silentroach карма −111,0 рейтинг 0,0
245 голосов

22 июня 2008 в 02:39

Конфиг-файлы в Delphi без проблем

Delphi*

Как-то было дело и я задумался над тем, как же удобнее всего настройки пользователя где-нибудь локально, быстренько это дело написать и забыть. Хранить это дело я решил в xml-файле. Куда уж без них.
Главное в этом способе то, что при добавлении каких-то новых параметров или изменении старых, не нужно будет переписывать код сохранения данных и их загрузки. Все будет делаться автоматически. Все, что нам нужно — это создать базовый класс, который будет за нас все делать, а сами данные мы будем хранить в объектах классов-наследников.

В общем, чтобы не пудрить мозг, сразу приведу код базового класса:

```
unit tlXMLClass;

interface

uses
  Classes, XMLIntf,
  // это важно!!! модуль позволяет работать со свойствами объекта
  TypInfo;

type
  TXMLClass = class(TPersistent)
  private
    // тут у нас будет имя файла с настройками
    FXMLFilePath: string;
    // а тут - название приложения для идентификации
    FApplicationName: string;
    // название корневой ветки файла
    FRootNodeName: string;
    // версия
    FVersion: byte;
  protected
    procedure SaveClass(oObject: TObject; Node: IXMLNode);
    procedure LoadClass(oObject: TObject; Node: IXMLNode);
  public
    constructor Create(const AppName, XMLFilePath: string; RootNodeName: string = 'config');

    procedure Initialize; abstract;

    // загрузка значений из файла
    procedure Load;
    // и их сохранение
    procedure Save;

    // виртуальный метод, его мы будем писать в наследнике
    procedure LoadDefaults; virtual;

    property ApplicationName: string read FApplicationName write FApplicationName;
    property RootNodeName: string read FRootNodeName;
    property Version: byte read FVersion write FVersion default 1;
  end;

implementation

uses
  // насчет XMLDoc и XMLIntf - эти два модуля появились в Delphi не так давно,
  // насколько я помню. если у вас их нет, то придется это дело реализовывать как-то по-другому.
```

```
  XMLDoc, SysUtils, Windows,
  resConfig;

{ TXMLConfig }

{$REGION 'Initialization'}
constructor TXMLClass.Create(const AppName, XMLFilePath: string; RootNodeName: string = 'config');
begin
  Initialize;

  FApplicationName := AppName;
  FXMLFilePath := XMLFilePath;
  FRootNodeName := RootNodeName;

  // задаем настройки по-умолчанию
  LoadDefaults;
end;

procedure TXMLClass.LoadDefaults;
begin

end;
{$ENDREGION}

{$REGION 'Loading'}
procedure TXMLClass.LoadClass(oObject: TObject; Node: IXMLNode);

  // тут мы пробуем найти свойство и задать его значение
  procedure GetProperty(PropInfo: PPropInfo);
  var
    sValue: string;
    TempNode: IXMLNode;
    LObject: TObject;
  begin
    // пробуем найти ветку с названием свойства
    TempNode := Node.ChildNodes.FindNode(PropInfo^.Name);
    // если не нашли, то выходим из функции. значение свойства останется значением по-умолчанию
    if TempNode = nil then
      exit;

    // если свойство не является объектом, то получаем значение из ветки
    if PropInfo^.PropType^.Kind <> tkClass then
      sValue := TempNode.Text;

    // анализируем тип свойства и задаем ему значение в соответствии с ним
    case PropInfo^.PropType^.Kind of
      tkEnumeration:
        if GetTypeData(PropInfo^.PropType^)^.BaseType^ = TypeInfo(Boolean)
          then SetPropValue(oObject, PropInfo, Boolean(StrToBool(sValue)))
          else SetPropValue(oObject, PropInfo, StrToInt(sValue));
      tkInteger, tkChar, tkWChar, tkSet:
        SetPropValue(oObject, PropInfo, StrToInt(sValue));
      tkFloat:
        SetPropValue(oObject, PropInfo, StrToFloat(sValue));
      tkString, tkLString, tkWString:
        SetPropValue(oObject, PropInfo, sValue);
      // а вот если свойство - объект, то рекурсивно выполняем процедуру
      // LoadClass, но уже для найденной ветки
      tkClass:
        begin
          LObject := GetObjectProp(oObject, PropInfo);
          if LObject <> nil then
            LoadClass(LObject, TempNode);
        end;
    end;
  end;
```

```
var
  i, iCount: integer;
  PropInfo: PPropInfo;
  PropList: PPropList;
begin
  // получаем количество публичных свойств объекта
  iCount := GetTypeData(oObject.ClassInfo)^.PropCount;

  if iCount > 0 then
  begin
    // запрашиваем кусочек памяти для хранения
    // списка свойств
    GetMem(PropList, iCount * SizeOf(Pointer));

    // и получаем их в PropList
    GetPropInfos(oObject.ClassInfo, PropList);
    try
      // пробегаемся по списку свойств
      for i := 0 to iCount - 1 do
      begin
        PropInfo := PropList^[i];
        if PropInfo = nil then
          break;

        // и для каждого свойства выполняем GetProperty (см.выше)
        GetProperty(PropInfo);
      end;
    finally
      // и в самом конце освобождаем занятую списком память
      FreeMem(PropList, iCount * SizeOf(Pointer));
    end;
  end;
end;

procedure TXMLClass.Load;
// процедура чтения из файла
var
  XMLRoot: IXMLNode;
  XML: IXMLDocument;
begin
  LoadDefaults;
  if not FileExists(FXMLFilePath) then
    exit;

  try
    // сам xml-Файл с настройками
    XML := LoadXMLDocument(FXMLFilePath);
    // корневая ветка xml-документа
    XMLRoot := XML.DocumentElement;

    // проверка на то, наш ли этот файл
    if (XMLRoot.NodeName <> FRootNodeName) or
       (XMLRoot.Attributes[rsApplication] <> FApplicationName) then
      exit;

    FVersion := XMLRoot.Attributes[rsFormat];

    // пошли загружать
    LoadClass(Self, XMLRoot);
  except
    // возникло исключение? загружаем значения по-умолчанию
    LoadDefaults;
  end;
end;
{$ENDREGION}

{$REGION 'Saving'}
```

```
procedure TXMLClass.SaveClass(oObject: TObject; Node: IXMLNode);
// здесь мы сохраняем значения и процедура эта очень
// сильно похожа на процедуру загрузки, поэтому комментировать
// я здесь буду только то, чего нет в той процедуре

  procedure WriteProperty(PropInfo: PPropInfo);
  var
    sValue: string;
    LObject: TObject;
    TempNode: IXMLNode;
  begin
    case PropInfo^.PropType^.Kind of
      tkEnumeration:
        if GetTypeData(PropInfo^.PropType^)^.BaseType^ = TypeInfo(Boolean)
          then sValue := BoolToStr(Boolean(GetOrdProp(oObject, PropInfo)), true)
          else sValue := IntToStr(GetOrdProp(oObject, PropInfo));
      tkInteger, tkChar, tkWChar, tkSet:
        sValue := IntToStr(GetOrdProp(oObject, PropInfo));
      tkFloat:
        sValue := FloatToStr(GetFloatProp(oObject, PropInfo));
      tkString, tkLString, tkWString:

        sValue := GetWideStrProp(oObject, PropInfo);
      tkClass:
        if Assigned(PropInfo^.GetProc) and Assigned(PropInfo^.SetProc) then
        begin
          LObject := GetObjectProp(oObject, PropInfo);
          if LObject <> nil then
          begin
            TempNode := Node.AddChild(PropInfo^.Name);

            SaveClass(LObject, TempNode);
          end;
        end;
    end;

    // тут мы создаем новую ветку в корне документа
    // и записываем в него значение свойства
    if PropInfo^.PropType^.Kind <> tkClass then
      with Node.AddChild(PropInfo^.Name) do
        Text := sValue;
  end;

var
  PropInfo: PPropInfo;
  PropList: PPropList;
  i, iCount: integer;
begin
  iCount := GetTypeData(oObject.ClassInfo)^.PropCount;

  if iCount > 0 then
  begin
    GetMem(PropList, iCount * SizeOf(Pointer));
    try
      GetPropInfos(oObject.ClassInfo, PropList);

      for i := 0 to iCount - 1 do
      begin
        PropInfo := PropList^[i];
        if PropInfo = nil then
          Break;

        WriteProperty(PropInfo);
      end;
    finally
      FreeMem(PropList, iCount * SizeOf(Pointer));
    end;
```

```
  end;
end;

procedure TXMLClass.Save;
var
  FRootNode: IXMLNode;
  FBackFileName: string;
  XML: IXMLDocument;
begin
  // куда уж без бекапа. на всякий случай не помешает
  FBackFileName := ChangeFileExt(FXMLFilePath, '.bak');
  try
    // оригинал удаляем
    if FileExists(FXMLFilePath) then
      DeleteFile(PChar(FXMLFilePath));

    try
      // создаем новый XML-документ
      XML := NewXMLDocument;

      // задаем ему кодировку и версию
      with XML do
      begin
        Encoding := 'UTF-8';
        Version := '1.0';
      end;

      // добавляем корневую ветку FRootNodeName
      FRootNode := XML.AddChild(FRootNodeName);
      FRootNode.Attributes[rsApplication] := FApplicationName;
      FRootNode.Attributes[rsFormat] := FVersion;

      SaveClass(Self, FRootNode);

      // сохраняем документ
      XML.SaveToFile(FXMLFilePath);
    except
      // а вот если произошла ошибка, то пытаемся
      // восстановить файл из созданной резервной копии
      if FileExists(FBackFileName) then
        RenameFile(FBackFileName, FXMLFilePath);
    end;
  finally
    // и в самом конце удаляем резервную копию
    if FileExists(FBackFileName) then
      DeleteFile(PChar(FBackFileName));
  end;
end;
{$ENDREGION}

end.
```

Вот такие вот дела. Код не шибко маленький, но, если разобраться, он совсем не сложный. Надеюсь еще и полезный. Для кого-нибудь :)
Да, код работает на D2007, но на версии раньше перевести его не будет проблем. На те версии, где есть поддержка XML.

Пример конфига, сгенерированного классом:

```
<?xml version="1.0" encoding="UTF-8" ?>
<config application="test" format="0">
        <Main>
                <HistoryDepth>40</HistoryDepth>
        </Main>
        <LookAndFeel>
```

```
                <WindowWidth>200</WindowWidth>
                <AlwaysOnTop>True</AlwaysOnTop>
                <AlphaBlending>False</AlphaBlending>
                <AlphaBlendValue>245</AlphaBlendValue>
                <AnimateWithAlpha>False</AnimateWithAlpha>
                <Elements>
                        <ItemDefault>
                                <Font>
                                        <Name>Tahoma</Name>
                                        <Size>8</Size>
                                        <Color>0</Color>
                                        <Bold>False</Bold>
                                        <Italic>False</Italic>
                                        <Strikeout>False</Strikeout>
                                        <Underline>False</Underline>
                                </Font>
                        </ItemDefault>
                        <ItemChecked>
                                <Font>
                                        <Name>Tahoma</Name>
                                        <Size>8</Size>
                                        <Color>9079434</Color>
                                        <Bold>False</Bold>
                                        <Italic>False</Italic>
                                        <Strikeout>True</Strikeout>
                                        <Underline>False</Underline>
                                </Font>
                        </ItemChecked>
                </Elements>
        </LookAndFeel>
        <Confirmation>
                <DeleteElement>True</DeleteElement>
        </Confirmation>
        <Windows>
                <HelpWindow>
                        <Top>182</Top>
                        <Left>73</Left>
                        <Width>1135</Width>
                        <Height>642</Height>
                        <WindowState>0</WindowState>
                        <SplitterLeft>156</SplitterLeft>
                </HelpWindow>
        </Windows>
</config>
```

P.S. все это дело поддерживает группировку свойств в отдельные объекты-наследники TPersistent.

страничка проекта на GoogleCode.

delphi, xml, настройки

Комментарии (93)

Ai_boy 22 июня 2008 в 03:01 # +2

Спасибо за статью :) А то я думал что я один такой "псих" на хабре, который верит что Delphi еще что-то может

silentroach 22 июня 2008 в 03:03 # ↑ 0

да нет, дельфийцев много. наверное, боятся выделиться, а то заминусуют к чертям =)

щас кто-нить прибежит, начнет рассказывать про то, что Delphi умерла давно и не следовало ей рождаться =)

Ai_boy 22 июня 2008 в 03:13 # ↑ 0

А ведь были времена когда C++ на нас смотрели свысока, а мы им показывали самую удобную среду разработки и огромное количество "реально" полезных программ. А сейчас.. даже немножко стыдно, за то что изучаю Delphi.

Ох Borland как они нас кинули...

silentroach 22 июня 2008 в 03:15 # ↑ +1

Ну, программы-то никуда не исчезли. чего только стоят такие монстры как TheBat, TotalCommander и FruityLoops. Сейчас Delphi - все та же удобная среда для разработки. Почему нет?

latrommi 22 июня 2008 в 17:56 # ↑ –3

Без паковщиков программы на делфи вскрываются с потрохами. Взять тот же DeDe, который даже формочки и кнопки покажет с кодом. Это нормально?
Делфи генерит мегаогромные файлы со всяким гамном, которое никогда нигде никто не использует. Это нормально?
И вообще синтаксис делфи взят из паскаля — студенческого языка программирования. Это нормально? :)

Какбэ, конечно, ну и фиг с ними, у каждого свой путь в жизни. Но называть делфи профессиональной средой — увольте.

silentroach 22 июня 2008 в 18:08 # ↑ 0

посмотрите ссылку в комментарии ниже

latrommi 22 июня 2008 в 18:28 # ↑ 0

Good Quality Applications Built With Delphi

сам факт существования таких списков наводит на мысли :)

silentroach 22 июня 2008 в 18:30 # ↑ 0

наводит на мысли, что некоторым людям без этого не верится.

aktuba 22 июня 2008 в 23:20 # ↑ +3

>>Без паковщиков программы на делфи вскрываются с потрохами. Взять тот же DeDe, который даже формочки и кнопки покажет с кодом. Это нормально?

Да, это нормально. И не просто нормально, а отлично, т.к. позволяет с легкостью, при умении, расширять программы.

>>Делфи генерит мегаогромные файлы со всяким гамном, которое никогда нигде никто не использует. Это нормально?

То, что ты что-то не используешь - не значит, что это "гамно". Я использую, например, и очень часто. Также поступают и мои знакомые. "Гамно" - это C#, который вообще без огромной кучи другого "гомна" работать не будет.

>>И вообще синтаксис делфи взят из паскаля — студенческого языка программирования. Это нормально? :)

ЭТО ОТЛИЧНО!!! В этом и есть один из плюсов! Во всяком случае, для меня. Будь синтаксис другим, скорее всего не остановился бы, после C++, на Delphi.

>>Но называть делфи профессиональной средой — увольте.
Delphi - это язык программирования. Профессиональная среда - BDS 2006-2007, которая намного профессиональнее многих других IDE.

Всегда улыбаюсь, когда читаю такие коменты - люди, не зная нихрена о предмете обсуждения, пытаются что-то доказывать. Для информации, на одном из наследников языка Паскаль, написана операционка полноценная (Оберон), на Delphi легко можно писать драйвера и системный софт, а в прикладном софте даже рядом, по удобству, скорости разработке и поддержке, никто не стоит.

latrommi 23 июня 2008 в 08:59 # ↑ +1

по поводу п. 1
Да, это нормально. И не просто нормально, а отлично, т.к. позволяет с легкостью, при умении, расширять программы.
Если бы человек хотел, чтобы его программу расширяли, он бы опубликовал ее открыто. Такая незащищенность — недостаток компилятора Делфи.

Я знаю несколько успешных компаний с населенностью от 100 человек, у которых ERP написана на VBA в

Excel. Да, оно работает, но это не нормально. Так же как и драйвера, написанные на Дельфи.

Я не менее часто улыбаюсь, когда люди не зная других людей делают о них выводы.

Успехов вам в ваших начинаниях!

aktuba 23 июня 2008 в 10:44 # ↑ 0

Мдя... Выводы мои подтвердились:

>>Если бы человек хотел, чтобы его программу расширяли, он бы опубликовал ее открыто. Такая незащищенность — недостаток компилятора Делфи.

Имелось в виду, что сам разработчик сможет, при помощи RTTI, легко расширять программу. При чем тут защищенность??? Для защиты программного кода используются другие методы, а не компиляторы ;).

>>Я знаю несколько успешных компаний с населенностью от 100 человек, у которых ERP написана на VBA в Excel. Да, оно работает, но это не нормально. Так же как и драйвера, написанные на Дельфи.

Бред. Ненормально - говорить такие утверждений. Пример - что "нормальнее", написать программу на пару недель и пару десятков лет пользоваться (VBA + Excel) или написать программу на пару десятков лет, потом содержать целый отдел, чтобы поддерживать эту "конструкцию" в рабочем состоянии (C++)? Драйвера на Delphi - это не "не нормально". Зависит от задач.

>>Я не менее часто улыбаюсь, когда люди не зная других людей делают о них выводы.

Ну мои выводы подтвердились - вы ничего не знаете о Delphi, а тем более о внутреннем устройстве компилятора или IDE BDS. Вам в голову вдолбили, что "Delphi это отстой", вы и рады на каждом углу так орать, а в действительности понятия не имеете об огромных плюсам Delphi. Пора расти, а то так и будет "в каждой дырке затычкой"...

latrommi 23 июня 2008 в 11:10 # ↑ 0

:)
У меня сейчас нет жаления объяснять дальтоникам цвета радуги.
Давайте на этом закончим.

aktuba 23 июня 2008 в 11:15 # ↑ +1

=) Доказывать и не надо ничего - я более 10 лет использую Delphi и считаю что это лучший язык программирования, а C++ и PHP - это худшие, хотя и их изучал и применял.

Но, похоже, вы не излечимы.

silentroach 22 июня 2008 в 03:20 # ↑ +1

http://delphi.wikia.com/wiki/Good_Qualit…
разве после посещения этой страницы за Delphi может быть стыдно?

Ai_boy 22 июня 2008 в 03:35 # ↑ 0

Да мне за среду разработки стыдно! (Я до сих пор стараюсь использовать Delphi 7 где только можно)

Я всегда восхищался дизайном среды, удобно, практично, супер... А теперь.. тебе BDS 2005-2007 ничего не напоминает? ОНИ ВСЕ СЛИЗАЛИ С Visual Studio? Обьясни мне, зачем отказываться от своих "гениальных" разработок и следовать моде?

silentroach 22 июня 2008 в 03:52 # ↑ 0

потому что это действительно удобно.
непривычно, но это дело времени. я уже привык к 2007.
хотя на работе пишу на D3.

Ai_boy 22 июня 2008 в 04:10 # ↑ 0

Мне пока тяжело! Посмотрим может и привыкну.

Радует одно - конкуренция с VB.NET заставила их шевелиться, и сам язык стал действительно лутше.

Bobos 22 июня 2008 в 04:11 # ↑ 0

Язык не изменился с первой версии

Ai_boy 22 июня 2008 в 04:17 # ↑ 0

Прости, Мозги под утро не варят :) Были добавленны компоненты (которых я кстати очень долго ждал)

silentroach 22 июня 2008 в 04:24 # ↑ 0

не менялся, но обрастал новыми возможностями

corristo 22 июня 2008 в 15:32 # ↑ +1

одна перегрузка операторов структур и классов (классов только в .net-проектах, Вроде) чего стоит в изменениях язка

corristo 22 июня 2008 в 15:39 # ↑ 0

удивлен. что нету замечательных программ PicturesToExe, PixBuilder Studio и WinNavigator (он правда стар уже) замечательной компании wnsoft - www.wnsoft.com

Mercury13 22 июня 2008 в 09:54 # 0

А я, не найдя хорошего XML-компонента, как-то написал свой простенький XML-подобный язык Multilevel (что-то вроде многоуровневого ini-файла).

Кстати: спасибо за подсказку, как работать с published.

zupernintendo 22 июня 2008 в 11:49 # +2

В Lazarus данная фича делается парой кликов мыши.

aktuba 22 июня 2008 в 13:31 # 0

XMLDoc и XMLIntf, насколько я помню, есть с D7. А по поводу Lazarus - в Delphi тоже можно сделать в пару кликов, только будет менее универсально, как и в Lazarus. Lazarus-у еще ОЧЕНЬ далеко до Delphi 7, не говоря уже о BDS2006-2007.

silentroach 22 июня 2008 в 14:41 # ↑ 0

ну рассказали бы хоть. я то я зря этот класс написал, получается?

aktuba 22 июня 2008 в 14:45 # ↑ +1

Поищи по слову "Binding" ;)

silentroach 22 июня 2008 в 14:56 # ↑ 0

XML Data Binding ?
Насколько я понял, это немножко другое. У меня это дело попроще.

aktuba 22 июня 2008 в 15:02 # ↑ 0

Да, про XML Data Binding. Предназначено как-раз для того, чтобы сделать класс по образцу. И используется соответственно, в пару кликов.

Кстати, я не использую ни его, ни твой подход (который я уже где-то видел). Я пишу свой класс для каждой программы.

silentroach 22 июня 2008 в 15:05 # ↑ 0

а если нужно добавить какое-то свойство, убрать или изменить?
в случае с XML Data Binding это займет дольше времени, нежели с использованием моего класса.

Я тоже использую свой класс для каждой программы. Класс-наследник того, что я написал =)

aktuba 22 июня 2008 в 15:09 # ↑ 0

О чем я и написал выше: "в Delphi тоже можно сделать в пару кликов, только будет менее универсально".

Maxter 22 июня 2008 в 14:40 # +1

Переходите уже на другую платформу. На .NET например. Там это 10 строк занимает, а у вас 100, это упадничество какое-то.
P.S. Я не в коем случае не хочу сказать, что *Дельфи уже умерла, и не следовало ей рождаться*, т.к. мне очень не хочется с кем то спорить в это чудесное утро.

silentroach 22 июня 2008 в 14:43 # ↑ –3

хотел в свое время, но как-то медленновато на нем все работает.

diamant 22 июня 2008 в 14:51 # ↑ –3

.NET — прямой наследник Delphi

aktuba 22 июня 2008 в 14:55 # +2

Ну это ты зря...
Maxter - покажи тоже самое на .Net в 10 строк ;)

Aledensoft 22 июня 2008 в 15:21 # 0

эт точно, и ведущий разработчик-идеолог Delphi RTTI и .NET один и тот же кстати.

silentroach 22 июня 2008 в 14:57 # 0

много строк тут занимает только класс-родитель, который делает за нас всю работу. с ним не надо возиться и переписывать. а вот описание класса-наследника, действительно с нашим конфигов - получается очень мал. и менять мы его можем так, как нам захочется - ничего не сломается и все будет сохраняться/загружаться нормально.

Ai_boy 23 июня 2008 в 05:07 # 0

Я по долгу службы изучаю .NET и скажу, что 10 строк там это все занимает только потому, что Микрософт за вас написала ну просто огромную кучу кода. 4000 классов o_O! И это только mscorelib! Сравните со 250 (приблизительно) в Delphi , и оцените насколько все грамотнее и лаконичнее (причем эти 250 включают в себя все компоненты). Ведь программы на Delphi пишутся с таким же успехом как и на .NET А то что структура int (обычное 32 битное число) имеет около 22 методов, вас не пугает o_O.

PS: Изучать надо язык на котором пишите.

mr_smile 22 июня 2008 в 15:17 # 0

дельфи прекрасно работает с ini-файлами
или сабж немого о другом? :)

silentroach 22 июня 2008 в 15:40 # 0

можно прочитать и проверить.
о другом, конечно.

mr_smile 22 июня 2008 в 15:52 # –1

т.е. не о конфиг-файлах?

silentroach 22 июня 2008 в 16:11 # 0

об **удобных** и красивых конфиг-файлах. и о способе их автоматической загрузки и сохранения.

mr_smile 22 июня 2008 в 16:20 # 0

ну не знаю. на вкус и цвет как говорится...
по мне ini-файлы намного красивее xml-ек и удобнее в разборе как руками так и абсолютно любыми существующими инструментами.

повторюсь: я говорю о конфигах.
ИМХО удобный конфиг - это тот, который понятен и который легко правится.

aktuba 22 июня 2008 в 17:38 # +1

В .ini многие вещи невозможно реализовать, в то время как в xml очень легко.
Да и править xml-конфиги намного легче и удобнее, если правильно реализовано.

leave 22 июня 2008 в 16:26 # 0

вы видели конфиги JBoss ? конфиги на XML - величайшее зло (ИМХО).

silentroach 22 июня 2008 в 16:28 # 0

нет, не видел. как и Вы не видели результатов отработки объекта моего класса.

leave 22 июня 2008 в 16:36 # 0

ну залейте на pastebin :)
или лучше сначала попробуйте создать 2-3кб конфиг "по-вашему", а потом в консоли в vi(m) позанимайтесь его редактированием. если останетесь при мнении, что это удобно - показывайте публике.

silentroach 22 июня 2008 в 16:49 # 0

не люблю vim и линукса под рукой нет.
конфиг-файлы - не для редактирования вручную, я так считаю.

mr_smile 22 июня 2008 в 17:13 # ↑ 0

поэтому Вы и используете xml для конфигов :)

silentroach 22 июня 2008 в 18:25 # ↑ 0

именно

leave 22 июня 2008 в 17:29 # ↑ 0

тогда понятно :)
что ж, успехов в девелопменте. и не поддавайтесь на провокации сишников:)

aktuba 22 июня 2008 в 17:45 # ↑ 0

Ты файлы .doc тоже в vi(m) редактируешь??? Нет? Тогда зачем там .xml редактировать? Это раз. Два - удобно, это когда не надо в файл настроек лезть руками. ;)

silentroach 22 июня 2008 в 18:27 # ↑ 0

добавил в конец топика пример полученного таким образом конфиг-файла.

dime 23 июня 2008 в 17:54 # ↑ 0

Гмм.. А в чём удобство xml-конфигов???
Для правки вручную - ужос-ужос.
Для парсинга... Довольно тяжёлая артилерия и так привлекается, а тут ещё и сверху такая портянка кода...
Не понимаю в чём смысл использования инструмента (xml) не по назначению.

silentroach 23 июня 2008 в 18:06 # ↑ 0

удобство примера в статье в том, что программисту, в случае изменения/удаления/добавления каких-либо свойств в настройки, не нужно переписывать/дописывать код, который эти настройки будет читать и сохранять.

Вы делаете такие программы, конфиги которых приходится редактировать вручную? - "ужос-ужос".
Расскажите нам, пожалуйста, в чем предназначение xml ?

dime 23 июня 2008 в 18:18 # ↑ 0

1. Я не про пример, я про то, зачем хранить конфиги в xml? в чём удобство xml-конфигов?
2. Вы делаете такие программы, настройки которой кроме как с использованием этой же самой программы никак не изменить? Ужос-ужос. Ещё, поди, час менюшками для этого хлопать приходится? Ну-ну. Хотя нет :). Раз таки xml, значит ещё не всё потеряно :).
3. xml? в основном для обмена данными. Часто кроссплатформенного. Но конфигами кроме вашей программы никто другой не будет пользоваться. Зачем там xml?

silentroach 23 июня 2008 в 18:49 # ↑ 0

каждый выбирает то, что ему больше не нравится. я не призываю пользоваться этим методом вопреки своим желаниям.

wdk 22 июня 2008 в 15:40 # ↑ –1

XML гламурнее.

silentroach 22 июня 2008 в 17:13 # ↑ 0

точно =)

hobbeat 25 июня 2008 в 13:27 # ↑ 0

Гламур для дур, XML это сурово.

Kolan 22 июня 2008 в 16:16 # 0

Не говоря о том, что сам по себе код довольно хреновый и дилетантский, он еще и некому не нужный. Вместо сабжа надо использовать стандартные механизмы сериализации объектов и не позориться.

ЗЫ

```
procedure TXMLClass.Initialize;
begin

end;
```

Нафиг пустой метод? Есть деректива abstract.

Префиксы а-ля венгерская нотация (oObject и iCount) не принято использовать в Delphi.

Если хочешь еще и много, то запость код на www.delphimaster.ru :)

silentroach 22 июня 2008 в 16:24 # ↑ 0

о сериализации мы уже говорили в комментариях выше.
initialize - для использования в наследниках.
с префиксами привык, извините.

спасибо за отзыв.

nickkee 26 июня 2008 в 06:15 # ↑ 0

директива "abstract" специально создана для этих целей, чтобы не городить пустых методов.

aktuba 22 июня 2008 в 17:55 # ↑ +1

Кто сказал "надо использовать стандартные механизмы сериализации объектов"? Правильнее - "надо использовать то, что удобно и практично в данный момент". Если делать программу с десятком-другим настроек - да, можно и .ini использовать, и сериализацию. А если настроек 2-3к., то тут сериализация уже плохо, а .ini вообще не реально использовать. Зато .xml как-раз то, что надо - удобно, практично.

А вообще, я сторонник того, чтобы пользователь не мог добраться ручками до настроек программы.

silentroach 22 июня 2008 в 18:28 # ↑ 0

спасибо за поддержку =)

Kolan 22 июня 2008 в 16:22 # 0

Ссылки почему-то не вставились :(.

Стандарт стилевого оформления исходного кода DELPHI
http://www.delphikingdom.com/asp/viewitem.asp?catalogid=802

Сериализация объектов стандартными средствами Delphi
http://rsdn.ru/article/delphi/serialization.xml

silentroach 22 июня 2008 в 16:33 # ↑ 0

о существовании обеих статей я осведомлен.
даже читал ;)

Kolan 22 июня 2008 в 16:39 # 0

>о сериализации мы уже говорили в комментариях выше.
Где? Я говорю о стандартной сериализации, которая используется самой средой и является естественной для Delphi.

>initialize - для использования в наследниках.
Я понимаю, так почему бы не сделать его абстрактным, коли реализации нет, а нужен он лишь для перекрытия в потомках?

А мое имхо, настройки проще хранить в ини, и не сериализовывать целые объекты. А на форум ты таки закинь исходник :)

silentroach 22 июня 2008 в 17:13 # ↑ 0

Я не люблю форумы и не хочу никому ничего доказывать. если Вы считаете, что я не прав, используя такой код для хранения настроек, то я это уже понял, но не вижу причин перестать им пользоваться - он мне кажется удобным. если у Вас есть пример, как сделать то же самое, только лучше - почему бы его не написать, я им с большим удовольствием воспользуюсь. глядишь, и чему новому научусь - я знаю о том, что я не идеален и мой код тоже таким может и не быть и с удовольствием займусь его изменением в лучшую сторону.

corristo 22 июня 2008 в 17:39 # ↑ 0

ну насчет того чтобы сделать пустой метод абстрактным товарищ абсолютно прав, к чему такая болезненная реакция?

silentroach 22 июня 2008 в 18:09 # ↑ 0

я уже давно в исходном коде поменял. реакция не болезненная, я разве тут чем-то возмущаюсь? =)
я критике только рад, честно.

Kolan 22 июня 2008 в 19:49 # 0

Не, я на форум, только именно на этот, советую как раз для того, чтобы "чему новому научиться".

aktuba 22 июня 2008 в 21:06 # ↑ +1

Тогда уж лучше на винград ;)

dborovikov 22 июня 2008 в 21:12 # 0

1) Конфиги лучше хранить в YAML формате
2) Конфиги зло, хотя в некоторых случаях действительно без них никак
3) Гибкие идеи не стоит искать среди интерпрайз решений

Это мое мнение, интересно будет послушать ваши возражения:)

silentroach 22 июня 2008 в 21:34 # ↑ 0

1. чем лучше?
2. согласен.
3. не понял.

dborovikov 22 июня 2008 в 22:36 # ↑ 0

1. YAML как раз создавался для таких целей. В общем здесь все написано - http://ru.wikipedia.org/wiki/YAML
2. -
3. Использование XML-х конфигов свойственно для java приложений, которые в большинстве своем громоздки и не блещут свежими идеями.

silentroach 23 июня 2008 в 01:52 # ↑ 0

1. для yaml нужно библиотеки какие-нить подключать сторонние. зачем, когда XML тоже с этим справляется. и уж тем более нативные майкрософтные библиотеки.
3. это же не значит то, что XML плох.

в чем плюс YAML перед XML ?

dborovikov 23 июня 2008 в 02:35 # ↑ 0

1. Для YAML уже написано куча библиотек под разные платформы. Если для вас скачать архив, распаковать и подключить компонент к проекту это препятствие перед тем что бы использовать лучшие решения.. чтож, пользуйтесь стандартными компонентами и бодрите себя мыслью, что они самые-самые....
3. А кто сказал что XML - плох? Плохим является решение использовать его для конфигурации каждого чиха в интрепрайз приложениях.

Про сравнение YAML и XML. Сравнивать их в общем смысле не корректно, ибо
"YAML — человекочитаемый формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования."

Т.е. XML язык разметки, а YAML нет, поэтому сравнивать их корректно только в нашем примере.

Ознакомиться с преимуществами YAML над XML в написании конфигурационных файлов можно на вики.

Ai_boy 23 июня 2008 в 05:48 # ↑ 0

1) "скачать архив, распаковать и подключить компонент к проекту это препятствие перед тем что бы использовать лучшие решения.. чтож, пользуйтесь стандартными компонентами и бодрите себя мыслью, что они самые-самые...." - Позвольте спросить, у вас много опыта в создании програмного обеспечения? (Своего, не для фирмы) Подключение любой сторонней библиотеки ведет к уйме проблем - изучение документации (если она есть), отладка и переотладка кода ибо сторонние библиотеки 99% имеют глюки которые приходиться учитывать и избегать. И все это ведет только к одному - трате драгоценного :) программистского времени. Намного проще и надежнее! написать код на стандартных компонентах. Этого же мнения придерживаются почти все ведущие специалисты.

3) Микрософт, Borland использует ХМЛ везде где только можно, и по вашему получается что они глупее вас? Зачем они выбирают "плохое" решение? Не смешите меня.

YAML и XML вообще не надо сравнивать ибо XML - универсален. Изучив его вы сможете применять его во множестве областей программирования, и вас всегда и везде поймут.

dborovikov 23 июня 2008 в 12:45 # ↑ 0

1. Я пользуюсь свободными инструментами, а там любая более менее внятная библиотека является сторонней. А что касается изучения, то никогда не испытывал проблем с этим, проблем с отладкой и изучением никогда не испытывал, я не знаю откуда вы вообще это придумали. Мои коллеги и начальство тоже так считают

3. >Зачем они выбирают "плохое" решение? Не смешите меня
Это вы не смешите, интрепрайз на то он и интерпрайз - все должно быть стандартно и понятно любому идиоту, они никогда не ставили цели использовать что-то более менее современное. Их решения хороши с маркетинговой точки зрение, с технической они просто отвратительны.

"Изучив его вы сможете применять его во множестве областей программирования"

Ну я же не говорил, что "XML - не нужен", я его постоянно использую, однако хранить в нем конфиги это сверх избыточное решение, YAML штука специализированная и в общем-то примитивная по сравнению с XML, более того интуитивно понятная.

aktuba 23 июня 2008 в 11:03 # ↑ 0

Да, стандартные решения тем и хороши, что они стандартные:
1. Если человеку разрешено копаться в файле настроек (что я считаю плохим признаком), то xml более удобен, т.к. его знают больше. При использовании YAML - его еще очень многим надо изучить, это не считая того, что очень многое в YAML зависит от непонятных (для новичка) пробелов и черточек. Короче - XML лучше, в данном случае.
2. Если файл настроек закрыт для ручного редактирования, то и разницы нет, для человека. А для программы XML проще - не надо использовать кучу лишнего хлама и чужих багов.

>>Если для вас скачать архив, распаковать и подключить компонент к проекту это препятствие перед тем что бы использовать лучшие решения..

Далеко не лучшие! К тому же, как уже написали ниже - скачать и установить не проблема, а вот изучение = потраченное время, часто (как в данном случае) впустую.

dragonhome 24 июня 2008 в 15:51 # ↑ 0

а у YAML есть сейчас порт для delphi?

dborovikov 24 июня 2008 в 18:03 # ↑ 0

Именно как delphi-модуля нету. Конечно есть реализации для .NET и еще куча разных реализаций:
- YAML C Libraries:
- libyaml # New Fast YAML (1.1)
- Syck # Old Fast YAML (1.0)
- PyYaml # YAML 1.1 Implementation
- JsYaml # JavaScript PyYaml port
- RbYaml # Ruby port of PyYaml
- JvYaml # Java port of PyYaml
- ocaml-syck # Syck bindings for OCaml
- Perl Modules:
- YAML # Pure Perl YAML Module
- YAML::XS # Binding to libyaml
- YAML::Syck # Binding to libsyck
- YAML::Tiny # A small YAML subset module
- PlYaml # Perl port of PyYaml
- YamlReference # Haskell 1.2 reference parser
- YPaste # Play with the Haskell 1.2 parser

silentroach 24 июня 2008 в 18:08 # ↑ 0

отлично, именно поэтому нам и следует использовать его в delphi.

fat_hamster 23 июня 2008 в 11:13 # 0

я не читал коменты, не думаю что там в холи-варе что-то полезное есть, но вдруг howto именно там конечно

но идею не очень понял, может я туповать, а может автору незачет

точнее вообще не понял: имелся в виду подход, что класс будет сохранять свойства обьектов, которые ему подсовывают? или же мне надо дописывать в классе-наследнике каждое свойство?

надо было рядом с базовым классом положить и пример наследника что-ли

з.ы. а делфи рулит

silentroach 23 июня 2008 в 11:46 # ↑ 0

все **published** свойства объекта класса-наследника от класса, приведенного мной в статье будут им сохраняться и читаться.

fat_hamster 23 июня 2008 в 13:36 # ↑ 0

ну если так до допустим

тогда в теории вроде сложные типы тоже можно сохранять? к примеру я хочу помнить настройки шрифта для неважно-чего; я должен в published добавить свойство MyFont, и перед сохранением настроек просто присвоить ему указатель на шрифт, настройки которого я хочу помнить? вроде в SaveClass попадаем, что свойство == класс определяем, но дальше

```
tkClass:
if Assigned(PropInfo^.GetProc) and Assigned(PropInfo^.SetProc) then
begin
LObject := GetObjectProp(oObject, PropInfo);
```

if LObject nil then

никогда не срабатывает

это бага или фича?

silentroach 23 июня 2008 в 14:26 # ↑ 0

под настройки шрифта создайте отдельный класс на базе TPersistent.
TFontProperty = class(TPersistent)
и его опишите точно так же - **published** свойства этого объекта попадут внутрь.
а в своем классе определите property Font: TFontProperty.
объект, на который будет ссылаться Font, нужно будет создать и уничтожить.

silentroach 23 июня 2008 в 14:30 # ↑ 0

небольшой пример - http://codepaste.ru/1068/

r3code 2 июля 2009 в 15:52 # 0

Решил опробовать модуль и получил ругань от 7 Дельфи
SetPropValue(Self, PropInfo^.Name, StrToInt(sValue));

r3code 2 июля 2009 в 15:54 # 0

Почему то пост сам отправился без моего ведома. Пишу снова.
На строки

```
SetPropValue(Self, PropInfo, StrToInt(sValue));
```

Шла ругань

```
[Error] tlXMLClass.pas(79): Incompatible types: 'String' and 'PPropInfo'
```

После исправления на:

r3code 2 июля 2009 в 15:55 # 0

```
SetPropValue(Self, PropInfo^.Name, StrToInt(sValue))
```

Все стало нормально.

Вы уверены, что в приведенном коде статьи ошибок нет?

r3code 2 июля 2009 в 18:25 # 0

silentroach
Все 3 моих коментария относяться к коду изложенному в вашем блоге, как оказалось он не соответсвует исходикам в статье.

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.