Systémové programovanie Windows

Windows Installer XML

Andrea Číková
Martin Osovský

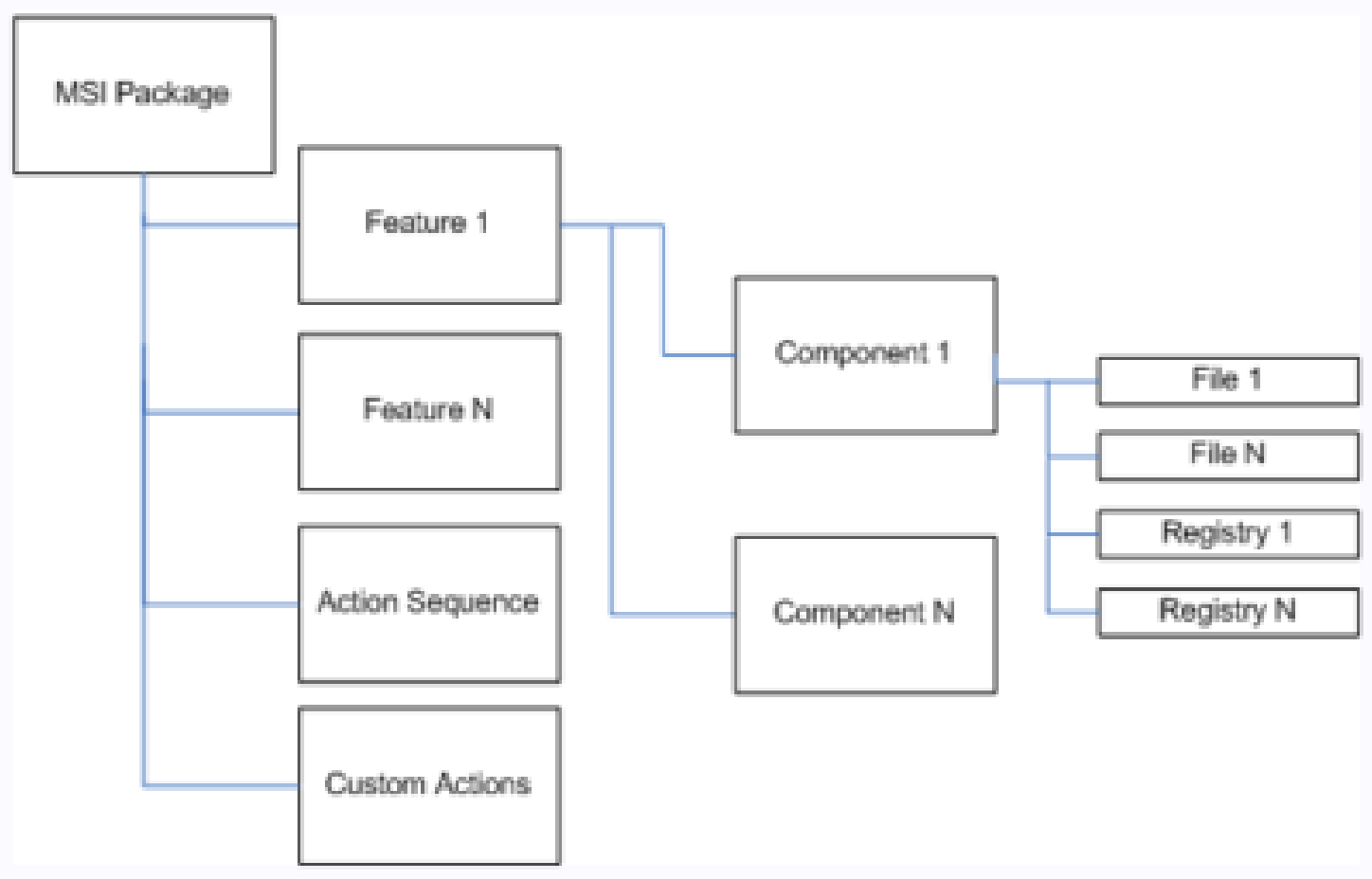
O čom to dnes bude?

- Windows Installer
- Windows Installer XML
- Dôležité elementy

Windows Installer

- Pôvodne Microsoft Installer (MSI)
- Štandard pre inštalovanie desktopových aplikácií na OS Windows
- Inštalácia sa deje pomocou tzv. inštalačných balíčkov (MSI balíčky)
- MSI balíčky sú vlastne relačné databázy obsahujúce všetky informácie, ktoré Windows Installer potrebuje k nainštalovaniu a odinštalovaniu aplikácie

Štruktúra MSI balíčka



Fázy inštalácie

- *User interface* fáza – užívateľ má možnosť prispôsobiť si inštaláciu pre svoje potreby
- *Execute* fáza – samotná inštalácia
  - *Immediate mode* – vytvorenie vnútorného skriptu
  - *Deferred mode* – vykonanie skriptu v kontexte privilegovanej Windows Installer služby

- *Rollback* fáza – pre každú operáciu je vygenerovaná protioperácia, ktorá odstráni vykonanú zmenu v systéme.

Windows Installer XML (WiX)

- Sada nástrojov pre vytváranie MSI balíčkov z XML dokumentov

- 2004 Rob Mensching a spol.
- V súčasnosti stabilná verzia 3.5
- Open source, free, download
- Nástroje WiXu a ich vzájomná spolupráca

Keď použijete WiX

- Všetky spustiteľné súbory budú zabalené v jednom balíčku, čo uľahčí distribúciu aplikácie.
- Software bude automaticky registrovaný v Add/Remove Programs.
- Windows sa pri odinštalovaní postará o odstránenie všetkých komponent, z ktorých je váš produkt utvorený.
- Ak sú niektoré súbory náhodou odstránené, bude ich možné obnoviť voľbou Repair po kliknutí pravým tlačítkom na MSI balíček.
- Bude možné vytvoriť rôzne verzie inštalačných programov a zistiť, ktorá verzia bola nainštalovaná.
- Bude možné vytvoriť patch na aktualizáciu len časti aplikácie.
- Ak počas inštalácie dôjde k niečomu neočakávanému, bude možné vrátiť systém do predchádzajúceho stavu.
- Bude môžné vytvoriť Sprievodcu inštaláciou.

Práca s WiXom

- Vytvorte .wxs súbor v obyčajnom textovom editore (napr. Notepad) a následne ho pomocou nástrojov WiXu v príkazovom riadku skompilujte a zlinkujte do výsledného .msi súboru.
- Použite Votive – plug-in Visual Studia, ktorý poskytuje WiX IntelliSense, zvýrazňovanie syntaxe a šablóny WiX projektov.
- Použite produkty vytvorené priamo pre WiX (napr. WixEdit).

XML elementy

Každý WiX projekt musí obsahovať nasledujúce elementy:

- XML deklráciu
- element *Wix*, ktorý slúži ako koreňový element
- element *Product*, ktorý je následníkom elementu WiX a všetky ostatné elementy sú jeho potomkovia
- element *Package*
- element *Media*
- aspoň jeden element *Directory* s aspoň jedným elementom *Component* ako potomkom
- element *Feature*

XMLdeklarácia a element *Wix*

```
<?xml version=”1.0” encoding=”UTF-8”?>
<Wix xmlns=http://schemas.microsoft.com/wix/2006/wi>
…
</Wix>
```

- element *Wix* môže obsahovať atribút *RequiredVersion* - .wxs projekt nebude môcť byť skompilovaný, pokiaľ na cieľovom počítači nebude nainštalovaná požadovaná, príp. vyššia verzia WiXu

Element *Product*

- Je vnorený do elementu *Wix*
- Obsahuje charakteristiku inštalovaného softwaru (meno, jazyk, výrobcu, verziu,...)
- Atribút *Id* reprezentuje tzv. *ProductCode* – GUID jednoznačne identifikujúci váš software; Id="*"– pri každom preklade vytvorený nový GUID
- Atribút *Language*
- Atribút *Version* vo formáte *[MajorVersion].[MinorVersion].[Build].[Revision]*

Element *Package*

- Je vnorený do elementu *Product*
- Popisuje samotný inštalačný program
- Atribút *Compressed*=“yes”–všetky MSI zdroje budú zabalené do CAB súboru
- Atribút *InstallerVersion*–verzia nutná k inštalácií

Element *Media*

- Je popri elemente *Package* vnorený do elementu *Product*
- Tento element umožňuje rozdeliť balíček na viac častí, alebo nechať všetko v jednom kuse
- Pre každý element *Media* bude vytvorený samostatný CAB súbor

Element *Directory*

- Najjednoduchšie je vnoriť váš adresár do niektorého z preddeklarovaných adresárov
- Hierarchiu *Directory* elementov musíte vždy začať s elementom s TARGETDIR ako *Id* atribútom a SourceDir ako *Name* atribútom
- Pri vytváraní vlastného adresára musíte uviesť atribút *Name*, podľa ktorého bude pomenovaný
- Do adresára môžete niečo uložiť priamo cez element *Component*, alebo nepriamo použitím elementu *DirectoryRef* (atr. *Id* určuje požadovaný adresár)

Element *Directory* - priame vloženie

```
<Directory Id="TARGETDIR"
           Name="SourceDir">
  <Directory Id="ProgramFilesFolder">
     <Directory Id="MyProgramDir"
                Name="Install Practice">
           <Component ... />
     </Directory>
  </Directory>
</Directory>
```

Element *Directory* - nepriame vloženie

```
<Directory Id="TARGETDIR„
           Name="SourceDir">
  <Directory Id="ProgramFilesFolder">
     <Directory Id="MyProgramDir"
                Name="Install Practice" />
  </Directory>
</Directory>

<DirectoryRef Id="MyProgramDir">
  <Component ... />
</DirectoryRef>
```

Element *Component*

- Každý súbor musí byť pred inštaláciou zabalený v nejakom komponente
- Komponenty, identifikované GUIDom, umožňujú nájsť každý nainštalovaný súbor – informácie o nich sú pri inštalácií ukladané do registra
- Komponent obsahuje súbory, kľúče a hodnoty registra,...
- Komponent by mal obsahovať maximálne jeden súbor – súvislosť s atr. *KeyPath*
- Komponent sa inštaluje aj odinštalováva ako jeden celok!

Element *File*

- Reprezentuje akýkoľvek súbor
- Vždy by mal obsahovať atr. *Id*, *Source* a *KeyPath*
- Ak nešpecifikujete atr. *Name*, súbor bude pomenovaný tak, ako je uvedené v atr. *Source*
- Nastavenie atribútu *KeyPath* na hodnotu *yes* zabezpečuje prípradnú možnosť opravy

Vytvorenie prázdneho priečinku

```
<Directory Id="TARGETDIR"
           Name="SourceDir">
  <Directory Id="MyProgramDir"
           Name="Install Practice">
     <Component Id=“CMP_MyEmptyDir"
                Guid=“some_GUID“
                KeyPath=“yes“
           <CreateFolderDirectory=“MyEmptyDir"
     </Component>
     <Directory Id="MyEmptyDir"
                Name="Empty Directory“ />
  </Directory>
</Directory>
```

Elementy*RegistryKey* a *RegistryValue*

- Sú vnorené do elementu *Component*
- Možné použiť samotný element *RegistryValue;* pri viacerých zápisoch do toho istého kľúča registra je vhodné použiť element *RegistryKey* a doňho vnoriť elementy *RegistryValue*
- •Atr. *Action* elementu *RegistryKey* môže byť nastavený na hodnotu *createAndRemoveOnUninstall*, čo spôsobí odstránenie nielen hodnôt vzniknutých pri inštalácií, ale všetkého, čo daný kľúč obsahuje

Inštalácia služby

- Vytvorte element *Component* pre službu, ktorý bude obsahovať:
  - Element *File* obsahujúci .exe súbor so službou
  - Element *ServiceInstall*, ktorý službu zaregistruje
  - Element *ServiceControl*, ktorý službu spustí, zastaví alebo odinštaluje

Inštalácia služby - príklad

```
<Component Id=“MyServiceComponent"
           Guid="some_GUID">
  <File Id=“MyService" Name=“MyService.exe"
     KeyPath="yes“ Source=".\MyService.exe" />
  <ServiceInstall Id=“InstallMyService"
     Name=“testsvc" DisplayName=“MyService"
     Type="ownProcess" Start="auto"
     ErrorControl="normal" />
  <ServiceControl Id=“sc_MyService"
     Name=“testsvc" Start="install" Stop="both"
     Remove="uninstall" Wait="yes" />
</Component>
```

Element *Feature*

- *Feature* je skupina komponentov, ktoré sa majú nainštalovať naraz
- Vo všeobecnosti by v jednej *Feature* mali byť komponenty, ktoré na sebe závisia, prípadne vytvárajú jeden spoločný celok
- Jednotlivé komponenty sú do elementu *Feature* pridávané pomocou elementu *ComponentRef*, ktorý má atribút *Id* korešpendujúci s *Id* elementu *Component*

Ďalšie možnosti

- Element *util:Event Source* – v *Component*
- Element *Merge* v *Directory* + *MergeRef* vo *Feature*
- Element *Binary*
- Element *CustomAction*
- Element *InstallUISequence*
- Element *InstallExecuteSequence* (*ScheduleReboot*,...)
- Element *Upgrade*

Vhodná literatúra

- **Nick Ramirez:** WiX: A Developer's Guide to Windows Installer XML, October 2010

Otázky?

Ďakujeme za pozornosť☺