UNIVERZITA J. E. PURKYNĚ V ÚSTÍ NAD LABEM
Přírodovědecká fakulta

Programování pro GUI

RNDr. Petr Kubera, Ph.D.

Ústí nad Labem 2020

Předmět: Programování pro GUI

Studijní program: Aplikovaná informatika

Klíčová slova: Programování GUI, události, WPF

Anotace: Kurz je zaměřen na vizuální programování v desktopových prostředích. Důraz je kladen na interaktivní graficky orientované aplikace a na návrh grafických uživatelských rozhraní. Student je prostřednictvím tohoto kurzu seznámen s filozofií událostmi řízeného programování. Cílem je poskytnout přehled o možnostech tvorby GUI napříč platformami. Primární zaměření je však na platformu .NET, resp. její grafickou nadstavbu WPF.

Jazyková korektura nebyla provedena, za jazykovou stránku odpovídá autor.

Autor: RNDr. Petr Kubera, Ph.D.

Obsah

Úvodní slovo

Předmět je koncipován jako přehledový kurz programování grafických uživatelských rozhraní (GUI) se zaměřením na desktop. V současnosti se používá celá řada frameworků a knihoven, jejichž cílem je usnadnit návrh a vývoj GUI. Liší se jak použitými jazyky, tak celkovou filozofií, kde uživatelské rozhraní se definuje buď na úrovni kódu, nebo pomocí nějakého popisného jazyka (XAML, QML). Co je ve většině případů společné, je model událostmi řízeného programování. Různé frameworky k tomu používají různé prostředky, v podstatě však jde o to, aby některá z komponent (např. tlačítko), byla při detekované události (např. kliknutí na něj) ji schopna obsloužit. V první části kurzu, lekce první až osmá, se zaměříme na ekosystém .NET a grafickou nástavbu WPF (Windows Presentation Foundation). Seznámíme se nejen s vývojem GUI, ale i s nástroji např. databáze, webové služby, které budeme využívat a které budou začleněny v našich aplikacích. Neboť chceme, aby naše aplikace i něco smysluplného dělala. Též se podíváme trochu hlouběji do prostředí .NET, kde se seznámíme se základy vícevláknových (asynchronních) aplikací a tvorbou pluginů a reflexí. Budeme používat MS Visual Studio ke stažení zde. Druhá část kurzu je zaměřena na tvorbu GUI v jiných frameworcích a zde bychom měli využívat již naučených principů z .NET. Poznamenejme, že vůči sylabu jsou sloučeny knihovny pro Javu do jedné kapitoly.

Zápočet je možné získat za prezentaci některých témat v rámci semináře a tvorbu vlastní funkční GUI aplikace. V případě studentů kombinované formy studia není prezentace v rámci semináře, ale je součástí zkoušky. Aplikaci je **nutné** si nechat schválit vyučujícím a v principu by měla používat buď webové služby, či databázi. Další požadavky na aplikaci se odvíjejí od zvoleného tématu.

Zkouška se skládá z diskuse nad aplikací. Hodnocena je jak zvolená architektura aplikace, tak znalost použitých technologií. Hodnocena je i schopnost úpravy aplikace a dodání dalších funkcionalit. Po dohodě je možné použít technologie, které na semináři nebyly probrány.

Splnění kurzu

Zde naleznete několik témat pro motivaci k Vašim prezentacím a seminárním programům. Témata prezentací a programů je nutné konzultovat.

Ukázky témat možných prezentaci

1. Jazyk XAML – přehled, elementy, pozicování, události
2. Jazyk QML a knihovna Qt/PyQt – přehled
3. Přehled ORM frameworků – ukázky a použití
4. Přehled a použití knihovny Json.NET
5. Přehled tvorby GUI v Javě – Swing, Java FX

Ukázka zadání zápočtového programu

Navrhněte a implementujte aplikaci, která stahuje kurzovní lístek ČNB (např. pomocí api). Aplikace umožní vybrat měnu, zvolit si interval sledování. Aplikace vykreslí graf kurzu a spočte základní metriky (např. průměr, rozptyl) stažené časové řady.

1 Událostmi řízené programování

Klíčovým pojmem je událostmi řízené programování. V této kapitole se dozvíte následující témata, podívejte se na uvedené odkazy.

CÍLE KAPITOLY

- Návrhový vzor observer.
- Pojem delegát a ukazatel na funkce.
- Co to je událost a jaká je její realizace v prostředí .NET.
- Šíření a obsluha událostí.
- Tvorba naší první WPF aplikace. Prostudujte si i použité komponenty.

KLÍČOVÁ SLOVA

události, návrhový vzor observer, události v GUI

ÚKOLY

1. Podívejte se na delegáty *Func<T,TResult>* a *Action<T>* a jejich použití.
2. Prostudujte si použití rozhraní *IObservable* a *IObserver*
3. Implementujte událostmi řízený program. Kde objekt *Zapisovatel* zapisuje náhodná čísla do souboru. Objekt generuje události *Zapsano*, tj. je jejich vydavatel. Dále implementujte objekty počítající statistiky (průměr, rozptyl, medián) nad čísli v souboru. Tyto objekty jsou odběrateli události *Zapsano*.
4. Implementujte pomocí WPF jednoduchý řešič kvadratické rovnice. Nepovinné rozšíření je nakreslit pomocí graf zachycující tvar dané paraboly.

OTÁZKY

1. Popište schéma návrhového vzoru Observer.
2. Vysvětlete použití delegátů v .NET.

OTÁZKY K ZAMYŠLENÍ

1. Který z objektů WPF *window* a *button* je pozorovatelem a který pozorovaným?

SHRNUTÍ

Po prostudování byste měli být schopni:

- Navrhnout a implementovat jednoduchou konzolovou aplikaci, kde budou objekty komunikovat pomocí událostí.
- Navrhnout a implementovat jednoduchou GUI aplikaci, která obsahuje několik komponent (tlačítka, textboxy) a je schopná přebírat informace od uživatele a reagovat na stisk - ukončení, kontrola zadaného, výpočet.

ODKAZY NA LITERATURU

- Základy tvorby WPF aplikací naleznete zde.
- Informace o událostech v .NET naleznete zde.

MÍSTO PRO VAŠE POZNÁMKY

2 Jazyk XAML – vizuální návrh aplikace

V této kapitole se zaměříme na návrh grafického uživatelského rozhraní. V zásadě máme dvě možnosti, buď rozhraní popsat pomocí kódu, nebo pomocí značkovacího jazyka. V současnosti se preferuje druhá možnost, v případě technologie WPF se používá značkovací jazyk XAML. Design aplikace můžete navrhnout buď pomocí grafického designéru (používáme MS Visual Studio), nebo přímo pomocí editoru kódu. Postupem času dospějete pravděpodobně k tomu, že budete preferovat přímo editor kódu XAML. Prostudujte si následující témata.

CÍLE KAPITOLY

- Popis tvorby WPF aplikace, pokud jste si to již neprostudovali v předchozí kapitole, je zde.
- Stručný popis jazyka XAML naleznete zde.
- Úvodní popis komponent (*controls*) je uveden zde, nebo jejich výčet. Detailnější pohled na model komponent ve WPF je zde.
- Pozicování komponent je popsáno zde a zde.

KLÍČOVÁ SLOVA

komponenty, XAML

ÚKOLY

1. Navrhněte aplikaci, která emuluje chování jednoduché kalkulačky. Zaměřte se hlavně na pozicování prvků pomocí komponent *StackPanel* a *Grid*.
2. Navrhněte aplikaci, která umožní vizualizovat *csv* soubor s následujícím formátem dat:

   ```
   jmeno1, prijmeni1, datum narozeni1
   jmeno2, prijmeni2, datum narozeni2
   ...
   ```

   Aplikace používá taby (*TabPanel*) a na každém z nich nabízí jiný pohled na data. Např.

   - Pohled pomocí *GridView*.
   - Stromový pohled (*TreeView*), kde uzly jsou roky narození.

3. Prostudujte si základní principy *databindingu* (vázání dat na komponenty) zde a použijte je v předchozím příkladu.

OTÁZKY

1. Vysvětlete princip použití jazyka XAML.
2. Uveďte základní vlastnosti a události komponent *button, grid, label, listview.*
3. Vysvětlete co to je databinding.

SHRNUTÍ

Po prostudování byste měli být schopni:

- zvolit vhodné komponenty pro danou situaci
- znát základní chování komponent

ODKAZY NA LITERATURU

- Detailní přehled jazyka XAML je zde.

MÍSTO PRO VAŠE POZNÁMKY

3 Vlastních grafické komponenty

Při tvorbě aplikací pravděpodobně dospějete k tomu, že Vám nebude stačit stávající funkcionalita nabízených komponent. Jedná se buď o vzhled, či nějakou přidruženou funkčnost. Zde se vyplatí buď změnit styl komponenty, či si jí přímo odvodit pomocí dědění a přepsat některé z poděděných metod od předka. Jako předka volíme nejčastěji komponentu s co nejpodobnějším chováním (použitím), nebo *Control* (což je dosti pracné na implementaci). Další častý způsob je vytvoření komponenty skládáním dílčích z komponent. Toto je situace, kdy víte, že budete opakovaně používat nějaké seskupení komponent, např. pro zadávání údajů o člověku potřebujeme opakovaně zadávat jméno, příjmení a datum narození, tak si je možné na to přímo vytvořit komponentu a nedělat to pokaždé znovu.

CÍLE KAPITOLY

- Úprava vzhledu komponenty pomocí vlastní šablony a přidání trigerů zde.
- Vytváření uživatelských prvků odvozením z *UserControl*
- Vytvoření uživatelských prvků odvozením od *Control* nebo jejích podtříd, viz zde.

KLÍČOVÁ SLOVA

vlastní komponenty, UserControl

ÚKOLY

1. Vyzkoušejte pomocí stylů úpravu tlačítka tak, aby se změnil font, když dostane fokus.
2. Navrhněte vlastní komponentu pro zadávání údajů o člověku. Implementujte i validaci jednotlivých polí, inspirujte se např. zde.

OTÁZKY

1. Co to je *ControlTemplate* a trigger?
2. Co to je *VisualState*?
3. Jaký je rozdíl při vytváření komponent pomocí dědění od *UserControl* a od *Control*.

SHRNUTÍ

Po prostudování byste měli být schopni:

- Upravovat styly komponent.
- Psát vlastní jednoduché komponenty.

ODKAZY NA LITERATURU

- Porovnání jednotlivých přístupů vytváření ovládacích prvků je zde.
- Přehled stylů komponent zde.

4 Asynchronní a vícevláknové aplikace

Pro tvorbu aplikací, které jsou výpočetně náročné, nebo čekají na nějakou vnější událost, např. načtení z databáze, je nutné spouštět danou operaci na jiném vlákně. Obsluha událostí aplikace totiž běží na jednom vlákně, a když bude hlavní vlákno aplikace na něco čekat, tak bude aplikace "zamrzávat". Bohužel, je zde obvykle ještě jeden problém (ve WPF je) a to je, že update stavů komponent musí být vykonán na hlavním vlákně aplikace. Toto vede k nutnosti po skončení výpočtu, nebo během výpočtu (informace o průběhu výpočtu), přepínat mezi výpočetním vláknem a vláknem hlavním. Problematika synchronizace vláken je poměrně komplikovaná a někdy vede k nehezkému chování aplikace (*deadlock*). Naštěstí WPF nabízí mechanismy jak toto elegantně řešit. Jedním z nich je použití úloh *Task* a dispečera úloh (*Dispatcher*).

CÍLE KAPITOLY

- Základy tvorby vláken (třída *Thread*) jsou popsány zde.
- Úlohy (*Task*) a asynchronní operace jsou popsány zde.
- Použití dispečera je uvedeno zde a zde.
- Z dnes již zastaralé technologie *WinForms* nám zůstal tzv. *BackgroundWorker*, viz např. tady.

KLÍČOVÁ SLOVA

Vlákna, asynchronní operace, tasky, dispatcher

ÚKOLY

1. Seznamte se s rozšířením .NET pomocí *async* a *await*, viz zde.
2. Implementujte vlastní (zjednodušenou) třídu *BackgroundWorker*.
3. Implementujte hru Game of Live.

OTÁZKY

1. Co to je *ThreadPool*?
2. Které asynchronní operace v .NET znáte?

OTÁZKY K ZAMYŠLENÍ

1. Jaký je rozdíl mezi vlákny a tasky?

SHRNUTÍ

Po prostudování byste měli být schopni:

- implementovat asynchronní operace jak v konzolových, tak v GUI aplikacích
- rozhodnout a umět vybrat vhodnou metodiku pro tvorbu asynchronicity.

MÍSTO PRO VAŠE POZNÁMKY

5 Relační databáze a ORM frameworky

Abyste se mohli připojit k nějaké databázi, je třeba mít k dispozici třídy, které zapouzdřují přístup k datům. V ekosystému .NET je to *ADO.NET*. Můžete používat většinu běžných databázových systémů, nicméně nejlepších (nejsnadněji) výsledků dosáhnete s MS produkty. Zde se zaměříme na provázání s MS SQL Server, pro ostatní DB systémy je to podobné, jen je třeba upravit jméno dané třídy (např. *OracleConnection* pro páci s DB Oracle, místo *SqlConnection*). Pro práci s DB systémem se využívají v zásadě dva principy. První je to, že pracujete přímo s databází. Vytvoříte si připojení pomocí třídy *SqlConnection* a vyvoláváte přímo SQL příkazy pomocí třídy *SqlCommand*. Výsledky SQL příkazu, pokud jsou potřeba, jsou pak vyčítány např. pomocí třídy *SqlDataReader*. Druhý přístup je použití *objektově relačního přístupu-modelu* (ORM). Zde programátor pracuje přímo s objektovou reprezentací položek v databázi a právě daný framework je zodpovědný za převod položek databáze na seznamy objektů při načítání a obráceně za převod objektů do řádků databáze při ukládání. V současnosti se používá *Entity framework*. Tento framework je nutné si doinstalovat pomocí *nuget* balíčků. Pro snadnou manipulaci s iterátory objektů je používána technologie *LINQ*.

CÍLE KAPITOLY

V této kapitole se dozvíte:

- Popis ADO.NET, co to je, jaké má možnosti a jak funguje je zde.
- Popis základních poskytovatelů přístupu na DB je zde.
- Použití třídy *SqlCommand* je k zde. Pokud máte mezery v SQL, lze si je doplnit.
- Použití Entity frameworku, přístupy *CodeFirst, DatabaseFirst* viz zde.

KLÍČOVÁ SLOVA

databáze, SqlConnection, SqlCommand, Entity framework

ÚKOLY

1. Podívejte se na objekty *DataSet*, *DataTable* a jejich použití zde.
2. Prostudujte si použití *LINQ To SQL*
3. Vytvořte si jednoduchou databázi s alespoň dvěma tabulkami a implementujte aplikaci, která vám umožní zobrazovat a modifikovat data v databázi. Vyzkoušejte si oba dva výše uvedené přístupy (Entity framework vs. SqlCommand). Jako motivační úlohu můžete

uvažovat databázi měst, kde máte ke každému městu přiřazeny některé meteorologické údaje v čase. Aplikace umožní přidávat/odebírat město a přidávat k danému dni teplotu a tlak a zobrazovat některé agregované údaje pro město, např. průměrná teplota.

OTÁZKY

1. Popište rozdíly mezi oběma přístupy k databázím.
2. Jak souvisí iterátory s třídou *SqlDataReader*?
3. Jak můžete svázat parametry SQL příkazu v *SqlCommand*?
4. Co to je *connection string*?

SHRNUTÍ

Po prostudování byste měli být schopni:

- Znát a používat třídy SqlCommand, SqlDataReader, SqlConnection.
- Porozumět principům ORM frameworků.
- Znát základy použití Entity frameworku.

ODKAZY NA LITERATURU

- Popis jak připojit různé datové zdroje do Vaší aplikace naleznete zde.

MÍSTO PRO VAŠE POZNÁMKY

6 Zpracování XML, JSON, použití webových služeb

Webové služby umožňují vzájemné předávání dat mezi zařízeními na počítačové síti. Samotná webová služba je popsána pomocí formátu *WSDL*. Zařízení si nejčastěji předávají data v SOAP formátu, což je protokol popsaný XML, nebo pomocí formátu *JSON*. Je tedy nutné umět oba formáty číst a extrahovat z nich podstatné informace. Pro jejich čtení, případně zápis, používáme buď specializované knihovny třetích stran (např. *Json.NET*), nebo specializovaných objektů *XmlDocument*, nebo *XDocument* jež jsou součástí .NET. Při procházení dokumentů je možné používat technologii LINQ, např. *Linq to XML*, *Linq to JSON*.

Pro webové služby se používá technologie *WCF*, případně pro posílání a příjem dat, je možné použít třídu *WebClient*. Pokud byste toužili napsat si vlastní webovou službu, tak popis její tvorby je zde.

CÍLE KAPITOLY

Seznámíme se s následujícími tématy:

- Formát JSON.
- Orientační znalost knihovny JSON.NET.
- DOM (přístup a jeho použití v třídě XDocument včetně napojení na LINQ. Orientační znalost třídy XDocument.
- Přístup *SAX* pomocí tříd *XmlReader* a *XmlWriter*, viz zde a zde.
- Použití třídy *WebClient* pro stahování dat viz zde.

KLÍČOVÁ SLOVA

webová služba, XML, JSON, XDocument

ÚKOLY

1. Prostudujte si vytvoření webové služby a její připojení na klienta pomocí proxy objektu zde.
2. Navrhněte a implementujte aplikaci pro stahování informací o počasí pomocí api popsaného zde.

OTÁZKY

1. Jaký je rozdíl mezi přístupem SAX a DOM v rychlosti?
2. Jaký je rozdíl mezi třídami XDocument a XmlDocument?

SHRNUTÍ

Po prostudování byste měli být schopni:

- Číst a strojově zpracovávat jednoduché XML a JSON dokumenty.
- Připojit se k existující webové službě a číst z ní data.

MÍSTO PRO VAŠE POZNÁMKY

7 Reflexe a tvorba pluginů aplikací

Reflexe je technologie umožňující programovacím jazykům zjistit si informace o použitých datových typech za běhu. V .NET je to realizováno tak, že v *assembly*, více viz zde, daného programu jsou uloženy metainformace o použitých typech. K tomu slouží celá řada tříd, které jsou v *System.Reflection*. Tyto nám umožňují zjistit o třídách obsažených v binárních souborech pro .NET (s příponou .exe, .dll) řadu užitečných informací, jako jsou jméno, implementované rozhraní, seznam metod a jejich parametrů, vlastnosti a to vše i v případě, že mají omezenou viditelnost (*private*). Dokonce je možno vytvořit si za běhu instance takových tříd a měnit jejich atributy přímo pomocí mechanizmu reflexe (i ty privátní).

Pluginem rozumíme zásuvný modul aplikace, který rozšiřuje její funkčnost. K tomuto je možné použít výše zmíněnou reflexi. Myšlenka je v celku jednoduchá, mezi aplikací a jejím rozšiřujícím modulem musí být uzavřený *kontrakt*, tj. je dohodnuto, jak bude vypadat rozhraní dané funkcionality. Aplikace si pak pomocí reflexe prohledává jednotlivé dodané knihovny (dll) a když nalezne třídy odpovídající danému interface, tak si je instancuje a může je používat.

CÍLE KAPITOLY

Seznamte se s následujícími tématy:

- Seznamte se s metodou *GetType()* a její návratovou hodnotou - třídou *Type*. Zaměřte se na metody typu *Get** pro zjištění informací o metodách, vlastnostech, rozhraních.
- V dokumentu zde se zaměřte na následující metody a jejich použití:
  1. *Assembly.GetModule(), Module.GetType()*
  2. *GetInterface(), FindInterfaces()*
- Seznamte se s třídou *Activator* a jejím použitím.
- Tvorba pluginů je popsána zde a zde.

KLÍČOVÁ SLOVA

reflexe, pluginy, typy, aktivátor

ÚKOLY

- Implementujte jednoduchou kalkulačku, kde jsou využity pluginy pro další dodané funkce. Např. plugin dodávající goniometrické funkce.

1. Aplikace si sama po startu vyhledá seznam dostupných pluginů. Zde je dobré mít třídu, která vám bude poskytovat seznam nalezených funkcí z pluginů.
2. Aplikace si sama přidá tlačítka a jejich obsluhy dle načtených pluginů.

- Zamyslete se, jak je umožněno grafickému návrhovému prostředí (např. *designer* ve VS) používat vámi vytvořené komponenty.

SHRNUTÍ

Po prostudování byste měli být schopni:

- Porozumět základním principům reflexe a jejímu použití.
- Tvořit jednoduché pluginy.

OTÁZKY

1. Vysvětlete, proč není dobré mít nastavená různá přístupová hesla přímo v kódu.
2. Proč použití privátních atributů není bezpečné za každých okolností.
3. Popište mechanismus pluginů v .NET.

OTÁZKY K ZAMYŠLENÍ

1. Vysvětlete, jak je umožněno grafickému návrhovému prostředí (např. *designer* ve VS) používat vámi vytvořené komponenty.

MÍSTO PRO VAŠE POZNÁMKY

8 Globalizace, lokalizace a konfigurace aplikací, použití zdrojů

Při psaní aplikací narazíme na problém, že každá, nejen jazyková kultura, používá různou měnu, různé oddělovače desetinných míst, případně i jiné pořadí některých znaků v abecedě. Při návrhu aplikace je na to třeba myslet a vytvářet ji takovým způsobem, aby v případě nutnosti převod do jiné kultury neznamenal vytvoření celé aplikace znovu.

Doporučuje se vytvářet kód aplikace a popis GUI odděleně. Pro popis uživatelského rozhraní je potřeba používat *zdrojů* (resources), tj. je třeba mít texty v samostatných souborech. Pro oddělení dalších dat, např. obrázků se doporučuje používat tzv. *content files*.

Další věcí, kterou je nutné řešit je konfigurace aplikace. Je chybou mít napsanou konfiguraci aplikace přímo v kódu. K tomu se v prostředí .NET používají konfigurační soubory, zde jsou uložené například možnosti připojení k DB.

CÍLE KAPITOLY

Seznamte se s následujícím:

- Obecný popis problematiky je uveden zde. Doporučení a techniky pro psaní globalizovatelných aplikací je zde.
- Popis použití *resource* a *content* souborů je k dispozici zde.
- Použití zdrojů v jazyku XAML je uvedeno zde.
- Popis jak lokalizovat za použití zdrojů naleznete zde.
- Přehled co a jak konfigurovat pomocí *App.Config* je v následujícím blogu. Podrobný popis, co a jak lze konfigurovat, je možné nalézt přímo v dokumentaci zde.

KLÍČOVÁ SLOVA

globalizace, lokalizace, konfigurace

ÚKOLY

1. Upravte libovolnou aplikaci z předešlých úkolů, tak aby používala zdroje a její uživatelské rozhraní bylo ve dvou jazycích.
2. Upravte aplikaci z kapitoly 5, tak aby používala konfigurační soubory (App.Config) pro nastavení databáze.

SHRNUTÍ

Po prostudování byste měli být schopni:

- Znát základní principy psaní lokalizovatelných aplikací.
- Vědět, kterým postupům se vyhnout při psaní aplikací.
- Znát a používat resources a konfigurační soubory.

OTÁZKY

1. Popište možné problémy při psaní lokalizovaných aplikací.
2. Popište, k čemu slouží konfigurační soubor aplikace. Uveďte **konkrétní** příklady.
3. Popište, k čemu slouží tzv. *resources*.

MÍSTO PRO VAŠE POZNÁMKY

9 Tvorba GUI aplikací v Pythonu – PyQt (Qt for Python), Tkinter

Pro psaní GUI aplikací v Pythonu existuje celá řada knihoven. Zde se zaměříme na dvě. První z nich je knihovna Tkinter, která se hodí spíše na tvorbu jednodušších uživatelských rozhraní. Druhá, mnohem komplexnější, je knihovna PyQt, což je wrapper pro C++ knihovnu Qt. Obě dvě knihovny jsou multiplatformní, což nám umožňuje psát přenositelné aplikace.

Základní principy jako události, komponenty a jejich vlastnosti, jsou podobné, co je odlišné, je jiný programovací jazyk (Python) a tím i použité výrazové prostředky. Odlišnosti jsou ve jménech konkrétních uživatelských prvků a jejich vlastnostech, dále v použití obsluh událostí. Příkladem buď např. v mechanismus událostí, který je v Qt implementován pomocí mechanismu *signálů* a *slotů*.

Další odlišností (zádrhelem) v ekosystému Qt je to, že v současné době existují dva podobné projekty Qt for Python (PySide2) zde a PyQt, které se liší v licencích a někdy i v některých přístupech. Tutoriály pro tvoření pomocí Qt v Pythonu jsou zde.

Poznamenejme, že detailní znalost daných knihoven není hlavní náplní. Berte to jako pohled, že existují i jiné přístupy pro tvorbu GUI než nám nabízí svět .NET.

CÍLE KAPITOLY

Cílem je získat základní přehled o použití daných knihoven.

- Detailní popis knihovny Tkinter je uveden zde. Pro naše účely se stačí podívat na zkrácený překlad zde. Prostudujte si zejména tvorbu ukázkové aplikace zde.
- Popis knihovny PyQt (verze 5) je k dispozici zde.
- Podívejte se na použití signálů a slotů zde.

KLÍČOVÁ SLOVA

Python, Qt, Tkinter

ÚKOLY

1. Pomocí knihovny Tkinter, nebo pomocí Qt implementujte jednoduchou kalkulačku.
2. Seznamte se s jazykem QML (jen velmi přehledově), použijte např. článek zde.

SHRNUTÍ

Po prostudování byste měli být schopni:

- Orientovat se v možnostech užití daných frameworků.
- Vytvářet velmi jednoduché GUI aplikace v Qt.
- V případě Qt umět použít (na elementární úrovni) QtDesigner.
- Posoudit podobnosti a rozdíly vůči WPF (.NET).

OTÁZKY

1. Co to je slot a signál?
2. Jaký je základní rozdíl mezi Tkinter a knihovnou Qt?

MÍSTO PRO VAŠE POZNÁMKY

10 Tvorba GUI aplikací v Javě – Swing a Java FX

Pro tvorbu GUI v Javě máme v zásadě dvě možnosti co použít. První z nich je starší knihovna Swing a druhou z nich je modernější a v současnosti preferovanější knihovna Java FX. Výhodou použití Javy je multiplatformnost aplikací.

Stejně, jako v případě Qt, je zcela mimo rozsah tohoto kurzu se věnovat detailům obou knihoven. Zaměřte se na tvorbu jednoduché ukázkové aplikace v každé z knihoven. Rozsah aplikace by měl být alespoň jedno okno a několik komponent, tlačítka, textboxy, menu a různé šoupáky (trackbary), či progressbary. Zjistíte, že spoustu věcí už umíte z prostředí .NET, akorát si musíte nastudovávat detaily chování daných komponent.

CÍLE KAPITOLY

V této kapitole se dozvíte (Zaměřte se alespoň na jednu z výše uvedených knihoven.):

- Tutoriál pro Swing je zde.
- Tutoriál pro Java FX je zde.

KLÍČOVÁ SLOVA

Java, Swing, Java FX

ÚKOLY

Naprogramujte jednu z výše uvedených aplikací, dle zvolené knihovny:

1. Napište ve Swing aplikaci pro řešení kvadratické rovnice. Vstupy jsou koeficienty a, b, c. Aplikace nechť určí řešení a vyšetří orientaci dané paraboly. Pokud chcete i nakreslit graf dané paraboly, můžete použít komponentu *http://www.jfree.org/jfreechart/*JFreeChart (bohužel již dále nevyvíjenou).
2. Protože Java FX má dobrou podporu multimédií pokuste se naprogramovat metronom. Funkcionalita je taková, že tempo si nastavíte pomocí slideru. Použijte třídy Timer (časovač) a TimerTask pro generování časových událostí.

SHRNUTÍ

Po prostudování byste měli být schopni:

- znát principy použití jedné z dominantních knihoven pro tvorbu GUI v Javě

OTÁZKY

1. Co je *JComponent*?
2. Co to je *Scene Graph*?
3. Co to je *Pulse*?
4. Co to je *FXML*?

OTÁZKY K ZAMYŠLENÍ

1. Můžete provést porovnání tvorby GUI v Javě s WPF?

MÍSTO PRO VAŠE POZNÁMKY

11 Tvorba GUI založeného na webu – Electron

Tvorba GUI založeného na webu se odlišuje od toho, co jsme dělali dosud. Zaměříme se na aktuální technologii Electron. Zde se tvorba GUI podobá psaní webové aplikace.

Architektura Electronu je taková, že backendová část je tvořena javascriptovým frameworkem Node.js a pro vykreslování je použito jádro prohlížeče Chromium. Toto zajišťuje v celku dobrou přenositelnost aplikací mezi platformami, na druhou stranu vytvořené aplikace zabírají dost systémových zdrojů, neboť se při startu aplikace startuje i engine Chromium.

Aplikace napsaná v Electronu používá dva druhy procesů, hlavní proces aplikace (*main process*) na kterém běží backendová část a který vytváří a spouští jednotlivá okna s GUI. Každé z těchto oken (Chromium engine) má vlastní překleslovací proces *renderer process*. Při updatování GUI na základě uživatelských akcí a dat musí tyto procesy spolu komunikovat.

Pro tvorbu desktopu dodává Electron své vlastní *Electron API* k již existujícímu *Node.js*. Z použitého frameworku Node.js samozřejmě vyplívá nutnost psát aplikace v JavaScriptu a je zde v podstatě nutnost psát události asynchronně.

Pro vlastní vytvoření GUI je použito Html a kaskádové styly (CSS).

Výhodou tohoto frameworku je poměrně silná komunita a dost existujících výukových materiálů.

CÍLE KAPITOLY

Nainstalujte si tutoriál podle pokynů uvedených zde

- Projděte si části zaměřené na vytváření oken a jednoduchých formulářových aplikací.

KLÍČOVÁ SLOVA

javascript, html, electron

ÚKOLY

1. Napište pomocí Electronu aplikaci pro řešení kvadratické rovnice. Vstupy jsou koeficienty a, b, c. Aplikace nechť určí řešení a vyšetří orientaci dané paraboly. Pokud chcete i nakreslit graf dané paraboly, můžete použít knihovnu *https://plotly.com/nodejs/*Plotly.

SHRNUTÍ

Po prostudování byste měli být schopni:

- znát principy frameworku Electron.
- být schopni vytvářet jednoduché aplikace v Electronu.

OTÁZKY

1. Co je *BrowserWindow*?
2. Jak je tvořeno GUI v Electronu?
3. Shrňte výhody a nevýhody použití webového vývoje desktopových aplikací?

MÍSTO PRO VAŠE POZNÁMKY

Literatura

[1] Windows Presentation Foundation Microsoft Docs. [online].
Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/framework/wpf/

[2] PyQt5 Reference Guide — PyQt v5.14.0 Reference Guide. [online].
Dostupné z: https://www.riverbankcomputing.com/static/Docs/PyQt5/

[3] Qt for Python [online].
Dostupné z: https://wiki.qt.io/Qt_for_Python

[4] JavaFX Documentation Home | JavaFX 2 Tutorials and Documentation. Moved [online].
Copyright © 2011, 2014 Oracle and [cit. 25.01.2020].
Dostupné z: https://docs.oracle.com/javafx/2/

[5] Creating a GUI With JFC/Swing (The Java™ Tutorials). [online]. Copyright © 1995, 2019 Oracle and [cit. 25.01.2020].
Dostupné z:https://docs.oracle.com/javase/tutorial/uiswing/

[6] Documentation | Electron. Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS. [online].
Dostupné z: https://www.electronjs.org/docs