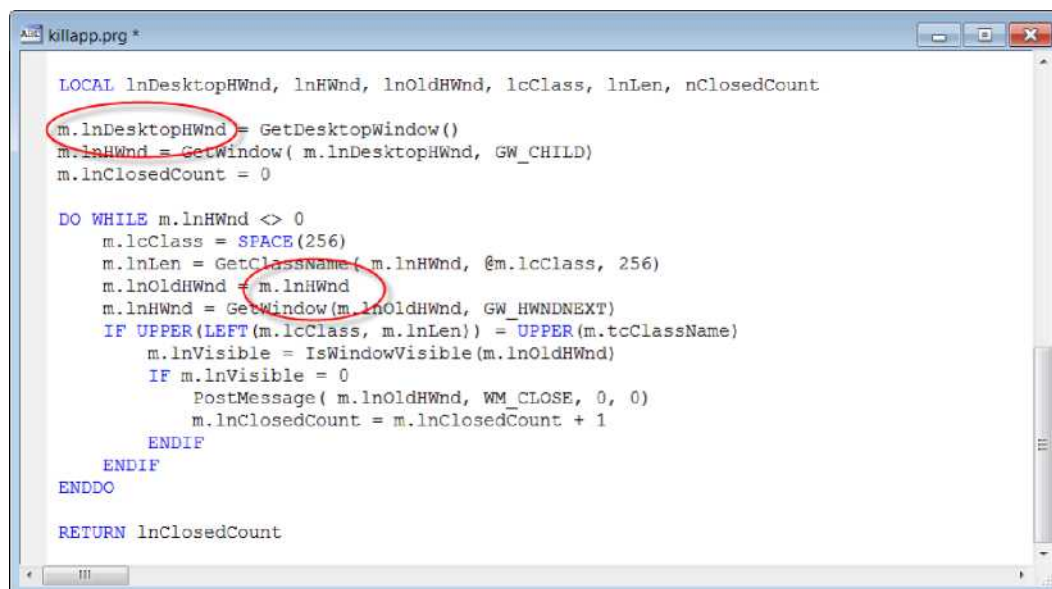


Рисунок 28. После использования инструмента «Добавить MDot к именам переменных» в коде с рисунка 27 перед всеми ссылками на переменные появляется «m.».

По умолчанию этот инструмент добавляет MDots в нескольких местах, где они не нужны, например, в левой части оператора присваивания. Однако вы можете управлять этим поведением с помощью вкладки «Параметры» формы конфигурации Thor, показанной на рисунке 29. На рисунке 30 показан тот же блок кода при использовании инструмента, настроенного на рисунке 29.

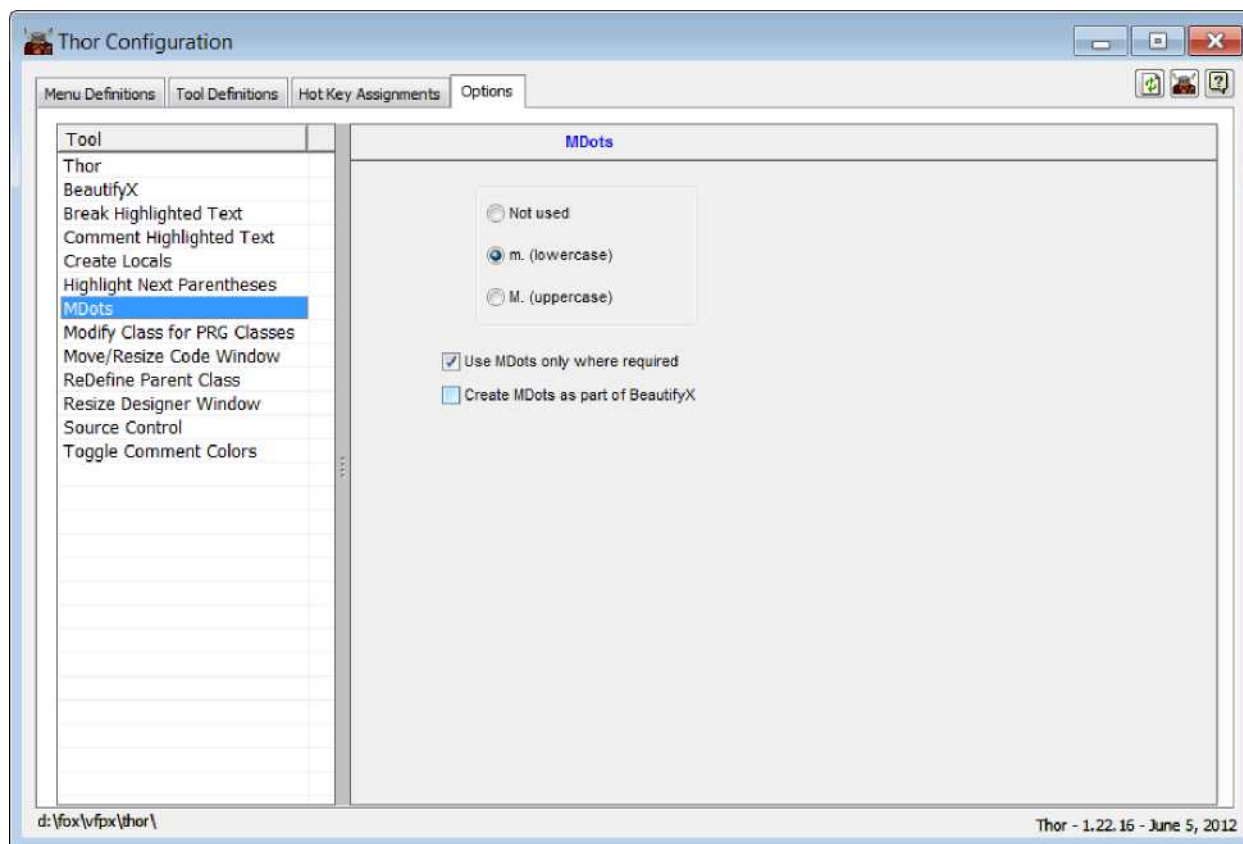


Рисунок 29. Thor предлагает опции для различных инструментов. Здесь вы можете определить, использует ли инструмент Add MDots строчные или заглавные буквы и помещает ли он MDots перед всеми использованиями переменных или только там, где это требуется, чтобы избежать конфликта с именами полей.

```

LOCAL lnDesktopHwnd, lnHwnd, lnOldHwnd, lcClass, lnLen, nClosedCount

lnDesktopHwnd = GetDesktopWindow()
lnHwnd = GetWindow( m.lnDesktopHwnd, GW_CHILD)
lnClosedCount = 0

DO WHILE m.lnHwnd <> 0
    lcClass = SPACE(256)
    lnLen = GetClassName( m.lnHwnd, @m.lcClass, 256)
    lnOldHwnd = m.lnHwnd
    lnHwnd = GetWindow(m.lnOldHwnd, GW_HWNDNEXT)
    IF UPPER(LEFT(m.lcClass, m.lnLen)) = UPPER(m.tcClassName)
        lnVisible = IsWindowVisible(m.lnOldHwnd)
        IF m.lnVisible = 0
            PostMessage( m.lnOldHwnd, WM_CLOSE, 0, 0)
            lnClosedCount = m.lnClosedCount + 1
        ENDIF
    ENDIF
ENDDO

RETURN m.lnClosedCount

```

Рисунок 30. После изменения опции включения MDots только там, где это необходимо, переменные в левой части оператора присваивания больше не получают префикс MDot.

Выделение скобок

Меню: Code | Highlighting text | Highlight parentheses

Это еще один инструмент, который особенно удобен, когда я исследую код, написанный кем-то другим, или старый код, который я давно не видел. Он также отлично подходит для тех случаев, когда вы получаете синтаксическую ошибку и не можете понять, в чем проблема. Когда вы запускаете этот инструмент, он просматривает обе стороны от позиции курсора, чтобы найти соответствующую пару круглых скобок, и выделяет совпадающие круглые скобки и весь код между ними. На рисунке 31 показана строка кода с вложенными скобками; курсор находится внутри внутреннего блока. На рисунке 32 показан тот же блок после использования инструмента.

```

cTmpField = field(m.i)
gtotal = m.gtotal + IIF(ISBLANK(&cTmpField),0,1)
--

```

Рисунок 31. Курсор здесь находится внутри скобок.

```

cTmpField = field(m.i)
gtotal = m.gtotal + IIF(ISBLANK(&cTmpField),0,1)
--

```

Рисунок 32. Выделение скобок находит первую содержащую пару скобок и выделяет содержащийся в ней код.

Подобно Highlight Control Structure, использование этого инструмента многократно повторяет это действие; Рисунок 33 показывает тот же блок кода после использования инструмента дважды. Как ни странно, если больше нет пар скобок, содержащих выделенный код, инструмент выделяет весь код в окне редактирования.

```
cTmpField = field(m.i)
gtotal = m.gtotal + IIF(ISBLANK(&cTmpField),0,1)
--
```

Рисунок 33. Каждое последующее использование скобок «Выделение» перемещает текст на одну пару скобок.

Подсветка скобок достаточно умна, чтобы сделать все правильно, когда совпадающая пара скобок содержит другие скобки. Например, на рисунке 34 курсор установлен на функцию FIELD(), но не внутри ее скобок. На рисунке 35 показан результат использования инструмента. Правильное сопряжение найдено.

```
IF THIS.shownulls
    gtotal = m.gtotal + IIF(ISNULL(EVAL(FIELD(m.i))),0,1)
ELSE
```

Рисунок 34. Здесь начальное положение курсора не находится во внутренней паре скобок.

```
IF THIS.shownulls
    gtotal = m.gtotal + IIF(ISNULL(EVAL(FIELD(m.i))),0,1)
ELSE
```

Рисунок 35. Инструмент «Выделение скобок» работает правильно, даже если начальное положение курсора не находится внутри самых внутренних скобок.

Дерево документа

Меню: Applications | Document Treeview

Когда в VFP 7 был добавлен Document View, он предоставил нам простой способ навигации внутри файлов, содержащих несколько процедур. Это было значительное улучшение по сравнению со списком процедур и функций, который он заменил. Я использую его все время при работе с классами, созданными в коде (PRG).

Но Document View никогда не был особенно полезен для форм или классов, хранящихся в библиотеке классов (VCX). Хотя он перечисляет каждый метод, содержащий код, он не дает никакого ощущения структуры, и в загруженной форме или классе он может быть довольно загроможденным.

Инструмент Document Treeview лучше подходит для визуальных классов (и, по сути, не работает для кода). Document Treeview основан на основном выпадающем списке редактора РЕМ, но вы можете использовать его как автономный инструмент. Он показывает объекты в форме или классе и те из их методов, которые содержат код. Как следует из названия, он использует элемент управления Treeview для организации информации, поэтому вы можете развернуть или свернуть любой раздел. Его можно изменять в размере, а также закреплять. Кроме того, вы можете контролировать то, что отображается в любое время.

На рисунке 36 показано дерево документов для формы IMover из примеров решений (находятся в папке Samples\Solution\Controls\Lists\); некоторые объекты и методы находятся вне поля зрения выше и ниже показанного раздела.

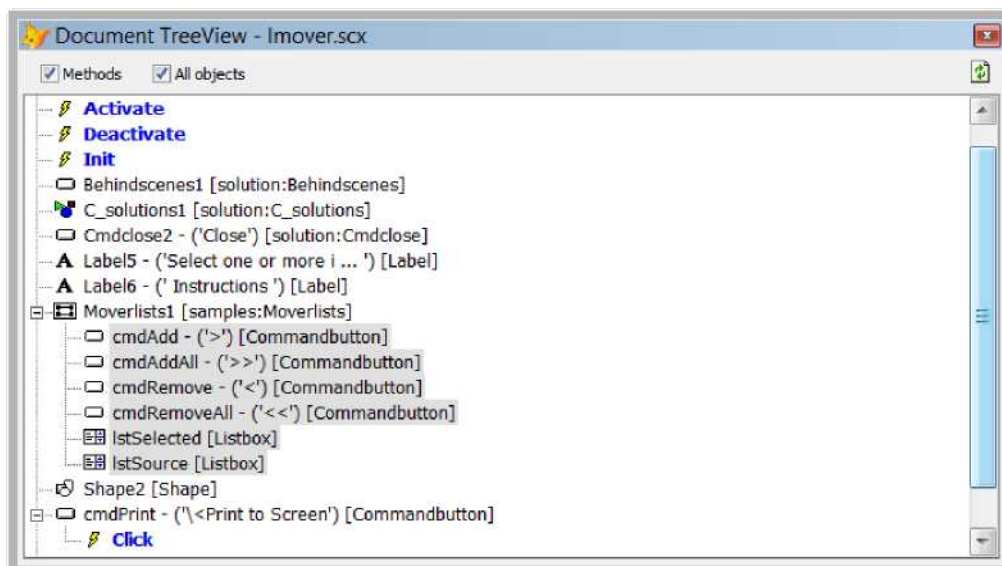


Рисунок 36. В дереве документа отображаются объекты класса или формы, а также любые методы, содержащие код.

Цвета в Document Treeview помогают быстро понять отображение. Имена объектов показаны черным цветом. Если объект был унаследован вместе со своим контейнером от родительского класса (как кнопки внутри Moverlists1 в примере), его фоновый цвет серый. Имена методов показаны синим цветом, если они содержат код на этом уровне; если они содержат только унаследованный код, они показаны серым цветом. Все имена методов выделены жирным шрифтом, что дает еще один способ отличить их от имен объектов.

Нажатие на объект делает его текущим объектом в PEM Editor, если этот инструмент открыт. Часто он также выбирает этот объект в Form или Class Designer и делает его текущим объектом в Property Sheet; однако бывают случаи, когда это невозможно сделать программно.

Вы можете щелкнуть по любому методу, чтобы открыть его для редактирования; это похоже на Document View. Но Document Treeview предлагает другую возможность. Используйте Ctrl+Click на методе, и откроется окно, показывающее вам весь код для этого метода на всех уровнях иерархии наследования. Конечно, вы не можете редактировать код там, но это гораздо более простой способ увидеть весь код, чем что-либо, предлагаемое изначально. (Другой инструмент Thor, Code Listings, также позволяет вам видеть весь код в одном месте.)

На рисунке 37 показано окно, которое открывается при нажатии Ctrl+Click на методе DisplayProperties главной формы для моего инструмента Object Inspector. Сначала отображается код, добавленный на уровне формы, затем код, унаследованный от класса sfExplorerFormTreeview, а затем код для этого метода в sfExplorerForm, уровне, на котором был определен метод.

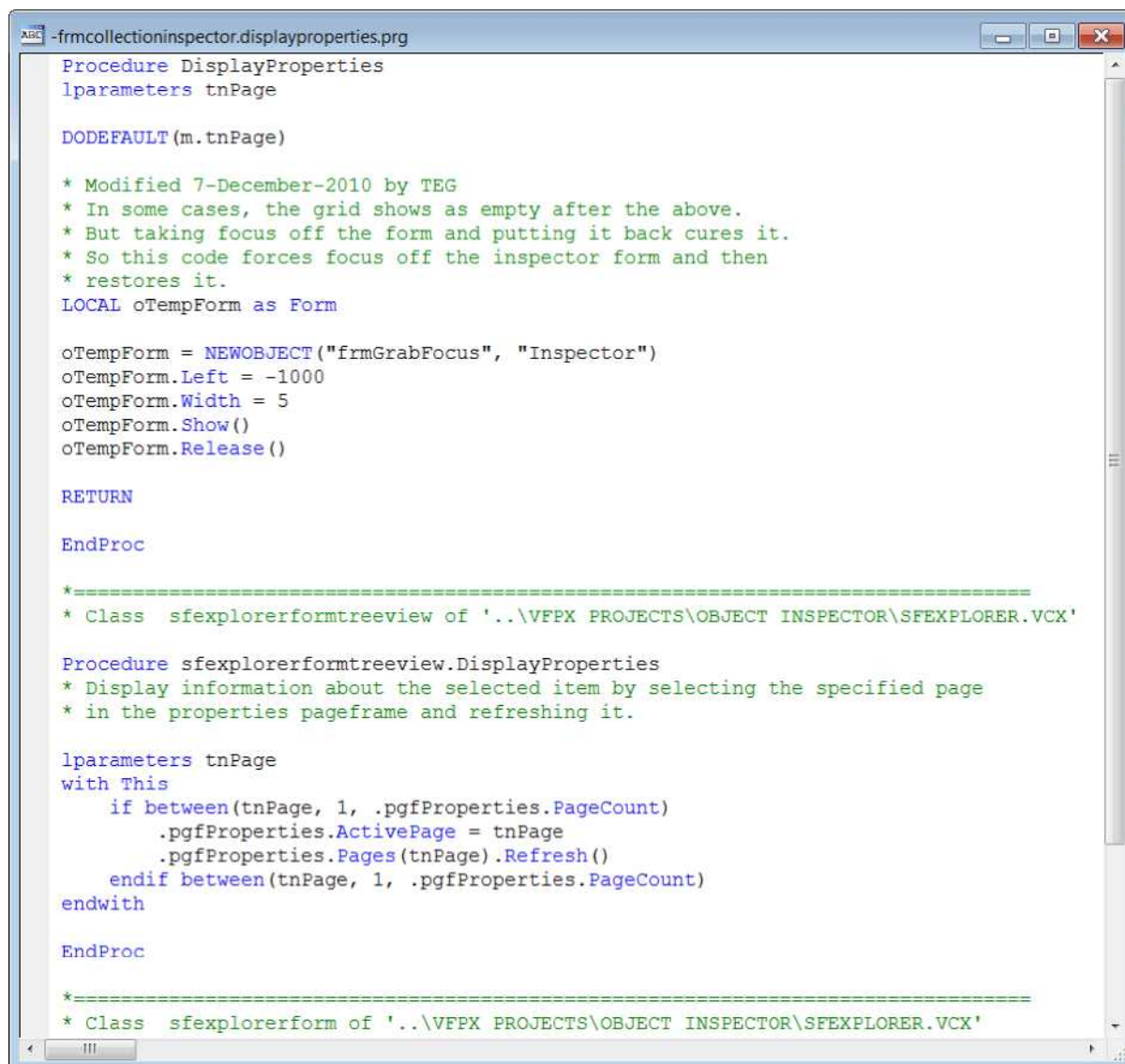


Рисунок 37. Использование Ctrl+щелчок по методу в Method Treeview показывает весь код для этого метода со всех уровней иерархии наследования.

Существует несколько способов управления тем, что отображается в Document Treeview. Два флажка над элементом управления Treeview предлагают три варианта. Когда оба флажка отмечены, вы видите все объекты, а также методы, которые имеют код, как на рисунке 36. Если вы снимите флажок All objects, вы увидите только те объекты, которые имеют какой-либо код, а также методы, которые содержат код. На рисунке 38 показан Document Treeview для формы Imover со снятым флажком All objects. Как вы можете видеть, этот параметр позволяет легко увидеть, где есть код. Наконец, если вы снимите флажок Methods, флажок All objects будет скрыт, и вы увидите все объекты в форме или классе. На рисунке 39 показан Document Treeview для формы Imover со снятым флажком Methods.

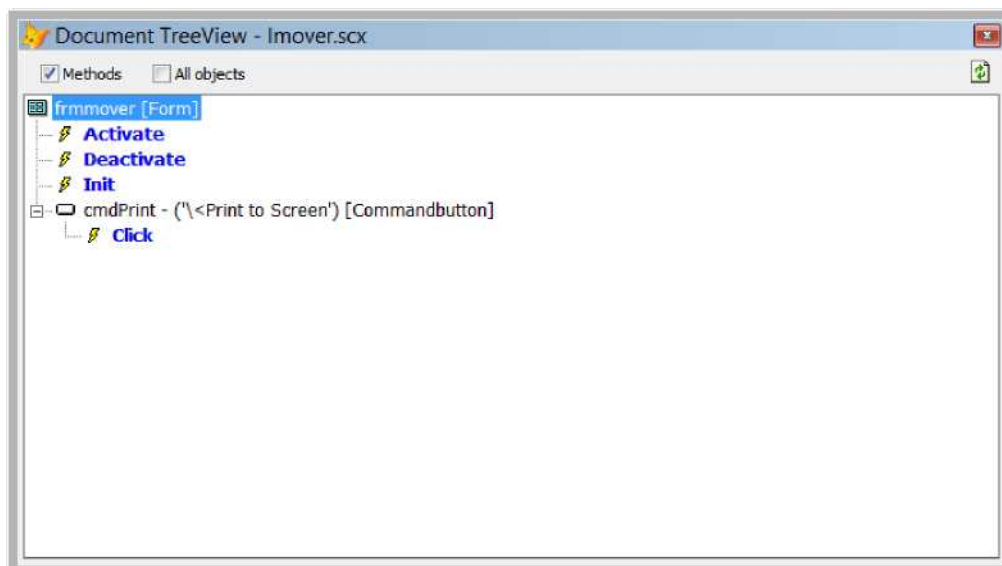


Рисунок 38. Вы можете настроить Document Treeview на отображение только тех объектов, которые содержат код.

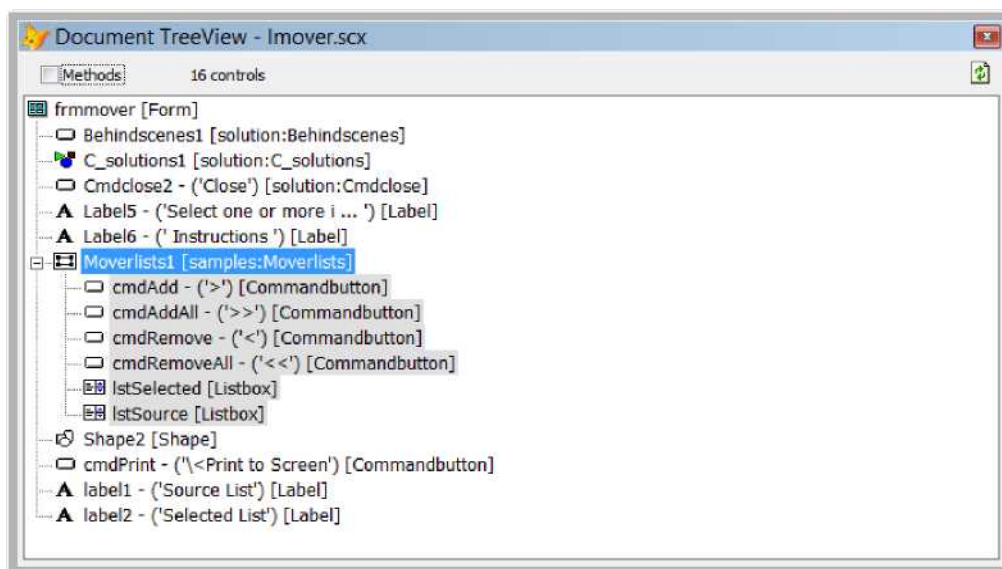


Рисунок 39. Снятие флажка «Методы» указывает Document Treeview отображать все объекты и не отображать методы.

Вы также можете управлять отображением с помощью контекстного меню инструмента. На рисунке 40 показано основное меню, как оно появляется при щелчке правой кнопкой мыши по фону инструмента. (Щелчок правой кнопкой мыши по узлу, конечно, включает опции, специфичные для узла.) Первые три элемента под разделителем на рисунке 40 определяют, какая информация включена для объекта. Первый элемент, Show Caption, ControlSource, указывает, хотите ли вы включить Caption или ControlSource объекта в отображение, чтобы помочь вам определить, какой это объект. На рисунке 39, например, вы можете видеть, что Caption для кнопки cmdClose2 – «Close». Как и ожидалось, параметры Show class name и Show class library and name являются взаимоисключающими; отметка одного снимает отметку другого. Однако вы можете снять отметку с обоих и не показывать информацию о классе объекта.

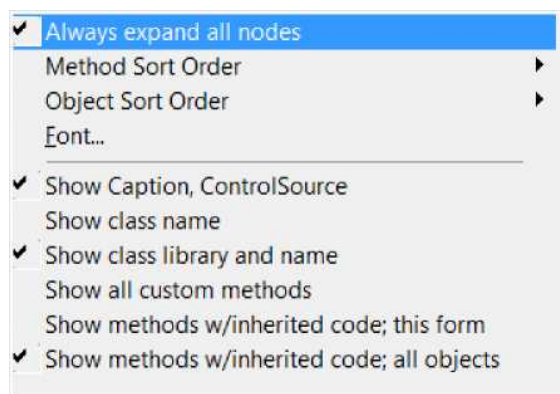


Рисунок 40. Контекстное меню Document Treeview включает несколько способов определения того, что отображается.

Последние три элемента в этом разделе контекстного меню определяют, какие методы отображаются. По умолчанию Document Treeview показывает только методы, имеющие код на этом уровне иерархии. Установите флажок Show all custom methods, чтобы также включить все методы, добавленные к текущему объекту. Установка флажка Show methods w/inherited code; this form также отображает методы текущего объекта, имеющие код выше в иерархии наследования. Установите флажок Show methods w/inherited code; all objects, чтобы также включить методы содержащихся объектов, имеющих код где-то в иерархии наследования.

Элементы над разделителем управляют внешним видом, но не содержимым Document Treeview. Первый элемент, Always expand all nodes, определяет, будут ли развернуты все элементы в Treeview при открытии инструмента или при открытии формы или класса с открытым инструментом. Следующие два элемента определяют порядок, в котором методы и объекты, соответственно, появляются в списке. Для методов единственными вариантами выбора являются сортировка по алфавиту с учетом регистра или сортировка по алфавиту без учета регистра. Объекты предлагают гораздо больше вариантов выбора, в дополнение к этим двум; список показан на рисунке 41. Вы можете оставить их несортированными, в этом случае они появятся в том же порядке, что и в Property Sheet. Вы также можете сортировать по TabIndex или сверху вниз или слева направо.

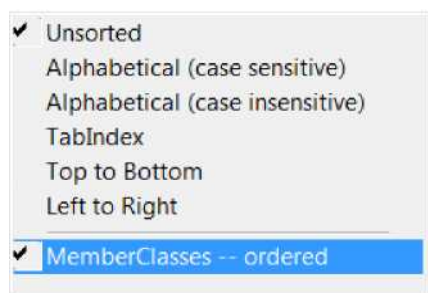


Рисунок 41. Существует несколько способов сортировки объектов в Document Treeview.

По умолчанию объекты в элементах управления, имеющих свойство MemberClass, сортируются в порядке создания, в том же порядке, в котором они отображаются в Property Sheet. Выбор последнего параметра на рисунке 41, MemberClasses -- ordered, сортирует их в соответствии с указанным порядком, например PageOrder или ColumnOrder.

Добавление инструментов в Thor

У меня есть несколько небольших инструментов, которые я написал в разное время, в основном без пользовательского интерфейса. Например, один из них проходит по проекту и заполняет курсор именами всех классов форм, используемых в этом проекте (то есть тех, на которых основана хотя бы одна форма). Все это занимает около 40 строк кода. Проблема с этими небольшими инструментами в том, что когда я хочу их использовать, мне приходится находить их и смотреть на код, чтобы вспомнить как. Затем мне приходится либо убедиться, что код находится в пути для поиска файлов, либо указать полный путь, чтобы запустить инструмент.

Одним из критериев дизайна Thor было сделать так, чтобы людям было легко добавлять инструменты и делиться ими. Таким образом, вы можете взять все маленькие инструменты, такие как тот, что я описал выше, и вставить их в меню инструментов Thor, чтобы они были под рукой. Среди других преимуществ, Thor управляет кодом, поэтому мне не нужно беспокоиться о путях.

Чтобы добавить инструмент в Thor, откройте форму конфигурации Thor (Thor | Configure в меню) и щелкните вкладку Tool Definitions. Щелкните кнопку Create Tool, чтобы открыть диалоговое окно Create Tool, показанное на рисунке 42. После того, как вы дадите инструменту имя с помощью текстового поля, которому предшествует «Thor_Tool_», щелкните кнопку Create, чтобы открыть файл шаблона для инструмента.

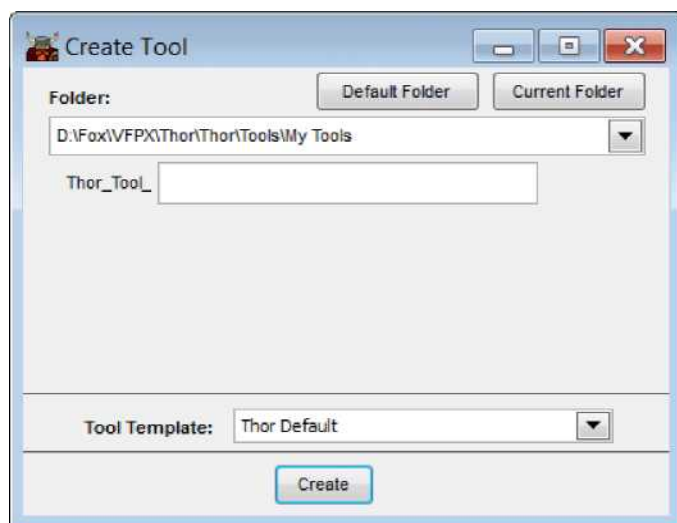


Рисунок 42. Чтобы добавить новый инструмент в Thor, укажите имя инструмента в этом диалоговом окне и нажмите «Создать».

Инструменты Thor требуют определенного формата, который предоставляется шаблоном. Шаблон по умолчанию показан в листинге 2, слегка переформатирован для соответствия странице. Основная часть шаблона предоставляет место для предоставления информации Thor об этом инструменте; чтобы создать инструмент, заполните одно или несколько перечисленных свойств. Prompt является обязательным и содержит подсказку, которая будет отображаться в меню инструментов Thor. Описание отображается только в диалоговом окне конфигурации Thor.

Листинг 2. Шаблон Thor по умолчанию показывает именно то, что требуется для создания инструмента Thor.

```
lparameters liParam1
*****
*****
```

```

* Standard prefix for all tools for Thor, allowing this tool to
* tell Thor about itself.

If Pcount() = 1 ;
    And 'O' = Vartype (lxParam1) ;
    And 'thorinfo' == Lower (lxParam1.Class)

    With lxParam1

        * Required
        .Prompt = 'Prompt for the tool' && used in menus

        * Optional
        Text to .Description NoShow && a description for the tool
Enter a description for the tool here
        ENDTEXT
        .StatusBarText = ''

        * These are used to group and sort tools when they are displayed
        * in menus or the Thor form
        .Source      = '' && where did this tool come from? Your own initials,
                        && for instance
        .Category    = '' && creates categorization of tools; defaults
                        && to .Source if empty
        .Sort        = 0 && the sort order for all items from the same Category

        * For public tools, such as PEM Editor, etc.
        .Version     = '' && e.g., 'Version 7, May 18, 2011'
        .Author      = ''
        .Link         = '' && link to a page for this tool
        .VideoLink   = '' && link to a video for this tool

    Endwith

    Return lxParam1
Endif

If Pcount() = 0
    Do ToolCode
Else
    Do ToolCode With lxParam1
Endif

Return

*****
*****
* Normal processing for this tool begins here.
Procedure ToolCode
Lparameters lxParam1
EndProc

```

Свойства Source, Category и Sort позволяют указать, где инструмент отображается в меню Thor Tools. Если указано Category, инструмент отображается в этой группе; вы можете указать несколько уровней в меню, разделив элементы вертикальной чертой («|»). Например, чтобы добавить элемент в группу Misc. в меню Code, укажите «Code|Misc.».

Если Category пусто, значение в Source указывает подменю, в котором появляется инструмент. Вы можете использовать свои инициалы или название своей компании, чтобы сгруппировать все свои собственные инструменты вместе.

Свойство Sort определяет положение данного элемента в указанном подменю.

Последний набор свойств в шаблоне актуален только для инструментов, которые совместно используются сообществом VFP. В листинге 3 показан набор свойств для инструмента, чтобы получить список используемых классов форм.

Листинг 3. Набор свойств Thor для инструмента «Получить классы форм».

```
* Required
.Prompt = 'Get form classes' && used in menus

* Optional
Text to .Description NoShow && a description for the tool
Fill a cursor with names of the form classes used in a project
EndText
.StatusBarText = ''

* These are used to group and sort tools when they are displayed
* in menus or the Thor form
.Source = 'TEG' && where did this tool come from? Your own initials,
&& for instance
.Category = '' && creates categorization of tools; defaults
&& to .Source if empty
.Sort = 0 && the sort order for all items from the same Category
```

Сердцем инструмента является процедура ToolCode; именно туда вы помещаете код для выполнения задачи. В листинге 4 показан код, добавленный в ToolCode для инструмента Get Form Classes. Как вы можете видеть, он не очень сложен. Он проверяет наличие активного проекта, и если он найден, создается курсор для хранения списка классов форм. Затем код проходит по файлам в проекте. Когда встречается файл формы, он открывается как таблица, и находится запись уровня формы. Если мы уже видели этот класс формы раньше, он просто учитывает этот экземпляр. Если это новый класс формы для нашего списка, запись добавляется в курсор FormClasses. После завершения цикла курсор открывается в окне BROWSE.

Листинг 4. Процедура ToolCode для инструмента Get Form Classes.

```
LOCAL cClassName, oFile, oProject, nOldSelect

IF TYPE("_VFP.ActiveProject") = "U"
    MESSAGEBOX("No active project", 0+48, "Get form classes")
    RETURN
ENDIF

oProject = _VFP.ActiveProject

nOldSelect = SELECT()

IF USED("FormClasses")
    USE IN SELECT("FormClasses")
ENDIF
```

```
CREATE CURSOR FormClasses (cClass C(30), nCount N(3))
INDEX on UPPER(cClass) TAG cClass

SELECT 0
FOR EACH oFile IN oProject.Files
  IF oFile.Type = "K"
    TRY
      USE (oFile.Name) ALIAS __Form
      LOCATE FOR UPPER(BaseClass) = "FORM"
      cClassName = __Form.Class
      IF SEEK(UPPER(m.cClassName), "FormClasses", "cClass")
        REPLACE nCount WITH nCount + 1 IN FormClasses
      ELSE
        INSERT INTO FormClasses VALUES (m.cClassName, 1)
      ENDIF
      USE IN DBF("__Form")
    CATCH
    ENDTRY

  ENDIF
ENDFOR

* Make sure last form was closed.
USE IN SELECT ("__FORM")

SELECT FormClasses
BROWSE NOWAIT
```

После того, как вы указали необходимые свойства и добавили код в процедуру ToolCode, сохраните программу. Она автоматически сохранится в нужном месте с нужным именем.

Чтобы протестировать свой инструмент, либо закройте форму конфигурации Thor, либо нажмите кнопку Thor. Любой из этих вариантов обновит меню и горячие клавиши. После этого новый инструмент будет включен в меню Thor Tools, как показано на рисунке 43.

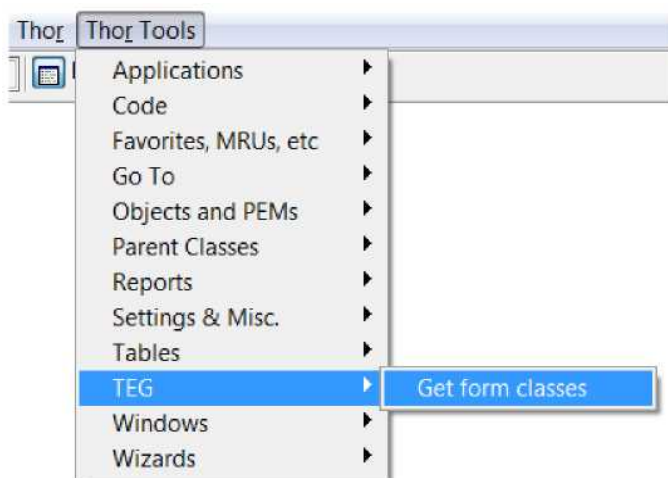


Рисунок 43. После того, как вы закончите определение нового инструмента и обновите Thor, инструмент отобразится в меню.

Использование возможностей Thor в новых инструментах

Хотя вы можете написать новый инструмент со стандартным кодом VFP (как в инструменте Get Form Classes), Thor предлагает большую библиотеку возможностей, которые упрощают написание инструментов. Thor Framework дает вам доступ к нескольким классам, а также к некоторым стандартным элементам, которые вам могут понадобиться.

Чтобы получить доступ к Thor Framework, выберите Thor | Thor Framework в меню. Откроется окно с кодом, который вы можете вырезать и вставить в код вашего инструмента. На рисунке 44 показана часть Thor Framework. (Помните, что Thor Framework достаточно умен, чтобы показать правильный путь для вашей установки. На рисунке 44 показано, где на моем компьютере находятся файлы.)



Рисунок 44. Thor Framework позволяет вам использовать преимущества кода Thor в ваших инструментах.

Хотя полное обсуждение Thor Framework выходит за рамки этой статьи, я покажу небольшой пример того, как вы можете его использовать. (См. (<http://vfpx.codeplex.com/wikipage?title=Thor%20Tools%20Making%20Tools> для получения дополнительной информации о платформе Thor Framework.) Инструмент Get Form Classes, описанный в последнем разделе, работает только при открытом проекте. Одной из крутых функций многих инструментов Thor является то, что они могут видеть, что находится под мышкой, и работать с этим элементом. Это было бы удобной возможностью для этого инструмента – если нет открытого проекта, то найдите имя под мышкой и попытайтесь открыть этот проект.

Чтобы выяснить, как эта возможность была предоставлена, я покопался в коде Thor tools, в котором это есть. Чтобы увидеть, как реализован любой инструмент Thor, откройте форму конфигурации Thor и перейдите на страницу определений инструментов. Выберите интересующий вас инструмент в дереве и нажмите кнопку «Edit Tool», показанную на рисунке 45. Появится форма, показанная на рисунке 46. Если вы просто хотите увидеть, как работает инструмент, выберите вторую кнопку «View this file in Read-Only mode».

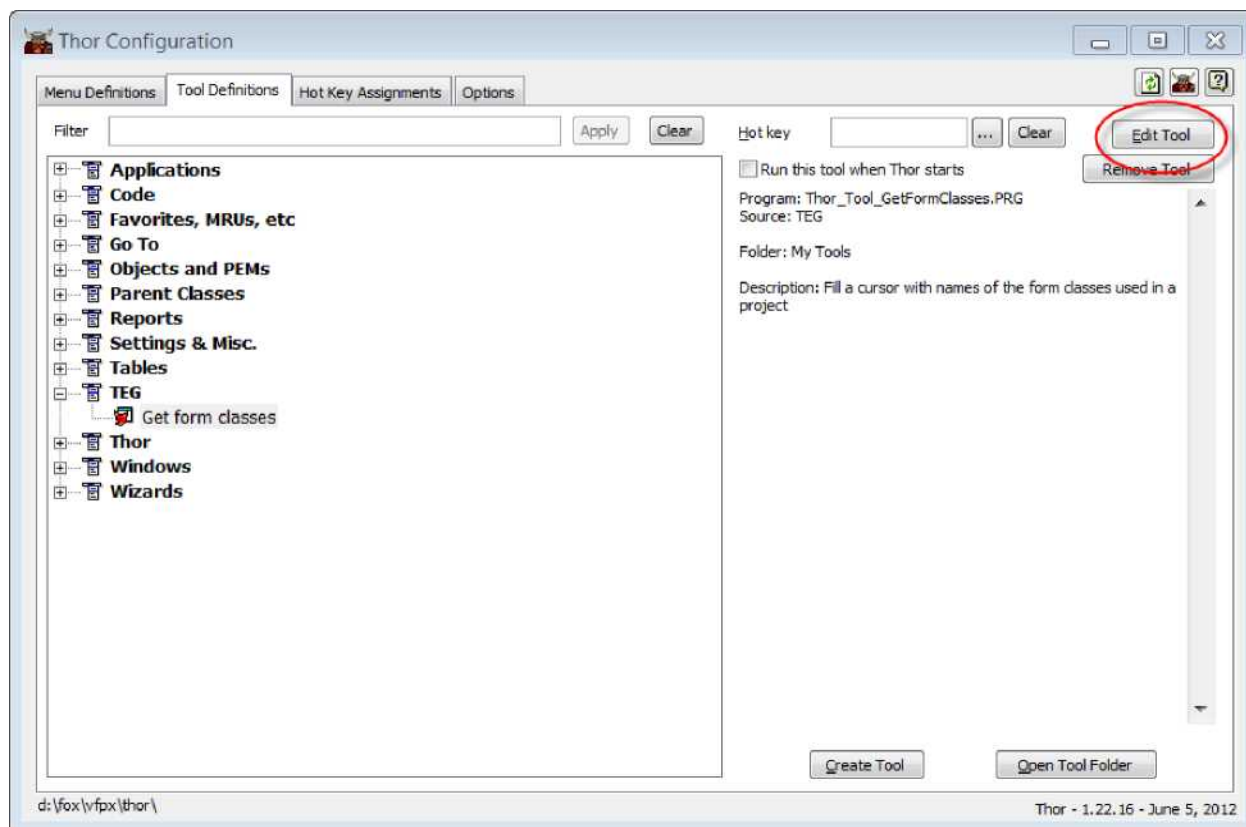


Рисунок 45. Вы можете изменить инструмент, найдя его в форме конфигурации Thor и нажав «Изменить инструмент».

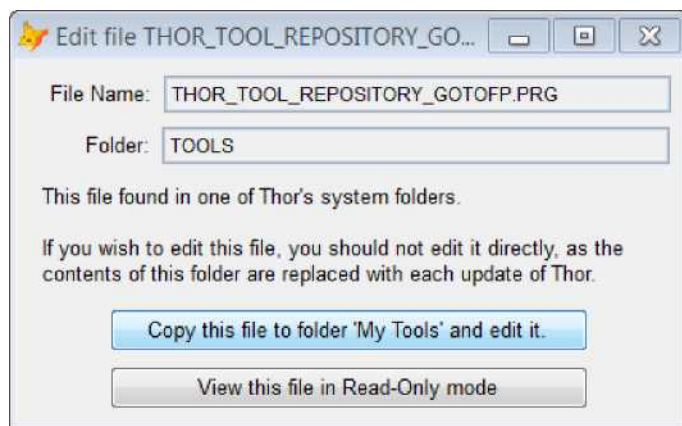


Рисунок 46. Эта форма появляется при попытке открыть любой встроенный инструмент Thor. Чтобы изменить инструмент, выберите первую кнопку. Чтобы просто посмотреть его код, выберите вторую кнопку.

Мне пришлось фактически просмотреть исходный код редактора PEM, чтобы выяснить, как Thor может работать с элементом, указанным под курсором мыши. Когда я посмотрел на код инструмента SuperBrowse, я нашел строки в листинге 5.

Листинг 5. Код, реализующий инструмент SuperBrowse, использует этот код для определения того, какую таблицу следует просматривать.

```
* tools home page =
* http://vfpx.codeplex.com/wikipage?title=thor%20tools%20object
loTools = Execscript (_Screen.cThorDispatcher, ;
    'class= tools from pemeditor')
loTools.UseHighlightedTable (Set ('Datasession'))
```

Я покопался в исходниках редактора РЕМ, чтобы найти метод UseHighlightedTable, который я нашел в классе PemEditor_Tools PEME_Tools.Vcx. В конце концов, я нашел строку кода в листинге 6. Поскольку название метода подразумевает, что он захватывает только выделенный текст, я провел тестирование, чтобы подтвердить, что метод на самом деле захватывает все слово, где находится курсор.

Листинг 6. Эта строка кода, используемая инструментом SuperBrowse, считывает текст под курсором.

```
lcAlias = This.oUtils.oIDEx.GetCurrentHighlightedText()
```

Следующим шагом было изменение кода для моего инструмента Get Form Class. Я открыл код инструмента, как описано выше.

Чтобы настроить возможность чтения текста под курсором, нам нужен класс Tools из фреймворка Thor. Рисунок 44 включает код, который делает этот класс доступным. Вы можете просто скопировать эти три строки из фреймворка и вставить их в соответствующее место в процедуре ToolCode. После создания экземпляра класса Tools мы можем использовать его для получения слова под курсором, а затем попытаться открыть проект с этим именем. Соответствующая часть измененной процедуры ToolCode показана в листинге 7 (слегка переформатирована, чтобы вписаться в страницу).

Листинг 7. Замените код для проверки того, открыт ли проект, на этот код, чтобы инструмент «Получить классы форм» мог работать с проектом, имя которого находится под курсором.

```
IF TYPE("_VFP.ActiveProject") = "U"
    * tools home page =
    * http://vfpx.codeplex.com/wikipage?title=thor%20tools%20object
    Local loTools as Pemeditor_tools of ;
        "d:\fox\vfpx\thor\thor\tools\apps\pem editor\source\peme_tools.vcx"
    loTools = ExecScript(_Screen.cThorDispatcher, "Class= tools from pemeditor")

    lcText = loTools.oUtils.oIDEx.GetCurrentHighlightedText()

    * Now find just the path and filename in the line.

    lProjWasOpen = .F.

    TRY
        MODIFY PROJECT (lcText) NOWAIT
        lSuccess = (TYPE("_VFP.ActiveProject") <> "U")
    CATCH
        lSuccess = .F.
    ENDTRY
ELSE
    lSuccess = .T.
    lProjWasOpen = .T.
ENDIF

IF NOT m.lSuccess
    MESSAGEBOX("No active project", 0+48, "Get form classes")
    RETURN
ENDIF
```

При тестировании я обнаружил, что этот код работает только тогда, когда указанный проект находится в пути для поиска файлов. Я уверен, что Thor Framework позволит мне прочитать весь путь проекта под курсором, но я пока не разобрался, как именно.

Настройка параметров инструмента

Как обсуждалось в разделах «Создание локальных переменных» и «Добавление MDots к именам переменных» ранее в этой статье, некоторые инструменты Thor позволяют вам настраивать их, устанавливая параметры. Архитектура для указания параметров открыта, поэтому вы можете добавлять параметры к существующим инструментам или включать их в свои собственные инструменты.

Первый шаг в настройке параметров – добавить определение класса для каждого параметра в программу, реализующую инструмент. Это только технически определения классов; их цель – предоставить место для определения каждого параметра. Для каждого параметра необходимо указать четыре свойства, описанные в Таблице 1. В Листинге 8 показаны определения классов, которые настраивают параметры для инструмента Add MDots.

Таблица 1. Для каждого варианта инструмента необходимо указать значения этих свойств.

Свойство	Предназначение
Tool	Название инструмента, на который влияет опция.
Key	Строка, однозначно идентифицирующая опцию.
Value	Значение по умолчанию для этого параметра.
EditClassName	Имя класса и библиотеки классов, которые предоставляют пользовательский интерфейс для указания значения этого параметра, в форме: <класс> from <библиотеки классов>

Листинг 8. Эти три определения классов в конце кода для инструмента Add MDots указывают три параметра (показаны на рисунке 29) .

```
Define Class clsMDotsProperties As Custom

    Tool          = 'MDots'
    Key           = 'MDots Usage'
    Value         = 1
    EditClassName = 'clsEditMDots from Thor_Options_MDots.VCX'

Enddefine

Define Class clsMDotsWhereRequired As Custom

    Tool          = 'MDots'
    Key           = 'MDots where required'
    Value         = .F.
    EditClassName = 'clsEditMDots from Thor_Options_MDots.VCX'

Enddefine

Define Class clsMDotsinBeautifyX As Custom

    Tool          = 'MDots'
    Key           = 'MDots in BeautifyX'
    Value         = .F.
```

```
EditClassName = 'clsEditMDots from Thor_Options_MDots.VCX'
```

```
Enddefine
```

Следующий шаг в предоставлении опций – перечисление классов в части определения программы, которая реализует инструмент. Заполните свойство `OptionClasses` списком классов, разделенных запятыми, как в листинге 9.

Листинг 9. Эта строка в верхней (определяющей) части инструмента `Add MDots` указывает, что у инструмента есть три параметра, определяемые перечисленными классами.

```
.OptionClasses = 'clsMDotsProperties, clsMDotsWhereRequired, clsMDotsinBeautifyX'
```

Далее вам нужно создать класс пользовательского интерфейса для отображения опций. Это класс, указанный в `EditClassName` каждой опции. Как показывает пример, вы можете (и должны) использовать один класс для хранения пользовательского интерфейса для нескольких опций одного инструмента. Класс должен быть основан на базовом классе `Container`. На рисунке 47 показан класс `clsEditMDots`, используемый для инструмента `Add MDots`.

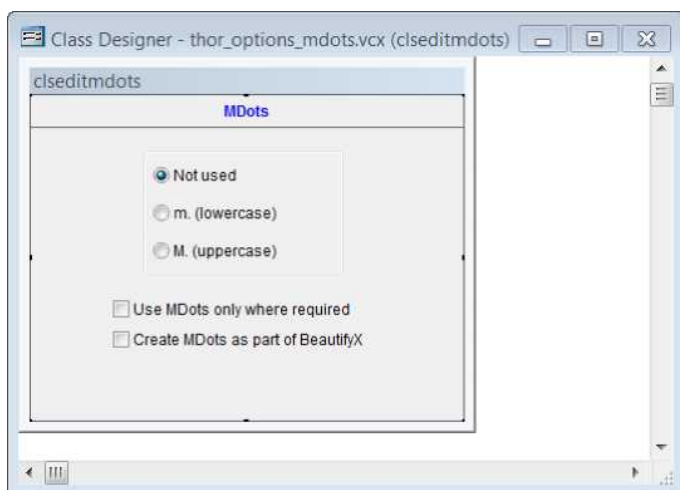


Рисунок 47. Этот класс, основанный на классе `Container`, содержит элементы управления для настройки параметров инструмента `Add MDots`.

Во время выполнения указанный класс добавляется к другому классу контейнера (и соответствующим образом изменяется в размерах). Этот класс контейнера имеет два метода, позволяющих вам сохранять и извлекать значения опций:

- `GetOption(Key, Tool)` извлекает текущее значение для указанной опции;
- `SetOption(Key, Tool, Value)` устанавливает значение для указанного параметра.

Вызывайте эти методы по мере необходимости для настройки страницы параметров и сохранения выбора пользователя. Например, метод `Refresh` первого флажка на рисунке 47 содержит код в листинге 10, в то время как метод `InteractiveChange` сохраняет выбор пользователя с помощью кода в листинге 11.

Листинг 10. Эта строка в методе `Refresh` флажка гарантирует, что элемент управления отражает текущую настройку параметра.

```
This.Value = This.Parent.Parent.GetOption('MDots, где требуется', 'MDots')
```

Листинг 11. Этот код в методе `InteractiveChange` флажка сохраняет выбор пользователя.

```
This.Parent.Parent.SetOption('MDots, где требуется', 'MDots', This.Value)
```

Последний шаг в предоставлении опций – использование опций в коде инструмента. Вы можете сделать это в процедуре `ToolCode` или в коде, который вызывает процедура. Вам нужно создать экземпляр движка Thor и вызвать его метод `GetOption` (с теми же параметрами, что показаны выше). Код в листинге 12 извлекает значение опции «MDots where required» и сохраняет его в переменной, чтобы его можно было использовать.

Листинг 12. Чтобы применить выбранную пользователем опцию, вы создаете экземпляр движка Thor и вызываете метод `GetOption`.

```
loThor      = Execscript (_Screen.cThorDispatcher, 'Thor Engine=')  
llMDotsWhereRequired = loThor.GetOption ('MDots, где требуется', 'MDots')
```

Дайте Тору шанс

Изменить свои рабочие привычки сложно, но когда новый инструмент предлагает реальные преимущества, это стоит усилий. На мой взгляд, Thor предлагает более чем достаточно преимуществ, чтобы сделать его достойным добавления в вашу среду разработки. Как только вы выясните, какие инструменты вы, скорее всего, будете использовать часто, вы можете назначить им горячие клавиши или поместить их во всплывающие меню, чтобы обеспечить легкий доступ.

Thor не только предоставляет вам несколько десятков удобных инструментов, но и позволяет вам взять все те небольшие инструменты, которые вы написали за эти годы, и сделать их легкодоступными.

Авторские права, 2012, Тамар Э. Гранор.

Скачать программу Thor можно здесь:

<https://github.com/VFPX/Thor>

Скачать дополнительные инструменты для Thor можно здесь:

<https://github.com/VFPX/ThorRepository>