Aplikace – Základní

Petr Novák / novakpe@fel.cvut.cz / 19.02.2021

Obsah

1 Úvod

Pojmenování různých verzí technologií **.NET** není zcela jasné. Proto je vhodné na úvod toto trochu objasnit. Původně (zhruba v roce 2002) vznikla technologie s názvem **.NET Framework** určená výhradně pro operační systém MS/Windows. A jako stěžejní jazyk využívající **C#** O něco později začala vznikat technologie s názvem **Mono** jako varianta pro různé verze LINUXu. Verze **Mono** se později stala základem pro **.NET Xamarin** jenž je (v současné době) určen hlavně pro tvorbu aplikací na mobilní zařízení, tedy pro operační systémy „Google/Android“ a „Apple/iOS“. Později se skupina vytvářející **.NET Xamarin** připojila oficiálně k Microsoftu, aby se tyto dvě odnože / verze sjednotili.
Sjednocování však neproběhlo jedním krokem. Microsoft vydával postupně tzv. **.NET Standard**, což není nový framework, nebo programový kód, ale pouze soubor API standardů pro **.NET**. Proto lze na WWW dohledat konstatování typu: **.NET Framework verze-X** a **.NET Xamarin verze-Y** již implementují **.NET Standard verze-Z**. Takto tedy došlo postupným vývojem k jistému sjednocení **.NET Framework** a **.NET Xamarin**. Spojení však není úplné, **.NET Standard** pouze říká, jaké API obě verze poskytují a tedy jaký (zdrojový) kód lze mezi nimi přenést bez jakékoli úpravy (například komunikace pomocí TCPIP ano, ale komunikaci pomocí BlueTooth nikoli).
Další krok opravdu velkého sjednocování je nazván **.NET Core**. Jde tedy o jádro technologie **.NET** společné pro různé verze **.NETu** a to i přes různé operační systémy (v podstatě implementující nějaký „.NET Standard“ o kterém se již moc nemluví, protože to nemá velký význam). Nové aplikace jsou tedy založeny výhradně na **.NET Core** (jako základu .NETu) a přidávají se různé komponenty podle cílového operačního systému. Aplikace napsaná v **.NET Core** tedy běží na mnoha verzích nejen operačních systémů (MS/Windows, MS/Windows IoT, Linux, Google/Android, Apple/iSO, …), ale i na mnoha HW platformách (I86, I64, ARM, …). Při vývoji aplikace v **.NET Core** je často potřeba stahovat / instalovat knihovní balíčky (NuGet) podle cílového použití. **.NET Core** například neobsahuje podporu pro „SerialPort“, ale existují balíčky pro jeho podporu pro různé cílové operační systémy (MS/Winodws, LINUX, …). **.NET Core** (zatím) neobsahuje žádnou grafickou nadstavbu, jde pouze o knihovny technologie **.NET**.

Bohužel aplikace bez grafické nadstavby není moc užitečná (tedy mimo LINUXu). První **.NET Framework** obsahoval tzv. **WinForms** pro tvorbu dialogů, což byla nějaké plocha na kterou se umístují grafické komponenty (tlačítka, texty, …), nebo se na ni přímo kreslí (pera, štětce, obrazce, …). Později byla pro **.NET Framework** vytvořena grafická technologie **WPF** (Windows Presentation Framework). **WPF** není v žádném případě (zamýšlená úplná) náhrada **WinForms**, ale jako jiná varianta pro tvorbu GUI. **WPF** je založeno na **XAML**, což je popis grafického dialogu pomocí XML. Naprostá většina GUI v **.NETu** dnes používá **WPF**. Jak **WinForms** tak i **WPF** má své výhody na nevýhody, ale **WPF** stále více **WinForm** nahrazuje.
.NET Xamarin používá svoji grafickou technologii zvanou **Xamarin.Forms**. I když se názvem více blíží k **WinForms**, tak opak je pravdou, je to v podstatě skoro to samé co **WPF** (popis pomocí XAML, ale bohužel obsahuje často jiné názvy značek a strukturu).
Pro LINUX (zatím) není Microsoftem podporovaná GUI technologie. Existuje však několik možností, jak v **.NETu** na LINUXu vytvořit GUI (není však součástí tohoto textu).

Pokud se dnes vytváří aplikace s technologií **.NET** tak se lze setkat s pojmy:

- **.NET Core App / Console** (aplikace založená na **.NET Core** bez grafického rozhraní)
- **.NET Core + WPF** (aplikace založená na **.NET Core** umožňující vytvářet GUI pomocí **WPF**)
- **.NET Core + WinForms** (aplikace založená na **.NET Core** umožňující vytvářet GUI pomocí **WinForms**)
- **.NET Core + Xamarin** (aplikace založená na **.NET Core** umožňující vytvářet GUI pomocí **Xamarin.Forms**)

I když stále existuje původní **.NET Framework**, tak se jej nedoporučuje používat, protože již nebude dále rozvíjen, je zcela nahrazen pomocí **.NET Core**. **.NET Core** však neposkytuje (a zřejmě jen tak nebude poskytovat) vše co poskytoval **.NET Framework**, který byl silně vázán na operační systém MS/Windwos, z tohoto důvodu se občas ještě stále vytváří aplikace s použitím **.NET Framework** (jsou však možnosti jak toto obejít).
V dalším popisu tedy bude uvedena pouze technologie **.NET Core** a její případné rozšíření. V mnoha případech / textech se již ani neuvádí **.NET Core**, ale pouze **.NET** což může být poněkud matoucí.

Poznámky:

- **UWP** (Universál Windows Platform). Toto je typ aplikací určených zejména pro „App-Store“ a pro různé verze MS/Windows. Jde o **.NET Framework** plus další API jednotné přes různé typy MS/Windows. Dnes je v podstatě nahrazeno pomocí **.NET Core**, ale velmi často se v **.NET Core** využívají knihovny určené původně pro **UWP**.

2 Visual Studio

Pro tvorbu aplikací pro **.NETu** je nejlepší **Microsoft Visual Studio**. Existuje však několik verzi. Pro malé projekty je ideální **Visual Studio - Community**, které je pro malé projekty zcela zdarma a zcela postačující. Pro tvorbu aplikací v prostředí MS/Windows, tedy aplikací typu **.NET Core + WPF**, **.NET Core + WinForms**, **.NET Core + Xamarin** a **.NET Core App** zcela doporučuji **Visual Studio - Community**. Pokud je potřeba tvořit aplikace v **.NET Core** přímo na LINUXu zle použít **Visual Studio - Code**. Pozor na skutečnost, že toto není tak sofistikovaný nástroj jako **Visual Studio - Community** pro MS/Windows.

3 Vytvoření nového projektu

Při založení (nového) projektu je potřeba udělat rozhodnutí:

- Zda je aplikace vytvářena pouze pro **MS/Windows**. Lze použít **.NET Core + WinForms** nebo **.NET Core + WPF** (nejvíce doporučeno). Případně pouze konzolovou aplikaci **.NET Core App / Console**.
- Zda je aplikace vytvářena pro mobilní zařízení **Google/Android** a **Apple/iOS**. Je potřeba použít **.NET Core + Xamarin** (pouze).
- Zda je aplikace vytvářena pro různé operační systémy bez GUI. Lze použít **.NET Core App / Console** (existují však možnosti tvorby GUI i v LINUXu).

Postup tvorby nového projetu je následující (Visual Studio 2019):

- Na úvodní obrazovce vybrat **Vytvořit nový projekt**.
- Na další obrazovce vybrat:
    - Jazyk: **C#**
    - Cílový systém: **Windows** nebo **Android**
    - Typ projektu raději ponechat na **Všechny typy projektů**

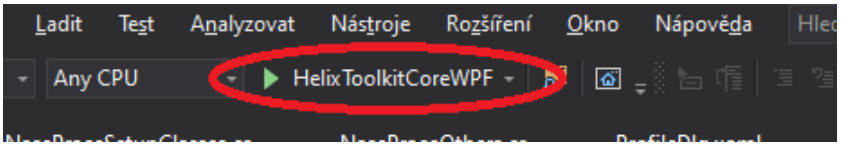
Pro MS/Windows je ideální projekt typu **WPF App (.NET)** případně **Windows Forms App (.NET)**. Jedná se tedy o aplikace založené na **.NET Core**. Pokud to není nezbytně nutné tak nedoporučuji vytvářet projekty typu **Aplikace WPF (.NET Framework)** nebo **Aplikace Windows Forms (.NET Framework)**.
Pro mobilní zařízení je ideální projekt typu **Mobilní aplikace (Xamarin Forms)** založená rovněž na **.NET Core**. Velmi doporučuji vytvářen obecnou aplikaci **Mobilní aplikace (Xamarin Forms)**, tedy v podstatě společnou pro **Google/Android** a **Apple/iOS** a nikoli pro každý systém samostatně.

Stručně řečeno:

- Pro MS/Windows vytvářet projekty **WPF App (.NET)**, tedy .NET Core + WPF.
- Pro Google/Android vytvářet projekty **Mobilní aplikace (Xamarin Forms)**, tedy .NET Core + Xamarin.Forms.

4 Kompilace

Nejjednodušší způsob kompilace projektu je jeho spuštění pomocí zelené šipky v horních nástrojích **Visual Studio**. Vpravo od zelené šiky je (skoro) vždy uveden název aktivního projetu.



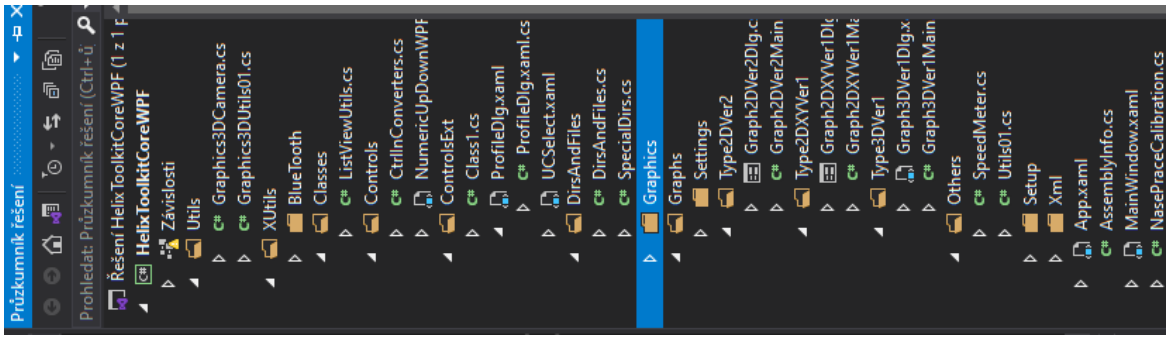
V některých případech, zejména pokud jsou do projektu přidány odkazy na jiné / další projekty, tak tímto stylem ne vždy dojde ke kompilaci všech změněných souborů. V tomto případě je vykonat tzv. re-kompilaci celého projektu pomocí: **Sestavit** (horní menu) – **Znovu sestavit řešení** (položka v rozbaleném menu).

Poznámka:

- Pokud se nějaký soubor v projektu modifikuje mimo **Visual Studio** (například úpravy obrázku), tak je vždy doporučeno celý projekt re-kompilovat.

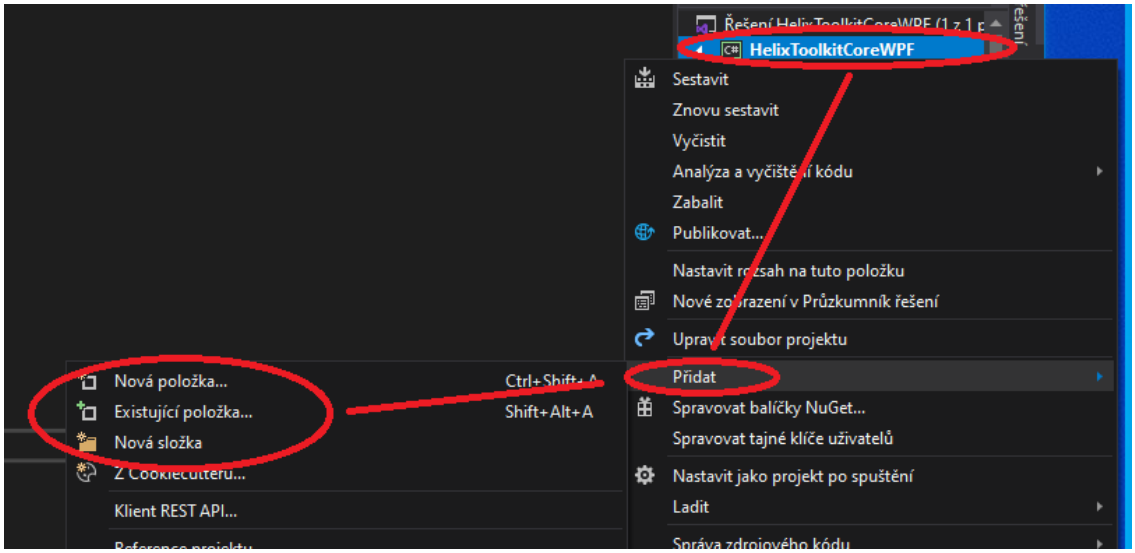
5 Adresáře a soubory

Všechny zdrojové soubory jsou v projektu umístěny v nějaké adresářové struktuře. Tak je vytvořen velmi dobrý přehled / orientace po projektu. Adresáře se nazývají podle nějaké části projektu a obsahují (zdrojové) souboru příslušející k této části projektu.



(Na ležato pouze pro úspory místa.)

Do projektu je tedy vždy potřeba přidávat vlastní zdrojové soubory a ty organizovat do nějakých adresářů (obdobně jako zřejmě v každém jiném programovacím jazyce / prostředí). Nový adresář / soubor lze do projektu přidat klikem levého tlačítka myši na položku, pod kterou bude přidán. Pokud je potřeba přidat adresář / soubor, tak klik pravým tlačítkem myši na název projektu (nikoli celého **Řešení** / Solution).

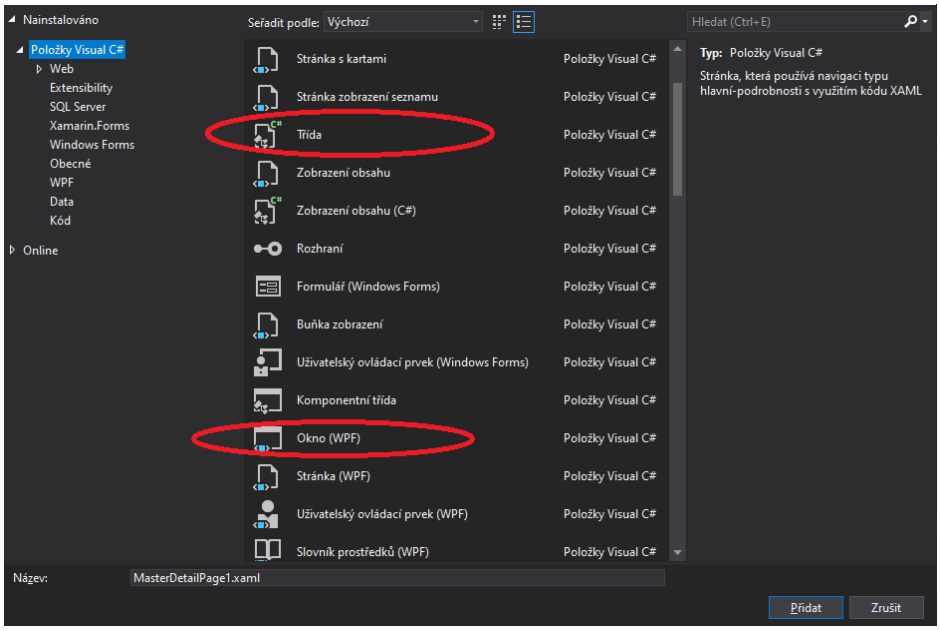


Zobrazí se lokální menu a zde najet myší na **Přidat**, zobrazí se další pod-menu s těmito možnostmi:

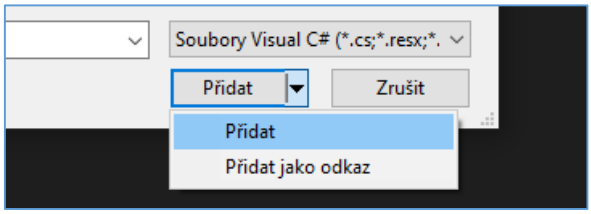
- **Nová položka …** – Zobrazí dialog pro výběr typu nově vytvářeného zdrojového souboru. Zdrojové soubory
- **Existující položka …** – Zobrazí dialog pro výběr již existujícího souboru pro přidání do projektu.
- **Nová složka** – Vytvoří nový adresář. Po vytvoření lze ihned přejmenovat. Adresáře jsou samozřejmě vnořovat.

Nová položka – Přidání nové položky, tedy zdrojového souboru do projektu. Do projektu lze přidat mnoho typů souborů. Soubor je vložen o úroveň níže, než byla vybraná v projektu pomocí myši. Ty nejčastější typy souborů jsou asi tyto dva:

- **Třída** – Běžný zdrojový soubor typu **C#**. Bude obsahovat nějaké úvodní **using**, název pro **namespace** podle místa jeho vytvoření v projektu a název třídy podle názvu zadaného souboru.
- **Okno (WPF)** – Samostatný dialog typu WPF, který lze v projektu zobrazit (například pro výběr položek uživatelem).



Existující položka – Vložení do projektu již existujícího souboru (zdrojový soubor, obrázek, …). Soubor je vložen o úroveň níže, než byla vybraná v projektu pomocí myši.



Při potvrzení pomocí tlačítka Přidat lze zvolit, jak bude vybraný soubor do projektu přidán:

- **Přidat** – Vybraný soubor bude do projektu zkopírován. V projektu tedy bude vytvořen nový soubor se jménem a obsahem původního / vybraného souboru. Jeho modifikace tedy neovlivní původní soubor. (Doporučeno pro začátečníky.)
- **Přidat jako odkaz** – Vybraný soubor do projektu nebude zkopírován, ale bude do projektu pouze přidán odkaz na původní / vybraný soubor. Všechny modifikace tohoto souboru v projektu tedy ovlivní obsah původního soubor. Toto je velmi vhodné, pokud je jeden soubor sdílen mezi více projekty (například mezi Windows a Xamarin).

6 Balíčky NuGET

Původní verze **.NET Framework** v podstatě obsahovala (skoro) vše co bylo potřeba (pro naprostou většinu aplikací) a tedy zcela minimálně bylo potřeba stahovat nějaké další knihovny / balíčky. Nástupem **.NET Core** se toto poněkud změnilo. Je to hlavně dáno jeho multi-platformostí, kdy se k základu **.NET Core** stahují balíčky podle toho na jaké cílové platformě jsou podporovány.

Tyto balíčky se nejčastěji stahují z NeGet úložiště a postup je následující:

- Pravé tlačítko myši na **Řešení** / Solution (raději ne projekt), tedy na ten první název v průzkumníku řešení.
- Z lokálního menu vybrat **Spravovat balíčky NuGet pro řešení …**
- Zobrazí se stránka, nastavit se na záložku **Procházet** (první vlevo v horní řadě záložek)
- Pod **Procházet** lze zapsat hledaný výraz, například **Systém.IO.Ports**
- Hledaný balíček označit
- Vpravo se zobrazí tabulka obsahující celé **Řešení** a v něm všechny dostupné **Projekty**
- Zde lze vybrat, do jakých projektů v řešení se bude označený balíček instalovat (lze tedy vybrat pouze některé projekty, pokud je již otevřeno několik současně)
- Pod seznamem projektů jsou tlačítka **Odinstalovat** (pokud již je instalován) a **Nainstalovat** (pokud ještě není instalován).

V další záložce **Nainstalováno** lze prohlížet, jaké balíčky jsou do jakých projektů již instalovány a lze je tedy zde i odebrat. V záložce **Aktualizace** se zobrazují dostupné nové verze již instalovaných balíčků.

Poznámky:

- Při instalaci a zejména při aktualizaci nějakého balíčku velmi doporučuji si celý projekt zálohovat (stačí například pouze zaZIPovat). Občas se povede instalovat balíček nekompatibilní s danou verzí **.NETu** použitou v projektu, případně s jiným již instalovaným balíčkem a jejich obnova nebývá zcela snadná.
- V tomto případě je lepší všechny balíčky odinstalovat, projekt uzavřít a pět jej otevřít a instalovat potřebné balíčky znova.

7 WPF vs WinForms

Rozdílností mezi **WPF** a **WinForms** je skutečně mnoho a nebudou zde popsány. Hlavní rozdíl je asi ten, že **WPF** umožňuje vytvářet dialogy, které se snadno roztahují podle velikosti obrazovky, obsahují například styly, umožnují svazovat GUI s tady a poskytují 2D i 3D objektovou grafiku. Na druhou stranu se nijak nehodí pro rychlé změny (3D/3D grafiky) a jiné typy kreslení. Naopak **WinForms** jsou ideální pro real-time grafy, ale celkem složité se v nich vytváří automatické přizpůsobení velikosti obrazovky. Pokud není žádný pádný důvod, tak doporučuji využívat GUI zásadně typu **WPF** a pouze v případě potřeby, například kreslení rychlého / real-lime grafu otevřít ve **WPF** aplikaci nový dialog typu **WinForms**.

8 Aplikace WPF + WinForm

Pokud je **.NET Core** aplikace vytvořena jako **.NET Core + WPF** nebo **.NET Core + WinForms**, tak nelze úplně snadno používat i to druhé (WPF / WinForms). Existuje však velmi snadné řešení. Při vytvoření aplikace je vytvořen projekt (adresáře a soubory) a v něm umístěn projektový soubor s koncovkou „.csproj“. Ten obsahuje informace z čeho je projekt sestaven a jak je nakonfigurován.

Pokud je vytvořen projekt jako **.NET Core + WPF** v souboru „csproj“ jsou (zhruba) následující řádky:

```
<PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>netcoreapp3.1</TargetFramework>
```

```
  <UseWPF>true</UseWPF>
</PropertyGroup>
```

Pokud je vytvořen projekt jako **.NET Core + WinForms** v souboru „csproj“ jsou (zhruba) následující řádky:

```
<PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <UseWindowsForms>true</UseWindowsForms>
</PropertyGroup>
```

Pokud potřebujeme do aplikace typu **WPF** přidat komponenty typu **WinForms** (případně naopak), tak stačí doplnit ještě další typ aplikace, tedy například takto:

```
<PropertyGroup>
  <OutputType>WinExe</OutputType>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <UseWPF>true</UseWPF>
  <UseWindowsForms>true</UseWindowsForms>
</PropertyGroup>
```

Tedy přidáním řádku **<UseWindowsForms>true</UseWindowsForms>** nebo **<UseWPF>true</UseWPF>** se přidají příslušné komponenty do projektu / aplikace.

9 Využití možností UWP v .NET Core

Poněkud dřívější verze (vytvářených) aplikací tzv. UWP (Univerzal Windows Platform) sjednocovaly a tedy poskytovaly nějaké schopnosti Windows pro různé typy aplikací běžících na různých zařízeních. Jedná se však stále pouze o aplikace pro Windows, ale jde o (typy) Windows běžící na různých platformách jako jsou: stolní počítače, tablety, mobilní telefony atd. a to i na různých typech procesorů / HW (x32/x64 nebo ARM). Podpora UWP tedy poskytuje / sjednocuje přístup zejména pro HW periferie při vytváření aplikací. Jednou z velmi využívaných periférií je například BlueTooth, pro který není jednotná podpora v různých verzích Windows, avšak UWP jej vhodně sjednocuje / zastřešuje. Z tohoto důvodu se občas naskýtá potřeba využít schopností UWP v **.NET Core**.

Do aplikace / projektu lze přidat následující soubory:

- C:\Program Files(x86)\Reference Assemblies\Microsoft\Framework\.NETCore\v4.5\System.Runtime.WindowsRuntime.dll
- C:\Program Files(x86)\Windows Kits\10\UnionMetadata\10.0.18362.0\Windows.winmd (umístění tohoto souboru se může lišit podle verze instalovaného „SDK WIN10“)

Soubory jsou přidány / připojeny do projektu tímto postupem:

- „Průzkumník řešení“ (záložka vpravo) ve Visual Studio
- V projektu pravé tlačítko myši na „Závislosti“ – „Přidat referenci projektu …“ (položka v menu)
- „Procházet“ (tlačítko vpravo dole) – najít požadovaný soubor na disku (jeden po druhém).

Požadavky:

- Musí být tedy instalováno **.NETCore v4.5** (případně asi vyšší).

- Musí být instalován Windows 10 SDK Kit například uvedené verze (nebo novější). Jak nainstalovat: „Nástroje“ (menu nahoře ve Visual Studio) – „Získat nástroje a funkce …“ (položka v menu) – přepnout zobrazení do „Jednotlivé komponenty“ (záložka nahoře na dialogu) – najít „Windows 10 SDK (XX.X.XXXXX.X)“ a nainstalovat, klidně verzi „10.0.18362.0“.
- Pokud bude potřeba připojit do projektu jinou / novější verzi je potřeba původní ze závislostí odebrat (pravé tlačítko myši na ni a „Odebrat“).

Po přidání souborů jsou řádky v projektovém souboru „.csproj“ a to zhruba následovně:

```
<ItemGroup>
  <Reference Include="System.Runtime.WindowsRuntime">
    <HintPath>..\..\..\..\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETCore\v4.5\System.Runtime.WindowsRuntime.dll</HintPath>
  </Reference>
  <Reference Include="Windows">
    <HintPath>..\..\..\..\Program Files (x86)\Windows Kits\10\UnionMetadata\10.0.18362.0\Windows.winmd</HintPath>
    <IsWinMDFile>true</IsWinMDFile>
  </Reference>
 </ItemGroup>
```

Případně lze tyto řádky do souboru „csproj“ přímo nakopírovat a pouze upravit číslo verze pro **.NETCore** a **SDK WIN10**.