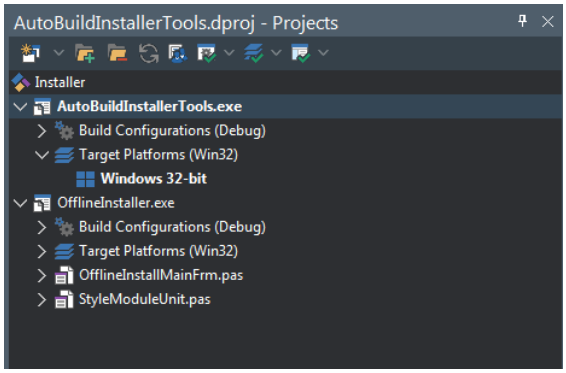
zInstaller 使用指南

开源地址 https://github.com/PassByYou888/zInstaller

zInstaller 是从 zAI 的在线安装器提炼而来的开源安装程序，它使用 Delphi-FMX 框架运行分发包。我们可以在此基础上，深度定制化自己的安装程序系统：有没有觉得 Vs2019 安装程序做的很炫，zIntaller 也可以做到！

使用自动化打包程序构建 zInstaller

使用 Delphi-XE 打开 Installer.groupproj 组以后会看见两个工程

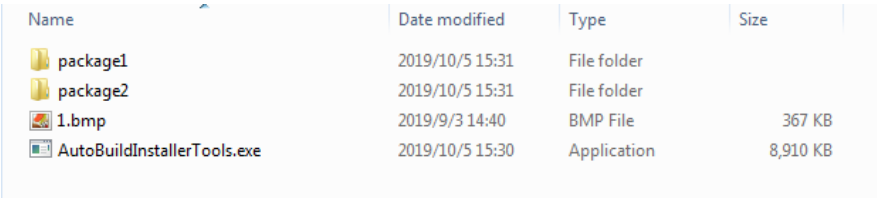


AutoBuildInstallerTools

这是一个自动化打包分发程序，如果我们要安装的程序包非常多，比如几十数百个，使用它可以非常快捷的生成安装程序包。这是基于 console 的程序，使用它时，将它 copy 到你需要打包的根目录，运行它即可完成打包

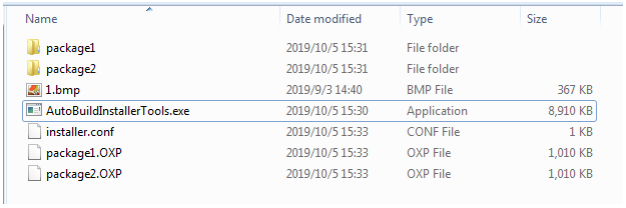
以下图为例

运行 AutoBuildInstallerTools.exe 以后，package1 和 package2 会生成两个独立程序包，而 1.bmp 并不会被打包，简单来说，使用 AutoBuildInstallerTools.exe，你必须创建一个子目录用于说明这是一个独立包



Name	Date modified	Type	Size
package1	2019/10/5 15:31	File folder	
package2	2019/10/5 15:31	File folder	
1.bmp	2019/9/3 14:40	BMP File	367 KB
AutoBuildInstallerTools.exe	2019/10/5 15:30	Application	8,910 KB

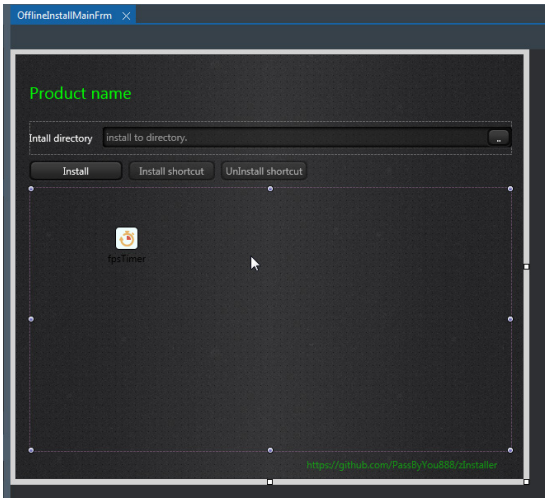
当我们运行 AutoBuildInstallerTools.exe 以后，生成完成，多了 3 个文件



Name	Date modified	Type	Size
package1	2019/10/5 15:31	File folder	
package2	2019/10/5 15:31	File folder	
1.bmp	2019/9/3 14:40	BMP File	367 KB
AutoBuildInstallerTools.exe	2019/10/5 15:30	Application	8,910 KB
installer.conf	2019/10/5 15:33	CONF File	1 KB
package1.OXP	2019/10/5 15:33	OXP File	1,010 KB
package2.OXP	2019/10/5 15:33	OXP File	1,010 KB

OfflineInstaller

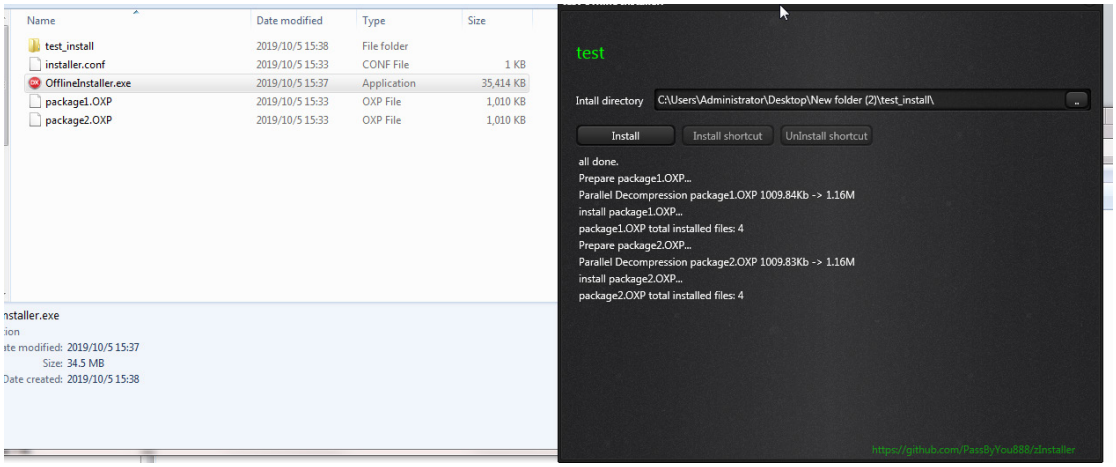
这是 zInstaller 的离线安装器，当我们从 github 下载完成 zInstaller 以后，可以直接编译它



在这时，我们可以到源码中找到 ProductID 和 ProductName 两个常量，现在我们可以修改它，让它符合我们的需要

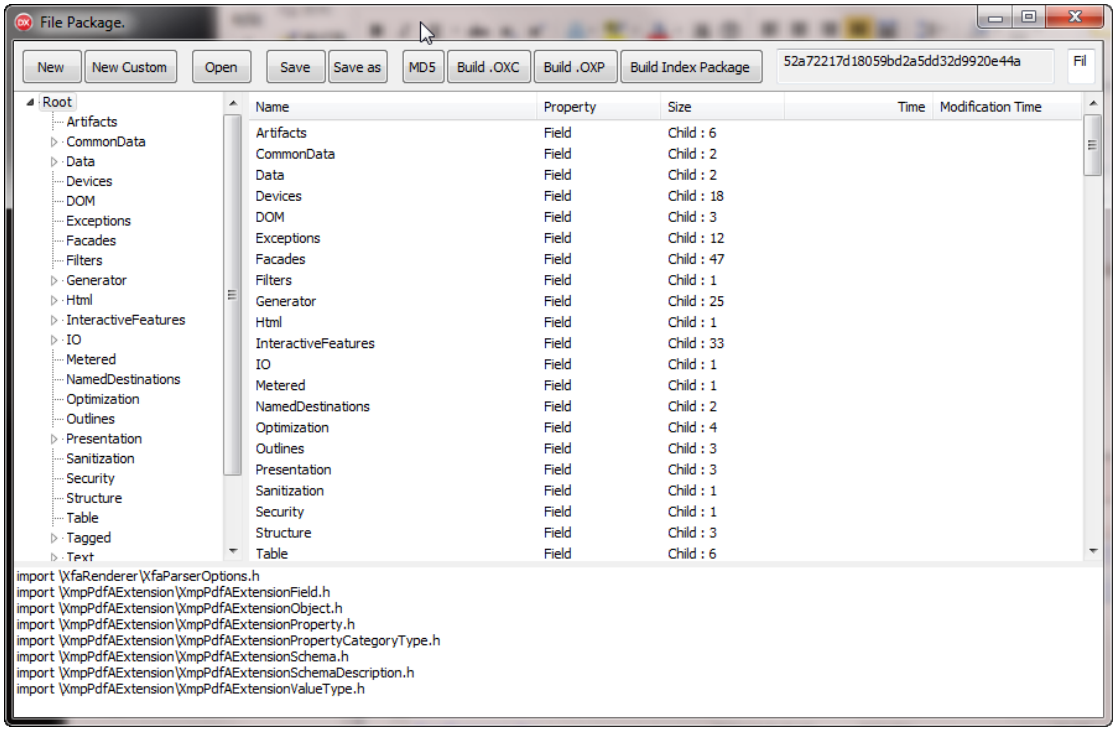
```
var
  OfflineInstallMainForm: TOfflineInstallMainForm;

const
  ProductID = 'MyProduct';      // 产品ID，这也是默认安装目录名
  ProductName = 'Product Name'; // 产品名称

  // 展开安装程序包到硬盘的目标目录
function OpenInstallPackage(fileName: U_String): TObjectDataManager;
procedure ExtractInstallerPackage(packageFile, SavePath: U_String);
// 构建windows的桌面快捷方式，在安装结束时使用
procedure BuildShellLinkToDesktop(const fileName, workDirectory, shortCutName: WideString);
// 构建windows的程序文件夹快捷方式，在安装结束时使用
procedure BuildShellLinkToProgram(const fileName, workDirectory, shortCutName: WideString);
// 构建windows的开机自动启动文件夹快捷方式，在安装结束时使用
procedure BuildShellLinkToStartup(const fileName, workDirectory, param, shortCutName: WideString);
// shell运行程序，在安装中使用
procedure ShellRun(ExeFile, param: U_String);
```

然后编译它即可，完成后，将刚才我们打包的那些文件复制到 OfflineInstaller.exe 的目录，运行它，即完成分发

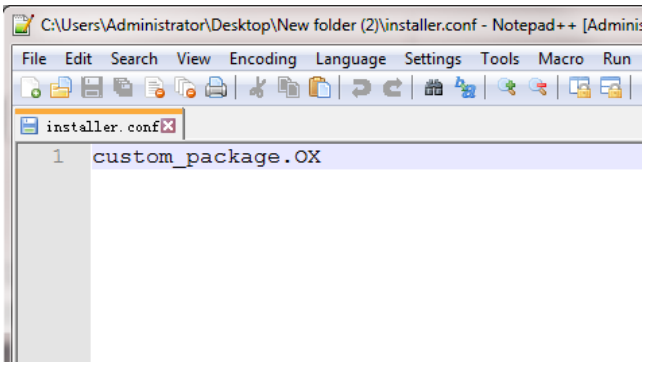


使用 FilePackageWithZDB 构建 Installer

我们在 zInstaller 的附属工具可以找到 FilePackageWithZDB
使用该工具，可以制作.oxp,.oxc,.ox，这些都是 zInstaller 可以支持的文件包格式
我们在 FilePackageWithZDB 构建的目录结构，均能还原到硬盘中
由于操作系统的文件名是非定长的，FilePackageWithZDB 要使用 New Custom 方式构建数据库，字符串长度要给 255



待编辑完成后，保存成 custom_package.OX
然后编辑 Installer.conf
在里面指定要安装的程序包



创建 windows 快捷方式

当我们打开 OfflineInstaller 源码文件以后，我们将会看到有详细中文注释的快捷方式函数

```
var
  OfflineInstallMainForm: TOfflineInstallMainForm;

const
  ProductID = 'test_install';      // 产品ID，这也是默认安装目录名
  ProductName = 'test'; // 产品名称

  // 展开安装程序包到硬盘的目标目录
function OpenInstallPackage(fileName: U_String): TObjectDataManager;
procedure ExtractInstallerPackage(packageFile, SavePath: U_String);
// 构建windows的桌面快捷方式，在安装结束时使用
procedure BuildShellLinkToDesktop(const fileName, workDirectory, shortCutName: WideString);
// 构建windows的程序文件夹快捷方式，在安装结束时使用
procedure BuildShellLinkToProgram(const fileName, workDirectory, shortCutName: WideString);
// 构建windows的开机自动启动文件夹快捷方式，在安装结束时使用
procedure BuildShellLinkToStartup(const fileName, workDirectory, param, shortCutName: WideString);
// shell运行程序，在安装中使用
procedure ShellRun(ExeFile, param: U_String);

implementation
```

当我们将程序往下翻，能看到创建快捷方式的位置指引，一目了然

```
procedure TOfflineInstallMainForm.InstallButtonClick(Sender: TObject);
var
  i: Integer;
  n: U_String;
  li: TListBoxItem;
  FileList: TPascalStringList;
begin
  DisableAll;
  FileList := TPascalStringList.Create;
  FileList.LoadFromFile(umlCombineFileName(TPath.GetLibraryPath, 'installer.conf'));
  if (FileList <> nil) and (FileList.Count > 0) then
    begin
      // 检查安装包的正确性
      for i := 0 to FileList.Count - 1 do
        begin
          if not umlFileExists(umlCombineFileName(TPath.GetLibraryPath, FileList[i])) then
            begin
              DoStatus('Loss of Installation Pack: %s', [FileList[i].Text]);
              DoStatus('');
              disposeObject(FileList);
              EnabledAll;
              Exit;
            end;
        end;

      umlCreateDirectory(InstallPathEdit.Text);

      // 开始在后台线程中运行安装程序
      TComputeThread.RunM(nil, FileList, DoInstallRun);

      DoStatus('all done.');
    end;
end;

procedure TOfflineInstallMainForm.InstallIconButtonClick(Sender: TObject);
begin
  // 在这里实现windows快捷方式构建
  // 使用下列API即可
  (*
    procedure BuildShellLinkToDesktop(const fileName, workDirectory, shortCutName: WideString);
    procedure BuildShellLinkToProgram(const fileName, workDirectory, shortCutName: WideString);
    procedure BuildShellLinkToStartup(const fileName, workDirectory, param, shortCutName: WideString);
  *)
end;

procedure TOfflineInstallMainForm.UnInstallIconButtonClick(Sender: TObject);
begin
  // 这里将windows快捷方式文件删除
  // 直接使用API umlDeleteFile(快捷方式文件名)
end;
```

By,qq600585

2019-10