Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Programování v .NET Framework verze 4.5

Jan Soukup

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Soukup Jan

Informatika

Název práce

Programování v .NET Framework verze 4.5

Anglický název

Development for .NET Framework 4.5

Cíle práce

Hlavním cílem bakalářské práce je popsat možnosti vývoj software s využitím nejnovějších verzí technologie .NET Framework a jejich porovnání s verzemi předchozími. Dalším cílem bude na ukázkových příkladech demonstrovat nové vlastnosti verze .NET 4.5.

Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků budou popsán vývoj platformy .NET se zaměřením na novinky, které přináší její verze 4.5. Dále budou na základě těchto poznatků zpracovány příklady demonstrující novou funkcionalitu verze 4.5.

Harmonogram zpracování

06/2013 - 12/2013 - studium a analýza zdrojů
12/2013 - kontrola průběhu práce (zápočet)
01/2014-03/2014 - tvorba příkladů
03/2014 - odevzdání práce + zápočet

Rozsah textové části
35-40 stran

Klíčová slova
C# , .Net Framework, Visual Studio, Windows, Objektové programování, msdn

Doporučené zdroje informací

JOHN SHARP, Microsoft Visual C# 2010: krok za krokem. Computer Press, 2010. 696, ISBN: 8025131475, 9788025131473

SHAWN R. MCLEAN, Instant .NET 4.5 Extension Methods How-to, Packt Publishing Ltd, 2013. 52, ISBN: 1849688575, 9781849688574

JOSEPH MAYO, C# 3.0 Unleashed: With the .NET Framework 3.5, Pearson Education, 2008. 1056, ISBN: 0768685427, 9780768685428

PATRICK SMACCHIA, Practical .NET2 and C#2, Paradoxal Press, 2006. 873, ISBN: 0976613220, 9780976613220

http://msdn.microsoft.com/cs-cz/library/

Vedoucí práce
Brožek Jiří, Ing., Ph.D.

Termín odevzdání
březen 2014

Ing. Martin Pelikán, Ph.D.
Vedoucí katedry

prof. Ing. Jan Hron, DrSc., dr. h. c.
Děkan fakulty

V Praze dne 12.9.2013

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Programování v .NET Framework verze 4.5" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.11.2014 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Jiřímu Brožkovi, Ph.D. za konzultace bakalářské práce.

Programování v .NET Framework verze 4.5

Development for .NET Framework 4.5

Souhrn

Tato práce obsahuje popis .NET Frameworku, popisuje jak je tvořen, jeho konstrukci, architekturu, dále tato práce popisuje jeho nejdůležitější vlastnosti a nejdůležitější funkce. V práci je dále rozebrán postupný vývoj .NET Frameworku, až po nejnovější verzi 4.5. Tento vývoj zahrnuje nejdůležitější vylepšení .NET Frameworku v jeho jednotlivých verzích.

Největší důraz se klade na verzi .NET Framework 4.5, kde nové, nejdůležitější vlastnosti jsou rozebrány více do hloubky. K práci je naprogramovaná aplikace, která ukazuje nové paralelní programování a asynchronní metody, v nové verzi .NET Framework 4.5.

Summary

This project includes description of .NET Framework, how is bulit, its construction, architecture, further this work describes its the most important features and functions. In this work, development of .NET Framework is described, from the beginning to the latest version of .NET Framework, which is 4.5. This development includes the most important features and improvements in every single version.

The greatest importance is targeting .NET Framework version 4.5, which new key features is disassembled deeper. The work includes aplication, which points out new way of parallel programming and asynchronous methods in .NET Framework 4.5.

Klíčová slova: C#, .NET Framework, Visual Studio, Windows, Objektové programování, msdn

Keywords: C#, .NET Framework, Visual Studio, Windows, Object oriented programming, msdn

Obsah

1. Úvod

.NET Framework prošel a neustále prochází dalším vývojem, vývojáři aplikací používají .NET Framework už přes dvanáct let. Za tu dobu si tento Framework oblíbil nejeden vývojář aplikací pro operační systémy Microsoft Windows a Microsoft Windows Server, pro které je výhradně určen. A není divu, vždyť Microsoft dělá vše proto, aby byl Framework stále lepší, a stále více usnadňoval vývojářům práci.

.NET Framework je velice rozsáhlý, a jeho prvky zasahují takřka do všech odvětví vývoje aplikací. Na vývoj webových aplikací s soustředí hlavně část Frameworku zvaná ASP.NET, na práci s daty, se zaměřuje část zvaná ADO.NET, další část Windows Forms, se pak zaobírá vytvářením aplikací v grafickém programovacím rozhraní, které zpřístupňuje základní elementy rozhraní Microsoft Windows.

.NET Framework jde použít v kombinaci s více programovacími jazyky a Microsoft se úspěšně snaží udržovat dokumentaci, na vysoké úrovni.

Díky těmto možnostem .NET Frameworku, lze při dobré orientaci v jeho dokumentaci, vytvářet v celku snadno, i složitější aplikace. Nicméně někdy nám nezbyde, než si třídy a metody vytvořit vlastní.

2. Cíl práce a metodika

2.1. Cíl práce

Hlavním cílem práce je popsat možnosti vývoje software s využitím nejnovějších verzí technologie .NET Framework a jejich porovnání s verzemi předchozími. Dalším cílem bude na ukázkových příkladech demonstrovat nové vlastnosti verze 4.5.

2.2. Metodika

Metodika práce je založena na studiu a analýze odborných informačních zdrojů. Na základě syntézy zjištěných poznatků bude popsán vývoj platformy .NET se zaměřením na novinky, které přináší verze 4.5. Dále budou na základě těchto poznatků zpracovány příklady demonstrující novou funkcionalitu verze 4.5.

3. .NET Framework

.NET Framework je softwarový Framework vytvořený společností Microsoft a běží zejména na operačním systému Microsoft Windows. .NET Framework obsahuje obsáhlé knihovny a umožňuje komunikaci mezi hrstkou programovacích jazyků. Programy psané s pomocí .NET Framework, které se vykonávají v softwarovém prostředí jsou označovány jako CLR (Common Language Runtime), aplikační virtuální stroj (Virtual Machine), zprostředkovává služby jako je bezpečnost, správa paměti a řízení výjimek.
Knihovna tříd a Common Language Runtime představují dohromady .NET Framework.

Základní knihovna tříd .NET Frameworku poskytuje uživatelské rozhraní, přístup k datům, připojení do databáze, kódování, vývoj webových aplikací, číselné algoritmy a internetovou komunikaci. Programátoři pak vytvářejí svůj software tím, že kombinují jejich vlastní zdrojový kód s .NET Frameworkem a ostatními knihovnami. .NET Framework je zamyšlený tak, aby ho používalo co nejvíce nových aplikací, vytvořených pro operační systém Windows. Microsoft vytvořil také vývojové prostředí pro .NET, které se nazývá Visual Studio.[1]

3.1. Historie

Microsoft začal s vývojem .NET Frameworku na konci 2. tisíciletí, pod původním jménem Next Generation Windows Services (NGWS).

Verze .NET 3.0 je obsažena ve Windows Server 2008 a Windows Vista. Verze 3.5 je obsažena ve Vindows 7 a Windows Server 2008 R2 a může být nainstalována na Windows XP a Windows Server 2003. 12. dubna 2010 byla vydána verze .NET Frameworku 4.0 společně s vývojovým prostředím Visual Studio 2010.

Rodina .NET Frameworku dále obsahuje dvě verze pro mobilní a zabudovaný systém. Jsou to redukované verze frameworku, .NET Compact Framework, který je dostupný na zařízeních s operačním systémem Windows CE (chytré mobilní telefony) a .NET Micro Framework, ten je určený pro nízko-účelová zařízení.

tabulka vývoje .NET Framework

Generace	Datum vydání	Vývojové prostředí	Distribuováno s OS
1.0	13.2.2002	Visual Studio .NET	
1.1	24.4.2003	Visual Studio .NET 2003	Windows Server 2003
2.0	7.11.2005	Visual Studio 2005	Windows Server 2003 R2
3.0	6.11.2006	Expression Blend	Windows Vista, Windows Server 2008
3.5	19.11.2007	Visual Studio 2008	Windows 7, Windows Server 2008 R2
4.0	12.4.2010	Visual Studio 2010	
4.5	15.8.2012	Visual Studio 2012	Windows 8, Windows Server 2012

[2]

3.2. Vlastnosti konstrukce .NET Frameworku

3.2.1. Interoperabilita

Jelikož operační systém normálně vyžaduje spolupráci novějších aplikací se staršími, .NET Framework zajišťuje prostředek pro přístup k funkcionalitě, realizované v novějších a starších programech, který se provádí mimo prostředí .NET. Přístup k COM (Component Object Model) komponentům zajišťují jmenné prostory frameworku System.Runtime.InteropServices a System.EnterpriseServices. Přístup k dalším funkcionalitám se provádí pomocí vlastnosti P/Invoke.

3.2.2. CLR (Common Language Runtime)

Common Language Runtime slouží jako prováděcí stroj .NET frameworku. Všechny .NET programy prováděné pod kontrolou CLR, zaručují určité vlastnosti a chování v oblastech správy paměti, bezpečnosti a řízení výjimek.

3.2.3. Jazyková nezávislost

.NET Framework představuje CTS (Common Type System). Tato specifikace definuje všechny možné datové typy a programovací jazyky, podporované CLR a dále jak budou či

nebudou komunikovat mezi sebou, v souladu se specifikací CLI (Common Language Infrastructure). Díky tomuto prvku, .NET Framework podporuje výměnu typů a instancí objektu mezi knihovnami a aplikacemi napsanými použitím jakéhokoliv programovacího jazyku podporovaného .NET frameworkem.

3.2.4. BCL (Base Class Library)

Base Class Library, jakožto součást FCL (Framework Class Library), je knihovna funkcionalit, které jsou dostupné pro všechny programovací jazyky používající .NET Framework. BCL poskytuje třídy, které zapouzdřují celou řadu běžných funkcí, včetně čtení a zápisu do souboru, grafického provedení, komunikace s databází, manipulace s XML dokumentem a tak dále. Skládá se z tříd a rozhraní opakovatelně použitých typů integrovaných v CLR. [3]

3.2.5. Zjednodušené nasazení

.NET Framework obsahuje nástroje, které pomáhají řídit instalaci počítačového softwaru tak, aby se ujistily, že nebude zasahovat do předem nainstalovaného softwaru a že vyhovuje bezpečnostním požadavkům.

3.2.6. Bezpečnost

Konstrukce projevuje nějaká zranitelná místa, jako například přetečení vyrovnávací paměti, což bylo využíváno škodlivým softwarem. Kromě toho .NET poskytuje běžný bezpečnostní model pro všechny aplikace.

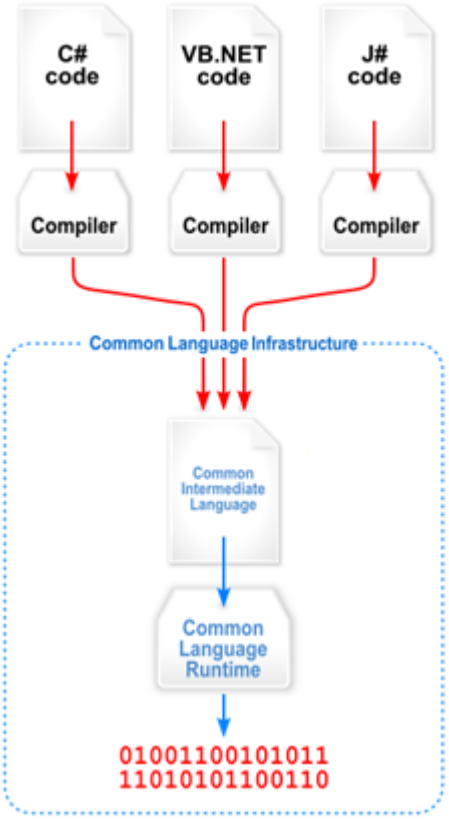
3.2.7. Přenosnost

Microsoft nikdy neimplementoval celý Framework na žádný operační systém vyjma Microsoft Windows. Více platformové implementace jsou k dispozici pro různé operační systémy. Microsoft předložil specifikace pro CLR, jazyk C#, a jazyk C++/CLI organizacím ISO a ECMA, čímž je zpřístupnil jako oficiální standardy. Díky tomu je umožněno třetím stranám vytvářet kompatibilní implementace frameworku na jiné platformy.[4]

3.3. Architektura .NET Frameworku

3.3.1. CLI (Common Language Infrastructure)

Účel Common Language Infrastructure je poskytovat jazykově neutrální platformu pro vývoj aplikací a jejich provádění, zahrnující funkce pro řízení výjimek, sběr smetí (garbage collection), bezpečnost a interoperabilitu. Implementováním hlavních aspektů .NET Framework v rámci CLI, tyto funkcionality nebudou vázány na jeden programovací jazyk, ale budou dostupné napříč všemi jazyky, které .NET podporuje. Implementace CLI od Microsoftu se nazývá CLR (Common Language Runtime).[5]



Obr. č.1. – CLI infrastruktura

3.3.2. CLR (Common Language Runtime)

Česky volně přeloženo jako běhové prostředí. CLR se stará o správu paměti, provádění vláken, provádění kódu, prověřování bezpečnosti kódu, kompilaci a další systémové služby. Tyto služby jsou vlastní pro spravovaný kód, jenž běží na CLR.

Vzhledem k bezpečnosti, spravované komponenty jsou ohodnocovány různými stupni důvěry. Tyto stupně závisí na řadě faktorů, které obsahují jejich původ. To znamená, že spravovaný komponent může a nemusí být způsobilý pro souborové operace, nebo jiné citlivé funkce.

CLR uplatňuje tzv. bezpečnost přístupu do kódu. Například uživatelé si mohou být jisti, že spustitelný soubor, vložený na webovou stránku může spustit třeba animaci, nebo zahrát písničku, ovšem nemá přístup k jejich osobním datům, síti nebo do systémových souborů. Bezpečnostní vlastnosti CLR tak umožňují legitimnímu internetovému softwaru být nabitý spoustou funkcí.

CLR dále vynucuje robustnost kódu implementováním striktního psaní a ověřování kódu. Tato infrastruktura se nazývá CTS (Common Type System). CTS zajišťuje, že všechen spravovaný kód je sebe-popisující. Různé kompilátory, ať už od Microsoftu nebo třetích stran, generují kód, který vyhovuje CTS. To znamená, že takový spravovaný kód může využívat další spravované typy a instance. A při tom striktně splňovat typy přesnosti a bezpečnosti.

Běhové prostředí, navíc eliminuje spoustu běžných softwarových problémů. Například automaticky zpracovává rozložení objektů, zajišťuje odkazy na jednotlivé objekty, a pokud už tyto objekty nejsou používané, maže je. Tato automatická správa paměti řeší dva nejčastější problémy aplikací, a to prosakování paměti a neplatné odkazy na paměť.

CLR také zvyšuje produktivitu vývojáře. Například, programátoři mohou psát aplikace v programovacím jazyce, který je podporován frameworkem, který si samy zvolí, a přesto mohou využívat všech výhod CLR, knihovny tříd, a komponentů psaných v jiných jazycích jinými vývojáři. Jazykové kompilátory zaměřené na .NET Framework umožňují dostupnost vlastností .NET Frameworku existujícímu kódu, psanému v tomto jazyce, což výrazně usnadňuje proces migrace pro existující aplikace.

Zatímco běhové prostředí je designováno pro software budoucnosti, podporuje i software současný a starší. Přenosnost mezi spravovaným a nespravovaným kódem umožňuje vývojářům v pokračování při používání COM komponentů a dalších knihoven.

Prostředí CLR je navrženo na zlepšení výkonu, třebaže obvyklé jazykové prostředí nabízí mnoho standartních služeb, spravovaný kód není nikdy interpretován. Vlastnost zvaná JIT (Just In Time) kompilování, umožňuje veškerému spravovanému kódu běžet v nativním strojovém jazyce systému, na kterém je vykonáván. Mezitím správa paměti odstraňuje možnosti fragmentované paměti a zvětšuje paměť takzvané lokality odkazu k dalšímu zlepšení výkonu.

Prostředí může být umístěno na velmi výkonných serverově založených aplikacích, jako například Microsoft SQL Server a IIS (Internet Information Services). Tato infrastruktura nám umožňuje používat spravovaný kód pro psaní vlastní logiky, a při tom stále využívat skvělých výkonů nejlepších podnikových serverů, které podporují umístění prostředí.[6]

3.3.3. Bezpečnost

.NET Framework disponuje vlastním bezpečnostním systémem, který obsahuje dvě hlavní složky. CAS (Code Access Security), validaci a verifikaci. Code Access Security je založena na evidenci, která je asociovaná se specifickým shromážděním (assembly). Standardně je evidence zdrojem tohoto shromáždění (ať už je nainstalován na lokálním počítači nebo byl stažen z internetu nebo intranetu). CAS používá evidenci k určení přidělených práv kódu.

3.3.4. Knihovna tříd

.NET Framework obsahuje sadu knihoven standardních tříd. Knihovna tříd je organizovaná v hierarchii jmenných prostorů. Většina vestavěných API je součástí jmenných prostorů buď System nebo Microsoft. Tyto knihovny tříd implementují velké množství běžných funkcí. Knihovny .NET jsou dostupné pro všechny CLI programovací jazyky. Knihovny tříd .NET Framework jsou rozděleny do dvou skupin: Base Class Library a Framework Class Library.

BCL (Base Class Library) obsahuje malou podmnožinu celé knihovny tříd a je jádrem kolekce tříd, která slouží základní API Common Language Runtime. Třídy v mscorlib.dll a některé třídy v System.dll a System.core.dll jsou považovány za součást BCL. Třídy BCL jsou dostupné v .NET Framework, ale i v alternativních implementacích Frameworku .NET Compact Framework, Microsoft Silverlight a Mono.

FCL (Framework Class Library) zahrnuje třídy BCL a odkazuje na všechny třídy celé knihovny, kterou má .NET Framework k dispozici. Zahrnuje rozšířené kolekce knihoven, Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation společně s dalšími.

3.3.5. Správa paměti

CLR frameworku .NET, osvobozuje vývojáře od správy paměti (alokace paměti a její následné uvolnění po dokončení). CLR řídí paměť automaticky tak, že detekuje, kdy může být paměť bezpečně uvolněna. Instance objektů .NET jsou alokovány z řízené hromady, což je část paměti spravovaná CLR. Tak dlouho, dokud existuje odkaz na objekt, ať už je to přímý odkaz na objekt, nebo odkaz nepřímý, je objekt považován za používaný. Když neexistuje žádný odkaz na objekt, tím pádem nemůže být používaný, stává se z něj odpad, způsobilý pro sběr. .NET Framework obsahuje sběrače odpadu, který pracuje periodicky, na odděleném vláknu od vlákna aplikace. Ten vyčítá všechny objekty, které už není možno využít a odebírá jim přidělenou paměť.

.NET sběrač odpadu, neboli GC (Garbage Collector) je nedeterministický, pracující na principu označení a odstranění. Garbage Collector pracuje pouze, pokud je zabraná určitá část paměti, nebo pokud je na paměť vyvíjen určitý nátlak v systému. Protože není zaručeno, kdy jsou podmínky pro zpětné získání paměti naplněny, GC pracuje nezávisle. Každá .NET aplikace má sadu kořenů, které fungují jako pointery na objekty, na řízené hromadě. Ty obsahují odkazy na statické objekty, definované jako lokální proměnné nebo parametry metody současného rámce, tak jako objekty, na které odkazují registry CPU. Když běží garbage collector, aplikace je pozastavena a pro každý objekt odkazovaný v kořenu, GC rekurzivně vyčte všechny objekty dosažitelné z kořenových objektů a označí je za dosažitelné. Při tom používá CLI metadata a zrcadlení, aby objevil objekty zapouzdřené objektem, a pak je rekurzivně projde. Poté vyčte všechny objekty na hromadě pomocí zrcadlení. Všechny objekty neoznačené jako dosažitelné jsou odpad. Tato fáze se nazývá označovací fáze. Poněvadž paměť zadržovaná odpadem nemá žádný důsledek, je považovaná za volné místo. Nicméně, toho zanechává kusy volných míst mezi objekty, které spolu původně sousedily. Proto jsou objekty poté opět zhuštěny k sobě, aby se paměť stala znovu styčnou. Jakýkoliv odkaz na objekt, který se stal nedosažitelným

kvůli pohybu objektu při zhušťování paměti, je automaticky aktualizován GC tak, aby odrážel svou novou polohu. Po skončení činnosti GC je opět obnovena činnost aplikace.

Garbage collector používaný .NET Frameworkem je také generační. Nově vytvořené objekty jsou přisuzovány generaci 0. Objekty, které přežily činnost GC, jsou označovány jako generace 1. Generace 2 jsou objekty generace 1, které přežily další činnost GC. .NET Framework používá nejvýš objekty druhé generace. Objekty vyšších generací jsou sbírány sběračem odpadu méně často, než objekty generací nižších. Tohle pomáhá zvýšit účinnost sběru odpadu, jelikož starší objekty vydrží většinou déle než objekty novější. Tudíž díky eliminaci starších objektů z prostoru, který sběrač prohledává při svém chodu, má za úkol zkontrolovat méně objektů a tím pádem jich musí být i zhuštěno méně.[7]

4. .NET Framework 1.0

Verze 1.0 byla prvním vydáním .NET Frameworku. Vydána byla v únoru 2002 a je dostupná pro operační systémy Microsoft Windows 98, ME, NT 4.0, 2000 a XP. Standartní podpora od Microsoftu pro verzi 1.0 skončila 10. Července 2007, prodloužená podpora pak 14. Července 2009.

Verze .NET Framework 1.0 obsahuje CLR verze 1.0 a také první verzi BCL. Používané programovací jazyky jsou C# verze 1.0, Visual Basic .NET verze 7.0. Vývojové prostředí pro tuto verzi frameworku 1.0 je Visual Studio .NET.[2]

5. .NET Framework 1.1

První velké vylepšení .NET Frameworku, verze 1.1 byla vydána v dubnu 2003 a jako první verze byla dodávána a operačním systémem Microsoft Windows Server 2003. Přišlo s ním i nové vývojové prostředí Visual Studio .NET 2003 a jazyk C# byl vylepšen na verzi 1.2. Byla také vylepšena verze CLR na 1.1. Standartní podpora skončila 14. Října 2008, prodloužená pak 8. Října 2013, kvůli implementaci do Windows Server 2003.

Verze 1.1 rozšiřuje předchozí verzi .NET frameworku o nové vlastnosti, vylepšuje některé stávající vlastnosti a také dokumentaci.

5.1. ASP.NET Mobile Controls (mobilní ovládací prvky)

Původně Microsoft Mobile Internet Toolkit, rozšiřuje .NET Framework a Visual Studio .NET, poskytováním podpory pro mobilní zařízení, jako jsou mobilní telefony a PDA.

Protože jsou mobilní ovládací prvky součástí .NET Frameworku, terminologie se změnila tak, aby zachycovala konvence používané v dokumentaci .NET a dokumentace mobilních ovládacích prvků se spojila do širší dokumentace sady .NET Framework.

ASP.NET Mobile Controls, rozšiřuje ASP.NET serverové ovládací prvky tak, že serverové ovládací prvky se přizpůsobují mobilnímu zařízení, na kterém se načítá webová aplikace. Detekce pomocí prohlížeče zajistí, že se mobilní ovládací prvky přizpůsobí schopnostem jednotlivých mobilních zařízení, ať už je to plnohodnotné PDA, nebo pěti řádkový displej v mobilním telefonu. Toto adaptivní načítání umožňuje načítání na mnoha nepříjemných, specifických mobilních přístrojích, které díky tomu nemusí vývojář řešit a může se tak plně soustředit na logiku aplikace.

5.2. Změny v ADO.NET

Ve verzi .NET Frameworku 1.1, byl přidán jmenný prostor System.Data.Odbc, který se pro verzi 1.0 musel stahovat jako doplněk.

Dále pak byl přidán jmenný prostor System.Data.OracleClient.

Objekt DataReader nyní obsahuje vlastnost HasRows, která slouží k určení, zda byly vypsány řádky, aniž byla volána metoda Read.

5.3. Provádění bok po boku (Side-by-Side Execution)

Provádění bok po boku je schopnost skladovat a provádět víc verzí aplikace nebo komponenty na jednom a tom samém počítači. To znamená, že vývojáři mohou mít na víc verzí prostředí a víc verzí aplikací nebo komponentů, které využívají verzi prostředí na jednom počítači ve stejný čas. Další instalace .NET Frameworku nepůsobí na již nainstalovanou.

5.4. Změny v bezpečnosti .NET Framework

Ve verzích 1.0 a 1.1, aplikace, které obdrží menší než plnou důvěru od systému pro bezpečnost přístupu do kódu, nemohou volat sdílené spravované knihovny, pokud jim to specificky neumožní autor knihovny. To se provádí pomocí atributu AllowPartiallyTrustedCallersAttribute. Pokud vývojář plánuje používání knihoven pro částečně důvěrný kód, musí si být vědom toho, že ne všechny knihovny mu budou k dispozici.

Základní bezpečnostní politika byla změněna tak, že aplikace spouštěné z internetové zóny (Internet Zone) a aplikace přidělené této kódové skupině, nyní přijímají povolení asociované se sadou internetových povolení. Ve výsledku, aplikace z internetu nyní přijímají dostatečná povolení pro provedení.

5.5. ASP.NET bezpečnost v hostovaných prostředích

ASP.NET podporuje částečnou důvěru ve webově založených aplikacích. Nabízí lepší bezpečnost pro víc aplikací, které hostí jeden webový server. Ačkoli účet operačního systému, pod kterým aplikace běží, ukládá bezpečnostní omezení aplikaci, systém pro bezpečnost přístupu do kódu, který používá CLR, může pomoci prosadit další restrikce na vybrané zdroje aplikace, založené na politice, kterou si určí sám vývojář. Tato vlastnost se dá využít ve sdíleném serverovém prostředí, pro izolaci oddělených aplikací a na samostatných severech, kde vývojář požaduje, aby aplikace běhaly pouze s minimem nezbytných privilegií.

V ASP.NET se to provádí stejně jako v celém Frameworku, pomocí atributu AllowPartiallyTrustedCallersAttribute. Některé knihovny tento atribut nemají, a tudíž se nedají pro částečně důvěryhodné aplikace použít.

5.6. Podpora IPv6

Už od verze 1.1 podporuje .NET Framework vznikající vylepšení IP.[8]

6. .NET Framework 2.0

Verze 2.0 vyšla v listopadu 2005, společně s novou verzí jazyka C# 2.0 a jazyka Visual Basic .NET 8.0. K .NET Frameworku 2.0 vylo vydáno nové vývojové prostředí Visual Studion 2005. Verze CLR byla vylepšena na 2.0. .NET Framework 2.0 je dodáván s operačním systémem Microsoft Windows Server 2003 R2.

6.1. ADO.NET

Nové vlastnosti v ADO.NET obsahují podporu pro UDT (User-Defined Types, typy definované uživatelem), asynchronní operace databáze, XML datové typy, typy rozsáhlých hodnot a nových atributů, které povolují aplikaci podporovat víc aktivních výsledných sad (MARS), spolu s SQL Serverem 2005.

6.2. ASP.NET

Microsoft .NET Framework 2.0 obsahuje velké vylepšení pro všechny oblasti ASP.NET. Nové ovládací prvky, pro vývoj dynamických webových stránek, zjednodušují přidávání běžně používaných funkcionalit. Nové ovládací prvky pro data, umožňují zobrazit a upravovat data na webové stránce ASP.NET, bez nutnosti psaní kódu. Vylepšený kód na pozadí modelu činí vyvíjené stránky ASP.NET jednodušší a víc robustní. Nové vlastnosti cache paměti poskytují pár nových způsobů, jak cachovat stránky, včetně možnosti vytvoření cache na základě tabulek v databázi SQL serveru.

Nyní může vývojář upravovat webové stránky mnoha různými způsoby. Profilové vlastnosti umožňují ASP.NET automaticky sledovat hodnoty vlastností, individuálně pro uživatele. Používáním webových prvků, může vývojář stvořit stránky, které si uživatelé mohou upravit v prohlížeči. Další novou vlastností ASP.NET je jednoduché přidávání menu, pomocí ovládacích prvků.

Vylepšení vlastností pro tvorbu webových stránek vede k rychlejšímu a lehčímu vytváření profesionálního webu. Vzorové stránky (Master pages) umožňují vytvoření konzistentního rozložení pro všechny stránky webu a náměty (Themes) pak umožňují definování konzistentního vzhledu pro ovládací prvky a statický text. Kvůli ochraně webových stránek, si vývojář může před-kompilovat web k výrobě spustitelného kódu ze zdrojových

souborů. Vývojář pak může nasadit výsledný výstup, který nezahrnuje žádné informace o zdroji, které nejsou určeny pro produkční server. Vylepšení ASP.NET také zahrnuje nové nástroje a třídy pro usnadnění spravování webu.

6.3. Autentizované toky dat

Aplikace mohou použít nové třídy NegotiateStream a SslStream pro autentizaci a zabezpečení informací mezi klientem a serverem. Tyto třídy podporují vzájemnou autentizaci a kódování dat. Třída NegotiateStream používá bezpečnostní protokol Negotiate, třída SslStream protokol SSL pro autentizaci.

6.4. COM Interop, vylepšení služeb

Operační systém udržuje omezené množství stavů (handles), které jsou používány jako odkazy na kritické zdroje operačního systému. Nové třídy SafeHandle, CriticalHandle a jejich specifické podtřídy, zajišťují bezpečný a spolehlivý způsob manipulace se stavy (handles) operačního systému.

Díky zařazování nových vylepšení je jednodušší interoperating s přirozeným kódem. Dvě hlavní vylepšení v tomto směru. Vlastnost zabalit pointery přirozené funkce do delegátů a vlastnost k zařazování řetězců struktur, které mají pevnou velikost, uvnitř struktur.

Nové switche Tlbimp.exe (Type Library Improter) a Tlbepx.exe (Type Library Exporter) eliminují závislosti na registru při řešení typu referencí na knihovnu.

6.5. Detekce změn v síťové konektivitě

Třída NetworkChange umožňuje aplikacím dostávat notifikace, když se změní IP adresa síťového rozhraní. To se stává většinou, když dojde odpojení a dalšímu připojení k síti.

6.6. Distribuovaný computing

Podpora pro požadavky FTP klienta byla přidána do jmenného prostoru System.Net, cachuje zdroje http, automatické objevení proxy, získávání síťového provozu a statistické informace. Jmenný prostor dále obsahuje třídu HttpListener, kterou může vývojář použít na vytvoření jednoduchého webového serveru, který bude odpovídat na http požadavky.

Do jmenných prostorů System.Web.Services byla přidána podpora pro SOAP 1.2 a nulovatelné elementy.

6.7. Doplňky třídy Console

Nové doplňky třídy Console umožňují aplikacím manipulovat s rozměry konzolového okna a screen bufferu. Přesunutí obdelníkové oblasti screen bufferu, což je užitečné pro hladké provádění jednoduchých animací. Další nové třídy obsahují ovládání barev pozadí a textu, viditelnost a velikost kurzoru myši, frekvenci a délku konzolového pípnutí.

6.8. DPAPI (Data Protection API)

Nová DPAPI obsahuje čtyři metody, které umožňují aplikacím šifrovat hesla, klíče atd., a to bez volání invoke platformy.

6.9. Globalizace

Verze 2.0 s sebou přináší pár nových vlastností pro vývoj aplikací v jiných jazycích. Například je možné definovat důležité informace spojené s kulturou aplikace.

6.10. Ping

Třída ping umožňuje aplikacím zjistit, jestli je vzdálený počítač dostupný pro síťové spojení.

6.11. Podpora 64-bitových platforem

Nová generace 64-bitových počítačů umožňuje vytváření aplikací, které jsou rychlejší a využívají výhod vyšší paměti, oproti 32-bitovým aplikacím. Nová podpora 64-bitových aplikací umožňuje uživatelům udělat spravované knihovny kódu, nebo jednoduše využívat nespravovaných knihoven kódu na 64-bitových počítačích.

6.12. Podpora ACL (Acces Control List)

ACL se používá pro přidělení nebo anulování práv, pro přístup ke zdrojům na počítači. Nové třídy, které obsahují spravovaný kód pro vytvoření a modifikaci ACL, byly přidány do .NET Frameworku 2.0. Nový členové, kteří používají ACL, byli přidáni do I/O, do registrů a do vláknových tříd (System.Threading).

6.13. Podpora FTP

Aplikace mohou využívat přístupu na FTP pomocí tříd: WebRequest, WebResponse a WebClient.

6.14. Podpora funkce debuggeru Edit and Continue

Ve verzi .NET 2.0, při debuggování ve Visual Studiu přibyla možnost měnit kód, při provádění v Break módu. Po provedení a aplikování změn kódu může pokračovat provádění kód a uživatel vidí dopady.

6.15. Transakce

Nový jmenný prostor System.Transactions obsahuje třídy, které umožňují aplikaci podílet se na transakcích řízených MSDTC (Microsoft Distirbuted Transaction Coordinator) nebo lokálním transakčním správcem.

6.16. Vlastnosti pro práci s XML

.NET Framework 2.0 přichází s mnoha vylepšeními jmenného prostoru System.Xml. Obsahuje nový XSLT procesor. Nové vlastnosti v třídách XmlReader, XmlWriter a XPathNavigator, nově pak editaci možností v XPathNavigator. Navíc je nový model pro vytváření objektů XmlReader a XmlWriter.

6.17. Vylepšení protokolu událostí (EventLog)

Nyní může uživatel využít knihovny pro zprávy, parametry a kategorie v EventLogu.

6.18. Získávání informací o síťové konfiguraci lokálního PC

Díky jmennému prostoru System.Net.NetworkInformation mají aplikace přístup ke statistikám síťového vytížení, mohou zjistit protokoly IP, IPv4, IPv6, TCP a UDP.[9]

7. .NET Framework 3.0

.NET Framework 3.0 byl vydán 21. Listopadu 2006 a používá stejnou verzi CLR jako předchozí verze. Ve verzi 3.0 nedošlo k zásadním změnám v architektuře Frameworku, tato verze se zaměřuje na čtyři nové technologie Frameworku a SDK. Tyto technologie jsou, WCF (Windows Communication Foundation), WPF (Windows Presentation Foundation), WF (Windows Workflow Foundation) a Windows CardSpace.

7.1. Windows Communication Foundation

WCF je sada API pro vytváření serverově orientovaných aplikací.

7.1.1. Architektura

Nástroj WCF je obvykle používán na implementaci a rozmístění SOA (Server-Oriented Architecture). Je navržen na používání zásad SOA na podporu vzdáleného computingu, kde služby mají vzdálené uživatele. Služby mohou být využívány více klienty. Služby jsou volně vázané jedna na druhou a většinou obsahují rozhraní WSDL (Web Services Description Language), které může využít jakýkoliv WCF klient při používání služby, nezávazně na typu platformy, na které je služba hostována.

7.1.2. Koncové body (Endpoints)

WCF klient se připojuje na služby WCF přes endpoint. Každá služba odhaluje svůj kontrakt přes jeden nebo více endpointů, ty obsahují specifickou URL adresu, určuje, kde může být endpoint dostupný, a vlastnosti vazby, které určují, jak budou přenášeny data.

Vazby specifikují, jaké komunikační protokoly budou použity pro přístup ke službě, jestli bude použit bezpečnostní mechanismus a tak dále. WCF obsahuje předdefinované vazby pro nejpoužívanější komunikační protokoly, jako jsou SOAP přes HTTP, přes TCP a přes Message Queues. Interakce mezi klientem a endpointem WCF je udělaná pomocí SOAP envelope, což jsou jednoduché XML formy, to je činí platformově nezávislými. Pokud chce klient přístup na službu přes endpoint, musí mít se serverem kompatibilní endpoint.

7.1.3. Behaviors

Behaviors jsou typy, které modifikují nebo rozšiřují funkcionalitu služby nebo klienta. To umožňuje vývojáři vytvářet vlastní zpracování, transformace, nebo inspekce zpráv, které jsou odesílány nebo přijímány.

7.2. Windows Presentation Foundation

WPF je grafický subsystém pro vykreslování uživatelského rozhraní v aplikacích založených na Windows. WPF používá DirectX, pokouší se poskytnout konzistentní programovací model pro vytváření aplikací a odděluje uživatelské rozhraní od logiky aplikace. WPF

aplikace mohou být vydávány jako samostatné počítačové programy, nebo mohou být hostovány na serverech.

7.2.1. Vazba dat

WPF má zabudovanou sadu datových služeb, které umožňují vývojářům aplikace, vázat data a manipulovat s nimi v aplikacích, třemi možnými způsoby: jedenkrát (klient ignoruje updaty na serveru), jedním směrem (klient má přístup k datům pouze pro čtení), a dvěma směry (klient může číst data a zapisovat do dat na serveru).

7.2.2. Služby médií

WPF poskytuje integrovaný systém pro budování uživatelského rozhraní s běžnými médii, jako jsou obrázky, zvuky a videa. Možnosti 3D ve WPF jsou podmnožinou Direct3D. Nicméně WPF poskytuje těsnější integraci s ostatními vlastnostmi jako uživatelské rozhraní, dokumenty a media. Toto umožňuje 3D uživatelské rozhraní, 3D dokumenty a 3D media.

7.2.3. Šablony

WPF umožňuje přímo definovat vzhled elementu přes jeho vlastnosti, nebo nepřímo přes šablonu nebo styl. Styl je kombinací nastavení vlastností, které mohou být aplikovány na element uživatelského rozhraní pomocí jednoho atributu. Šablona je mechanizmus pro definování alternativních uživatelských rozhraní pro části WPF aplikace.

7.2.4. Animace

WPF podporuje časově založené (time-based) animace v kontrastu k rámcově založenému přístupu (frame-based). To odděluje rychlost animace od funkčnosti systému. WPF podporuje animace nízké úrovně přes časovače (timers) a animace vyšší úrovně přes třídy Animation.

7.2.5. Efekty

WPF 3.0 poskytuje bitmapové efekty (třída BitmapEffect), což jsou rastrové efekty aplikované do Visual Studia. Tyto rastrové efekty jsou napsány v nespravovaném kódu a vynucují vykreslování Visual Studia na CPU a ne na GPU.

7.3. Windows Workflow Foundation

WF je technologie od Microsoftu, která poskytuje API, in-process workflow engine a znovu hostovatelného konstruktéra pro implementaci dlouho běžících procesů jako pracovní postupy (workflows) v .NET aplikacích.

Workflow je série odlišných programovacích kroků nebo fází. Každý krok, je navrhován ve WF jako aktivita (Activity). .NET Framework poskytuje knihovnu aktivit. Vlastní aktivity mohou být také vyvíjeny pro přídavnou funkcionalitu.

Zapouzdření programovacích funkcionalit do aktivit, umožňuje vývojáři vytvořit lépe spravovatelnou aplikaci. Každý komponent provádění může být vyvíjen jako objekt CLR, jehož provádění bude spravováno Workflow runtimem.

7.3.1. Workflow Engine

Workflow engine poskytuje následující vlastnosti.

- Plánování a provádění pracovních procesů a aktivit.
- Správa provozu provádění mezi aktivitami.
- Přetrvávající pracovní procesy.
- Správa dat pro prováděné aktivity.
- Vestavěný poskytovatel sledování, který zaznamenává vestavěné události Workflow.
- Poskytuje rozšiřitelnost v podobě Workflow Extensions.

7.4. Windows CardSpace

Nyní už zrušený klient od Microsoftu pro Identity Metasystem. CardSpace je instance třídy softwaru klienta identity nazývaného Identity Selector. CardSpace ukládal odkazy na uživatelovy digitální identity, tyto identity jim pak prezentoval jako informační karty. CardSpace poskytuje konzistentní uživatelské rozhraní navržené na pomoc lidem jednoduše a bezpečně používat tyto identity v aplikacích a webových sítích, které je akceptují.

V únoru 2011, Microsoft vydal prohlášení, že Windows CardSpace 2.0 nebude vydána, a nyní pracuje na náhradě zvané U-Prove.[10]

8. .NET Framework 3.5

Verze 3.5 byla vydána 19. Listopadu 2007, stejně jako předchozí verze používá verzi CLR 2.0, což je stejné i jako .NET Framework 2.0. Při instalaci, .NET Framework 3.5 instaluje dále i .NET Framework 2.0 Service Pack 1 a .NET Framework 3.0 SP1, což dodává některé nové metody a vlastnosti do tříd BCL ve verzi 2.0, potřebných pro funkce verze 3.5, jako například LINQ. Tyto změny nijak neovlivňují aplikace psané pro verzi frameworku 2.0.

S .NET Frameworkem 3.5 přislo také vylepšení verzí programovacích jazyků C# na verzi 3.0 a Visual Basic .NET na verzi 9.0. Dále přišlo i nové vývojové prostředí Visual Studio 2008.

8.1. ASP.NET

Nejvýraznější výhodou je zlepšená podpora, pro webové sítě, které využívají AJAX. ASP.NET podporuje vývoj serverově orientovaného AJAXU sadou nových ovládacích prvků serveru a novými API.

ASP.NET dále podporuje vývoj klientově orientovaného AJAXU s novou klientskou knihovnou, která se nazývá Microsoft AJAX Library. Tato knihovna podporuje klientově a objektově orientovaný vývoj aplikací, který je nezávislý na prohlížečích. Používáním knihovny tříd ECMAScript (JavaScript), lze dosáhnout rozsáhlých funkcí uživatelského rozhraní, bez nutnosti komunikace se serverem. Visual Web Developer obsahuje vylepšenou podporu IntelliSense pro JavaScript, a dále podporuje knihovnu Microsoft AJAX Library.

Visual Web Developer a ASP.NET nyní podporují vytváření ASMX a WCF založených webových služeb, a jejich hladké použití pomocí Microsoft AJAX Library. Dále serverové aplikační služby, ověřování formulářů. Správa uživatelských práv a profily, jsou nyní vystavené jako Web služby, které se mohou použít v kompatibilních WCF aplikacích. ASP.NET umožňuje všem serverově založeným aplikacím sdílet tyto základní aplikační služby.

Další vylepšení ASP.NET zahrnují, ListView, LinqDataSource, ASP.NET Merge Tool, nový designový nástroj CSS, podpora LINQ pro SQL databáze.

8.2. Add-In a rozšiřitelnost

Knihovna System.AddIn.dll, poskytuje flexibilní podporu pro vývojáře rozšiřitelných aplikací. Představuje novou architekturu a model, což pomáhá vývojářům přidávat rozšíření do aplikace.

Model poskytuje tyto vlastnosti

- Discovery (objevení), vývojář může pohodlně najít a spravovat sady Add-inů ve více lokacích na počítači třídou AddInStore.
- Activation (aktivace), poté co si aplikace vybere Add-In, třída AddInToken se postará o zbytek, stačí jen vybrat úroveň Isolation a Sandboxingu.
- Isolation (izolace), izolační úroveň každého Add-inu je kontrolována hostem. Systém manipuluje s načítáním aplikačních domén a vypíná je, pokud jejich add-iny přesnatou pracovat.
- Sandboxing, vývojář může jednoduše nastavit úroveň důvěryhodnosti.
- UI Composition, model podporuje přímou změnu UI kompozice prvků WPF.

8.3. LINQ (Language Integrated Query)

LINQ je nová vlastnost Visual Studia 2008 a .NET Framework 3.5. LINQ rozšiřuje schopnosti dotazu jazykové syntaxe C# a Visual Basic, formou standardních, lehce naučitelných dotazových vzorů. Tato technologie může být rozšířená pro podporu potenciálně jakéhokoliv druhu datových úložišť. .NET Framework 3.5 obsahuje poskytovatele sestavy LINQ, která umožňuje použití LINQ, pro pokládání dotazů .NET Framework collections, databázím SQL Serveru, datovým sadám ADO.NET a XML dokumentům.

8.4. Peer-to-Peer Networking

Peer-to-Peer networking je síťovací technologie, která vylučuje server. Umožňuje pár síťovým zařízením komunikovat přímo mezi sebou a sdílet zdroje. Jmenný prostor System.Net.PeerToPeer poskytuje sadu tříd, které podporují PNRP (Peer Name Resolution

Protocol), což umožňuje objevení jiných peer uzlů přes objekty PeerName, které jsou registrované v peer-to-peer cloudu. PNRP dokáže rozlišit jména peerů na IP adresách IPv4 a IPv6.

8.5. Stromy výrazů

Stromy výrazů jsou novinkou v .NET Frameworku 3.5, a poskytují způsob, jak reprezentovat kód na jazykové úrovni, ve formě dat. Jmenný prostor System.Linq.Expressions obsahuje typy, jež jsou základními kameny stromu výrazů. Tyto typy mohou být použity, aby reprezentovaly různé typy kódových výrazů.

Stromy výrazů jsou značně využívány v LINQ dotazech, které jsou směřovány na vzdálené zdroje dat, jako například SQL databáze. Tyto dotazy jsou reprezentovány jako stromy výrazů, tato reprezentace umožňuje poskytovatelům dotazu prohledat je, a přeložit je do jazyka dotazu, specifického pro doménu.[11]

8.6. Service Pack 1

.NET Framework 3.5 se dočkal také service packu 1, ten byl vydán 11. Srpna 2008. Toto vydání přidávalo vylepšení po výkonové stránce, v závislosti na určitých podmínkách, zejména u WPF, kde bylo předpokládáno zvýšení výkonu o 20-45%.
Dále byly přidány dvě datové služby, ADO.NET Entity Framework a ADO.NET Data Services. System.Web.Routing a System.Web.Abstraction, byly přidány pro vývoj webu a jsou použity v ASP.NET MVC Framework.[12]

9. .NET Framework 4.0

.NET Framework 4.0 byl vydán 12. Dubna 2010. Vedle toho bylo vydáno i nové Visual Studio 2010. .NET Framework 4.0 přinesl vylepšení programovacích jazyků, C# na verzi 4.0 a Visual Basic .NET na verzi 10.0. Pro verzi 4.0 pak vyšly ještě tři menší updaty na verze 4.0.1, 4.0.2 a 4.0.3.

9.1. Diagnostika a výkon

Předchozí verze .NET Frameworku neposkytovaly žádný způsob, jak zjistit jestli určitá aplikační doména, nějak ovlivňuje ostatní domény, protože nástroje a API operačního

systému, jako je například správce úloh, byly určené jen pro procesní úroveň. .NET Framework 4.0 umožňuje dostat využití CPU a fyzické paměti, propočítané na jednotlivé aplikační domény.

Vývojář může monitorovat CPU a paměť individuálních aplikačních domén. Monitorování zdrojů aplikačních domén je možné, pomocí spravovaných a nativních API a ETW (Event Tracing for Windows) na hostitelském zařízení. Když je tato vlastnost povolená, sbírá statistiky o všech aplikačních doménách v procesu. Tato vlastnost se nazývá AppDomain.MonitoringIsEnabled.

9.2. Globalizace

Verze 4.0 poskytuje defaultní a specifické typy kultur, zlepšuje řízení řetězců a další vylepšení v oblasti globalizace a lokalizace.

9.3. Garbage Collection

.NET Framework 4.0 poskytuje sběr odpadu na pozadí, tento prvek nahrazuje současný sběr, a zlepšuje tak celkový výkon Frameworku.

9.4. DLR (Dynamic Language Runtime)

DLR je nové runtime prostředí, které přidává sadu služeb pro dynamické jazyky do CLR. DLR usnadňuje vyvíjet dynamické jazyky, aby běžely na .NET Frameworku, a přidávat dynamické vlastnosti do staticky psaných jazyků. Pro podporu DLR je přidán nový jmenný prostor System.Dynamic do .NET Frameworku.

Stromy výrazů jsou rozšířeny o nové typy, které reprezentují ovládací tok (control flow), například System.Linq.Expressions.LoopExpression a System.Linq.Expressions.TryExpression. Tyto nové typy jsou použity v DLR, nikoliv však v LINQ.

Navíc pár nových tříd, které podporují infrastrukturu .NET Frameworku jsou přidány do jmenného prostoru System.Runtime.CompilerServices.

9.5. 64-bitové operační systémy a procesy

Vývojář může identifikovat 64-bitové operační systémy a procesy, pomocí vlastností Enviroment.Is64BitOperatingSystem a Enviroment.Is64BitProcess. Dále může specifikovat 32-bitový nebo 64-bitový pohled na registry, pomocí výčtu Microsoft.Win32.RegistryView při otevření základních klíčů.

9.6. Paralelní programování

.NET Framework 4.0 představuje nový programovací model pro psaní vícevláknového a asynchronního kódu, který výrazně zjednodušuje práci vývojářům aplikací a knihoven. Nový jmenný prostor System.Threading.Tasks a s ním spojené typy, podporují nový model. Paralelní LINQ (Parallel LINQ - PLING), umožňuje podobnou funkcionalitu skrz určité syntaxe.

9.7. Vylepšení sítí

Bezpečnostní vylepšení pro ověřování systému Windows v pár třídách, System.Net.HttpWebRequest, System.Net.HttpListener, System.Net.Security.SslStream a System.Net.Security.NegotiateStream. Rozšířená ochrana je dostupná pro aplikace na Windows 7 a Windows Server 2008 R2.

Podpora pro přechod Network Address Translation (NAT) pomocí IPv6 a Teredo.
Nové počítadla síťového výkonu poskytují informace o objektech HttpWebRequest.
Podpora Secure Sockets Layer (SSL) pro System.Net.Mail.SmtpClient a přidružené třídy.
Zlepšená podpora pro hlavičky mailů ve třídě System.Net.Mail.MailMessage.

9.8. WEB

- Nová verze ASP.NET, verze 4, přináší nové vlastnosti.
- Základní služby, včetně nové API, které dovolují rozšířit cache. Nový aplikační preload manager, vlastnost automatického zapnutí.
- Webové formuláře, více integrovaná podpora pro směrování (routing) ASP.NET, vylepšená podpora pro Webové standardy, nové prvky pro kontrolu dat.
- MVC, obsahuje nové pomocné metody pro pohledy (view), podpora oddělených MVC aplikací, a asynchronních ovladačů.

- Microsoft Ajax, přidána podpora pro klientově orientované aplikace do Microsoft Ajax Library.
- Dynamic Data, obsahují podporu pro už vytvořené webové aplikace, dále podporu pro mnoho vztahů a dědičností, nové šablony a atributy pole, a vylepšené filtrování dat.

9.9. WPF

- Windows Presentation Foundation obsahují nové ovládací prvky, Calendar, DataGrid a DatePicker.
- Touch and Manipulation umožňuje vytvářet aplikace, které přijímají vstup z více dotyků současně, na Windows 7.
- Text, u něj je vylepšeno renderování a podporuje nastavování barvy vynechávky v textu a barvy označené položky v textovém boxu.

9.10. WCF

- Windows Communication Foundation, přináší Konfiguračně založenou aktivaci, což odstraňuje potřebu, mít .svc soubor.
- Integrace Syste.Web.Routing, tato vlastnost dá vývojáři větší kontrolu nad služební URL tím, že mu umožní používání nerozšiřitelných URL.
- Služby workflow, integrují WCF a WF poskytováním aktivit na posílání a přijímání zpráv a vlastnosti roztřídit zprávy na základě obsahu.

9.11. WF

- Windows Workflow Foundation, vylepšený model Workflow aktivit. Třída Activity poskytuje základní abstrakci chování Workflow.
- Obsáhlé, kompozitní nastavení aktivit, Workflow profituje z nových aktivit flow-control, které modelují tradiční strukturu flow-control, jako je Flowchart, TryCatch a Switch<T>.
- Expandovaná vestavěná knihovna aktivit, nové prvky knihovny aktivit obsahují nové aktivity flow-control, aktivity pro manipulaci s členy dat a aktivity pro ovládání transakcí.[13]

10. .NET Framework 4.5

Verze 4.5 byla vydána 15. Září 2012. Do této verze by přidána sada nových nebo vylepšených vlastností. Tato verze používá CLR verze 4.0, s pár novými runtime funkcemi. Přinesla s sebou také nové verze programovacích jazyků, C# verze 5.0 a Visual Basic .NET verze 11.0. Kromě toho, s ní přišlo i nové vývojové prostředí Visual Studio 2012.

10.1. Aplikace ve stylu Metro

Tyto aplikace jsou navrhovány pro specifickou formu faktorů a vlivu výkonu operačního systému Windows. Podmnožina .NET Frameworku, je dostupná pro vývoj Metro aplikací pro Windows 8, použitím jazyků C# nebo Visual Basic. Tato podmnožina se nazývá .NET API pro aplikace (.NET APIs for apps).

Verze .NET Frameworku, runtimu a knihoven, používaných pro metro aplikace jsou součástí nového Windows Runtime, což je novou platformou a aplikačním modelem pro aplikace stylu Metro. Je to ekosystém, který obsahuje mnoho platforem a jazyků, včetně .NET Framework, C++ a HTML5/JavaScript.[14]

10.2. Nové funkce

10.2.1. Redukce vynucených restartování systému při instalaci rozhraní

.NET Framework používá instalační službu tzv. Restart Manager, aby zabránila restartu systému, kdykoliv je to potřeba. Když instalační program, instaluje rozhraní .NET Framework 4.5, může komunikovat a využívat funkce, již zmíněného Restart Managera.

Instalace .NET Framework 4.5 vyžaduje restartování systému, pokud v průběhu instalace běží aplikace používající .NET Framework 4, protože .NET Framework 4.5 přepisuje soubory verze 4.0, čili potřebuje, aby přepisované soubory byly k dispozici. Ve většině případů jde zamezit restartování systému preemptivní detekcí a uzavíráním aplikací, používajících .NET Framework 4. Některé systémové aplikace nemohou být ukončeny, v tom případě je restartování systému nutné.[15]

10.2.2. Podpora pole, většího než 2GB na 64-bitových platformách

Podpora pro pole, která jsou větší než 2 gigabajty. Tato vlastnost může být aktivována v konfiguračním souboru aplikace. A to pomocí prvku <gcAllowVeryLargeObjects>, jenž má další nastavení omezení velikosti objektu a pole. Nastavení probíhá za pomocí atributu enabled.

```
<configuration>
  <runtime>
    <gcAllowVeryLargeObjects enabled="true" />
  </runtime>
</configuration>
```

Configuration je hlavní prvek každého konfiguračního souboru používaného CLR a aplikacemi .NET Framework.

Runtime informuje o možnostech inicializace runtimu.

Použitím elementu gcAllowVeryLargeObjects v konfiguračním souboru aplikace, umožní řetězcům mít větší velikost než 2GB, ovšem nemění to další limity velikosti objektu nebo velikosti řetězce.[16]

10.2.3. Schopnost omezení doby, po kterou se bude modul regulárního výrazu snažit o překlad regulárního výrazu.

Jmenný prostor: System.Text.RegularExpressions

Třída: Regex

Vlastnost: Regex.MatchTimeout – Dostává čas vypršení intervalu stávající instance.

Syntaxe

```
public TimeSpan MatchTimeout { get; }
```

Typ: System.TimeSpan

Hodnota vlastnosti

Maximální časový interval, který může uplynout ve vzorově shodné operaci před tím, než

je vyhozena výjimka RegexMatchTimeoutException nebo Regex.InfiniteMatchTimeout pokud jsou časy vypršení zakázané.

Vlastnost MatchTimeout definuje přibližný maximální časový interval pro Regex instanci na provedení jedné vhodné operace před tím, než operaci vyprší čas. Engine regulárních výrazů hodí výjimku RegexMatchTimeoutException během další kontroly času, pokud čas uplynul. To zabraňuje enginu regulárních výrazů zpracovávat vstupy, které vyžadují nadměrný backtracking.

Tato vlastnost je pouze pro čtení. Hodnoty se dají nastavovat explicitně, pro individuální Regex objekt, voláním konstruktoru Regex.Regex(String, RegexOption, TimeSpan). Dále jde nastavit hodnotu pro všechny vhodné Regex operace v aplikační doméně, voláním metody AppDomain.SetData a poskytnutím hodnoty pro vlastnost REGEX_DEFAULT_MATCH_TIMEOUT. Jinak se používá defaultně nastavená hodnota a odpovídajícím operacím nevyprchává čas.[17]

Konstrukce třídy regex s časem vypršení 1000ms.

```
Var regex = new Regex (
            "čas vypršel",
             RegexOptions.None,
             TimeSpan.FromMiliseconds(1000));
```

10.2.4. Definice výchozí kultury pro doménu aplikace

Výchozí kultura se definuje pomocí třídy CultureInfo, tato třída obsahuje informace o specifické kultuře domény, kde je aplikace umístěna. Součástí těchto informací jsou názvy pro kulturu, systém psaní, jaký byl použit kalendář, formátování dat kalendáře a řazení řetězců.

Jmenný prostor: System.Globalization

Třída: ClutureInfo

Vlastnost: DefaultThreadCurrentCulture

Syntaxe

```
public static CultureInfo DefaultThreadCurrentCulture { get; set;}
```

Typ: System.Globalization.CultureInfo

Hodnota vlastnosti

Defaultní kultura pro vlákna stávající domény, ve které je aplikace, nebo hodnota null, když je kultura aktuálního systému stejná jako defaultní kultura v doméně.

V předchozí verzi .NET Frameworku byla výchozí kultura všech vláken nastavena na kulturu systému Windows. Pro aplikace, které se liší stávající kulturou od výchozí systémové kultury, je toto chování často nežádoucí. Proto ve verzi .NET Framework 4.5, vlastnost DefaultThreadCulture, umožňuje aplikacím definovat výchozí kulturu všech vláken v doméně s aplikací.

Změna kultury stávajícího vlákna

```
CultureInfo culture;
culture = CultureInfo.CreateSpecificCulture("cs-CZ");
Thread.CurrentThread.CurrentCulture = culture;
```

Další podobnou vlastností je vlastnost DefaultThreadCurrentUICulture

Je také ve třídě CultureInfo, má stejnou syntaxi kromě jména vlastnosti.

Hodnota vlastnosti

Defaultní UI kultura pro vlákna stávající domény s aplikací, nebo hodnota null pokud je UI kultura aktuálního systému stejná jako defaultní UI kultura vláken v doméně.[18]

10.2.5. Konzole podporuje kódování Unicode (UTF-16)

Změna kódování výstupu konzole[14]

```
Console.OutputEncoding = Encoding.UTF16;
```

10.2.6. Třída SortVersion

Třída SortVersion poskytuje informace o verzi Unicode, která je použita na porovnávání a seřazování řetězců.

Jmenný prostor: System.Globalization

Konstruktor

- SortVersion, vytvoří novou instanci třídy.

Vlastnosti

- FullVersion, dostává číslo plné verze objektu SortVersion. Hodnota vlastnosti FullVersion odráží verzi Unicode používanou na normalizaci a porovnávání řetězců.
- SortId, dostává globální unikátní identifikátor pro tento objekt SortVersion. Hodnota vlastnosti SortId odráží kulturu, jejíž konvence ovlivňují porovnávání a řazení řetězců.

Metody

- Equals(Object), vrací hodnotu, která ukazuje, zda je tato instance SortVersion totožná se specifikovaným objektem. (přetěžuje Object.Equals(Object))

  Syntaxe

  ```
  public override bool Equals (Object obj)
  ```

  Vrací hodnotu true, pokud obj je objekt SortVersion, který reprezentuje stejnou verzi jako instance, jinak false. Dva objekty SortVersion jsou totožné, pokud jejich FullVersion a SortId vlastnosti jsou totožné.
- Equals(SortVersion), vrací hodnotu, která ukazuje, zda je tato instance SortVersion totožná se specifikovaným objektem SortVersion.

  Syntaxe

  ```
  public bool Equals (SortVersion other)
  ```

  Vrací hodnotu true, když other reprezentuje stejnou verzi jako instance, jinak false. Dva objekty SortVersion jsou totožné, pokud jejich vlastnosti FullVersion a SortId jsou totožné.

Operátory

- Equality, indikuje, jestli jsou dvě instance SortVersion totožné.
- InEquality, indikuje, jestli nejsou dvě instance SortVersion totožné.

Od verze frameworku 2.0 po verzi 4.0, každá verze zahrnovala tabulky, které obsahovaly typ váhy a dat na řetězcovou normalizaci, která byla založena na každé konkrétní verzi Unicode. Ve verzi frameworku 4.5, výskyt těchto tabulek záleží na operačním systému.

Verze .NET Framework	Operační systém	Verze Unicode
.NET Framework 4	Všechny operační systémy	Unicode 5.0
.NET Framework 4.5	Windows 7	Unicode 5.0
.NET Framework 4.5	Windows 8	Unicode 6.0

Jak je vidět, výsledná verze kódování závisí jak na operačním systému, tak na verzi .NET Framework.

Třída SortVersion poskytuje informace o verzi Unicode používané .NET Frameworkem pro porovnávání a řazení řetězců. To umožňuje vývojářům psát aplikace, které mohou detekovat a úspěšně řídit změny ve verzi Unicodu, která je použita na porovnávání a roztřízení řetězců aplikace.[19]

10.2.7. Třída CustomReflectionContext

Schopnost přizpůsobit kontext odrazu, aby přeskočil defaultní chování odrazu, pomocí třídy CustomReflectionContext.

Jmenný prostor: System.Reflection.Context

Konstruktory

- CustomReflectionContext(), inicializuje novou instanci třídy.
- CustomReflectionContext(ReflectionContext), inicializuje novou instanci třídy, jejíž základem je specifický kontext odrazu.

Metody

- AddProperties, když je přeskočena (overriden) v děděné třídě, poskytuje kolekci dodatečných vlastností pro specifikovaný typ, zastoupenou v kontextu tohoto odrazu.

  Syntaxe

  ```
  Protected virtual IEnumerable<PropertyInfo> AddProperties (Type type)
  ```

  Parametr type, typ pro přidané vlastnosti.

  Vrací hodnotu typu System.Collections.Generic.IEnumerable<PropertyInfo>, kolekce dodatečných vlastností pro specifikovaný typ. Při přeskočení tato metoda specifikuje, které vlastnosti mají být přidány do daného typu. Pro vytvoření těchto vlastností se používá metoda CreateProperty.

- CreateProperty(Type, String, Func<Object, Object>, Action<Object, Object>), vytvoří objekt, který představuje vlastnost, přidanou do typu a je použit v metodě AddProperties.

  Syntaxe

  ```
  protected PropertyInfo CreateProperty (Type propertyType, string name,
  Func<Object, Object> getter, Action<Object, Object> setter)
  ```

  Parametr propertyType, typ vytvářené vlastnosti. Parametr name je její název. Parametr getter, objekt představující get vlastnosti a setter představuje set. Návratová hodnota typu System.Reflection.PropertyInfo, je objekt, který představuje vlastnost.

  Objekty vracené touto metodou nejsou kompletními objekty PropertyInfo, a měly by být použity pouze v kontextu metody AddProperties.

- CreateProperty(Type, String, Func<Object, Object>, Action<Object, Object>, IEnumerable<Attribute>, IEnumerable<Attribute>, IEnumerable<Attribute>), vytvoří objekt, který představuje vlastnost, přidanou do typu a je použit v metodě AddProperties se specifikovanými atributy.

Syntaxe

```
protected PropertyInfo CreateProperty (Type propertyType, string name,
Func<Object, Object> getter, Action<Object, Object> setter,
IEnumerable<Attribute> propertyCustomAttributes,
IEnumerable<Attribute> getterCustomAttributes,
IEnumerable<Attribute> setterCustomAttributes)
```

Parametr propertyType, typ vytvářené vlastnosti. Parametr name je její název. Parametr getter, objekt představující get vlastnosti a setter představuje set. Nově v této metodě je parametr propertyCustomAttributes, představuje kolekci atributů, které budou aplikovány na vlastnost. Dále getterCustomAttributes, kolekce atributů aplikovaných na get vlastnosti, setterCustomAttributes představuje kolekci atributů aplikovaných na set vlastnosti.
Objekty vracené touto metodou nejsou kompletními objekty PropertyInfo, a měly by být použity pouze v kontextu metody AddProperties.

- GetCustomAttributes(MemberInfo, IEnumerable<Object>), když je přeskočena (overriden) v děděné třídě, poskytne seznam atributů pro specifikovaného člena (memeber), zastoupený v kontextu tohoto odrazu.

  Syntaxe

  ```
  protected virtual IEnumerable<Object> GetCustomAttributes(
  MemberInfo member,
  IEnumerable<Object> declaredAttributes)
  ```

  declaredAttributes kolekci atributů člena, v jeho stávajícím kontextu.
  Návratovou hodnotou je kolekce, která představuje, atributy specifického člena, v kontextu svého odrazu.

- GetCustomAttributes(ParameterInfo, IEnumerable<Object>), víceméně je stejná jako metoda GetCustomAttributes(MemberInfo, IEnumerable<object>), jenom poskytuje seznam atributů, pro určitý parametr, nikoliv člena.

  Syntaxe

  ```
  protected virtual IEnumerable<Object> GetCustomAttributes(
  ParameterInfo parameter,
  IEnumerable<Object> declaredAttributes)
  ```

Parametr parameter, určuje, pro jaký parametr mají být vráceny atributy. Návratovou hodnotou je opět kolekce, která představuje, atributy specifického parametru, v kontextu svého odrazu.

- Třída obsahuje ještě další metody, ty jsou však již děděny z jiných tříd.[20]

10.3. ASP.NET

- Podpora pro nové formulářové typy HTML5.
- Podpora pro přiřazování ve webových formulářích, umožňuje přiřadit data přímo k metodě, která s nimi pracuje. Dále pak umožňuje automaticky konvertovat vstup od uživatele do datových typů .NET Frameworku.
- Podpora nevtíravého JavaScriptu ve validačních scriptech na straně klienta.
- Podpora protokolu WebSocket.
- Podpora čtení a psaní HTTP odpovědí a požadavků asynchronně.
- Integrované zabezpečení proti skriptovacím útokům cross-site, pomocí knihovny AntiXSS.

10.4. Asynchronní souborové operace

V .NET Frameworku 4.5, byly přidány nové asynchronní prvky do jazyků C# a Visual Basic. Tyto prvky přinášejí model, založený na úlohách, pro provádění asynchronních operací. Pro použití tohoto nového modelu, je potřeba asynchronních metod v I/O třídách.

10.5. Paralelní programování

Verze . NET Framework 4.5, poskytuje nové prvky a vylepšení pro paralelní programování. Ty zahrnují, vylepšení výkonu, větší kontrolu, lepší podporu pro asynchronní programování, novou knihovnu dataflow a vylepšenou podporu pro paralelní debuggování a analýzu výkonu.

C# .NET ve verzi 4.5 podporuje asynchronní metody.

10.5.1. Asynchronní metoda

Asynchronní metoda je metoda, která vrátí svou hodnotu, až po nějakém časovém intervalu. Po dobu tohoto intervalu na ní program nečeká a pokračuje normálně dále. Asynchronní metody nalezneme ve spoustě tříd .NET, např. práce se soubory, třídy StreamWriter, StreamReader, XmlReader a StorageFile, pak v třídách pro webový přístup jako jsou HttpClient a SyndicationClient dále ve třídách pro práci s obrázky, MediaCapture, BitmapEncoder a BitmapDecoder. Asynchronní metody končí na slovo Async, např. metoda StreamWriter.WriteLineAsync() a jejich návratovou hodnotou je objekt typu Task, na kterou si počkáme pomocí klíčového slova ***await***. Metoda, která používá await musí být označena slovem ***async***, které označuje, že se jedná o asynchronní metodu.

10.5.2. Vlákna

Technologie spočívá ve snaze neblokovat hlavní vlákno metodami, které čekají na dokončení úlohy. Asynchronní programování nemusí používat MultiThreading, jedině pokud chceme přenést zátěž programu na jiné vlákno, můžeme pomocí metody Task.Run().

Asynchronní metodu smíme volat jen v asynchronní metodě.[21]

10.6. Networking

.NET Framework 4.5 poskytuje nové programovací rozhraní pro HTTP aplikace. K němu jsou přidruženy i dva nové jmenné prostory System.Net.Http a System.Net.Http.Headers.

Podpora je také zahrnuta pro nové programovací rozhraní, pro akceptování a interakci se spojením WebSocket, pomocí již exitující třídy HttpListener a příbuzných tříd, a nově vytvořeného jmenného prostoru, System.Net.WebSockets.

10.6.1. Další novinky

- Podpora parsování IDN (Internationalized Domain Name), třída Uri a příbuzné třídy.
- Podpora EAI (Email Address Internalization), jmenný prostor System.Net.Mail.
- Vylepšená podpora Piv6, jmenný prostor System.Net.NetworkInformation.

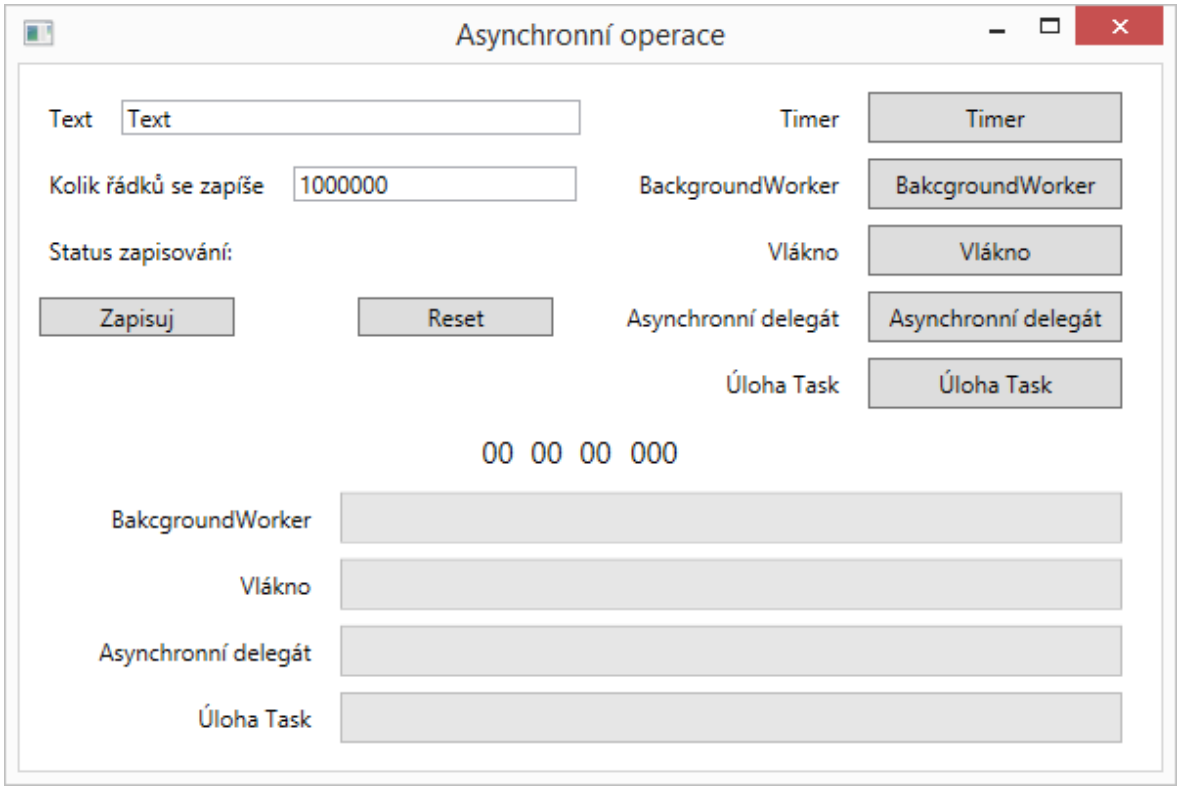
10.7. WPF

- Windows Presentation Foundation, ve verzi Frameworku 4.5 přináší. Nová kontrola Ribbon, která umožňuje implementovat ribbon uživatelské prostředí, které hostuje Quick Access Toolbar (lištu rychlého přístupu), Application Menu (aplikační menu) a záložky.
- Nové rozhraní INotifyDataErrorInfo, které podporuje synchronní a asynchronní validaci dat.
- Datová vazba na statické vlastnosti, datová vazba na vlastní typy, které implementují rozhraní ICustomTypeProvider, a vyhledávání informací datových vazeb z jejich výrazu.

10.8. WCF

- Windows Communication Foundation, v nové verzi byly přidány prvky, které zjednodušují psaní a údržbu WCF aplikací.
- Schopnost zjednodušeně konfigurovat kompatibilitu ASP.NET.
- Vylepšení třídy XmlDictionaryReaderQuotas, na zredukování pravděpodobnosti, že bude muset vývojář manuálně konfigurovat kvóty pro čtečky XML slovníku.
- Nové mapování HTTPS protokolu, které usnadňuje vystavit endpoint přes HTTPS pomocí IIS (Internet Information Services).
- Podpora transportu UDP, který umožňuje vývojáři psát služby, které používají tzv. “fire and forget“ zprávy. Klient pošle zprávu službě a nečeká od ní žádnou odpověď.[14]

11. Praktická část – program demonstrující asynchronní programování



Vytvořil jsem WPF aplikaci, která ukazuje jak je v nové verzi .NET 4.5 vylepšeno asynchronní programování. Aplikace asynchronní operace nám ukazuje, jak se dělá asynchronní programování v nejnovějším prostředí .NET a, další čtyři způsoby, kterými se dá dělat.

Aplikace umí asynchronně zapisovat do souboru nějaký text na principu nejnovější technologie a dále ukazuje, jak se řeší asynchronní programování pomocí komponenty BackgroundWorker, vytvořením nového vlákna, asynchronním delegátem a úlohou Task.

Timer je pak přidán, aby aplikace dělala něco v hlavním vláknu, kvůli demonstraci asynchronního zapisování do souboru.

Tlačítko reset slouží pouze pro vrácení UI aplikace do počátečního stavu.

Tuto aplikaci jsem naprogramoval v jazyce C#.

11.1. Asynchronní zapisování do souboru

pomocí asynchronní metody (klíčová slova async a await)

```
private async void zapisujButton_Click(object sender, RoutedEventArgs e)
        {
            zapisujButton.IsEnabled = false;
            Zapisovac z = new Zapisovac();
            zapisovaniStatusLabel.Content = "zpracovává se";
            string str = await z.ZapisAsync(textTextBox.Text,
                  Convert.ToInt32(radkyTextBox.Text));
            zapisovaniStatusLabel.Content = str;
            zapisujButton.IsEnabled = true;
        }
```

Metoda zapisujButton_Click je označena slovem async, metoda zakáže tlačítko zapisuj, vytvoří instanci třídy Zapisovac a pomocí této instance zavolá asynchronní metodu ZapisAsync, pro zapisování do souboru, která vrací string, na který se čeká, pomocí klíčového slova await. Před zapisováním se změní popisek statusu na „zpracovává se“ a když zapisování vrátí string, přepíše se podle něj popisek statusu. Nakonec se opět povolí tlačítko pro ZapisujButton.

Metoda ZapisAsync na asynchronní zapisování do souboru

```
private string soubor =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData
), "soubor.txt");

public async Task<string> ZapisAsync(string s, int i)
        {
            using (StreamWriter sw = new StreamWriter(soubor))
            {
                for (int x = 0; x < i; x++)
                {
                    await sw.WriteLineAsync(s);
                }
                return "úspěšně zapsáno";
            }
        }
```

Tato metoda je také označena slovem async, a pomocí třídy StreamWriter a její metody WriteLineAsync(), na kterou se čeká, zapisuje do souboru text s na počet řádků i, po zapsání vrací řetězec „úspěšně zapsáno“.

11.2. Komponenty BackgroundWorker a DispatcherTimer

```
BackgroundWorker worker = new BackgroundWorker();
DispatcherTimer timer = new DispatcherTimer();

public MainWindow()
        {
            InitializeComponent();

            worker.WorkerReportsProgress = true;
            worker.DoWork += worker_DoWork;
            worker.ProgressChanged += worker_ProgressChanged;
            worker.RunWorkerCompleted += worker_RunWorkerCompleted;

            timer.Interval = TimeSpan.FromMilliseconds(1);
            timer.Tick += timer_Tick;
        }
```

Komponenty BackgroundWorker a DispatcherTimer se inicializují ihned po spuštění aplikace, kde se zároveň nastaví jejich metody.

U BackgroundWorkera co má vykonávat (DoWork), co se stane, když se změní hodnota (ProgressChanged) a co se udělá na závěr (RunWorkerCompleted).

U DispatcherTimeru pak interval tiků (Interval) a co má vykonávat při tiku (Tick).

11.2.1. DispatcherTimer

```
private void timerButton_Click(object sender, RoutedEventArgs e)
        {
            timerButton.IsEnabled = false;
            timer.Start();
        }

void timer_Tick(object sender, EventArgs e)
        {
            hodinyTextBlock.Text =  DateTime.Now.Hour.ToString();
            minutyTextBlock.Text = DateTime.Now.Minute.ToString();
            sekundyTextBlock.Text = DateTime.Now.Second.ToString();
            milisekundyTextBlock.Text = DateTime.Now.Millisecond.ToString();
        }
```

Pomocí DispatcherTimeru bylo jednoduše vytvořeno vypisování času. Tlačítkem se aktivuje timer a při tiku vypisuje časové údaje do UI. Důležité je, že pracuje v hlavním vláknu, stejně jako asynchronní metoda zapisování do souboru.

11.2.2. BackgroundWorker

```
private void backgroundWorkerButton_Click(object sender, RoutedEventArgs e)
        {
            if (!worker.IsBusy)
            {
                backgroundWorkerButton.IsEnabled = false;
                worker.RunWorkerAsync();
            }
        }

void worker_DoWork(object sender, DoWorkEventArgs e)
        {
            for (int i = 0; i <= 100; i++)
            {
                worker.ReportProgress(i);
                Thread.Sleep(50);
            }
        }

void worker_ProgressChanged(object sender, ProgressChangedEventArgs e)
        {
            backgroundWorkerStatusLabel.Content = "Running";
            backgroundWorkerProgressBar.Value = e.ProgressPercentage;
            backgroundWorkerLabel.Content = e.ProgressPercentage;
        }

void worker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
        {
            backgroundWorkerStatusLabel.Content = "Hotovo";
            backgroundWorkerButton.IsEnabled = true;
        }
```

Tlačítko nám ošetří a zapne worker pomocí RunWorkerAsync(). Worker pak vykonává metodu worker_DoWork, kde pomocí ReportProgress posílá hodnoty do metody worker_ProgressChanged. Ta se zavolá při každým dalším cyklu a mění prvky UI, popisek a progressbar, podle do ní poslaného parametru.

Po dokončení se zavolá metoda worker_RunWorkerCompleted, která jen změní popisek a povolí tlačítko.

11.3. Vlákno

```
public delegate void StringCallback(string str);
public StringCallback delegatVlakno;
private void vlaknoButton_Click(object sender, RoutedEventArgs e)
        {
            vlaknoButton.IsEnabled = false;
            delegatVlakno = new StringCallback(UpdateVlakno);
            Thread vlakno = new Thread(VlaknoTest);
            vlakno.Start();
        }

private void VlaknoTest()
        {
            for (int i = 0; i <= 100; i++)
            {
                Thread.Sleep(50);
                vlaknoLabel.Dispatcher.Invoke(delegatVlakno, new object[] {
                     i.ToString() }
                );
            }
        }

private void UpdateVlakno(string str)
        {
            vlaknoStatusLabel.Content = "Running";
            vlaknoLabel.Content = str;
            vlaknoProgressBar.Value = Convert.ToInt16(str);
            if (Convert.ToInt16(str) == 100)
            {
                vlaknoStatusLabel.Content = "Hotovo";
                vlaknoButton.IsEnabled = true;
            }
        }
```

Tlačítkem se vytvoří delegát delegatVlakno, který deleguje string metodě UpdateVlakno. Vytvoří se nové vlákno, přidělí se mu metoda VlaknoTest a uvede se do chodu metodou Start(). VláknoTest pak vykonává svou funkcionalitu na jiném vlákně a posílá svůj stav pomocí delegáta delegatVlakno do delegované metody UpdateVlakno, která přepisuje položky UI a pokud je cyklus dokončen, opět povolí tlačítko a přepíše popisek.

11.4. Asynchronní delegát

```
public delegate void StringCallback(string str);
private StringCallback delegatUpdateAD;
private StringCallback delegatHotovoAD;
private void asyncDelegatButton_Click(object sender, RoutedEventArgs e)
        {
            asyncDelegatButton.IsEnabled = false;
            delegatUpdateAD = new StringCallback(UpdateAD);
            delegatHotovoAD = new StringCallback(HotovoAD);
            Func<int, int, int> metoda = Funkce;
            metoda.BeginInvoke(0, 100, Hotovo, metoda);
            asyncDelegatStatusLabel.Content = "Running";
        }

public int Funkce(int a, int b)
        {
            string str;
            for (; a <= b; a++)
            {
                str = a.ToString();
                Dispatcher.Invoke(this.delegatUpdateAD, new Object[] { str });
                Thread.Sleep(50);
            }
            return 1;
        }

public void Hotovo(IAsyncResult vysledek)
        {
            var metoda = (Func<int, int, int>)vysledek.AsyncState;
            int x = metoda.EndInvoke(vysledek);
            Dispatcher.Invoke(this.delegatHotovoAD, new Object[] { x.ToString()
            });
        }

public void UpdateAD(string str)
        {
            asyncDelegatLabel.Content = str;
            asyncDelegatProgressBar.Value = Convert.ToInt32(str);
        }

public void HotovoAD(string str)
        {
            asyncDelegatStatusLabel.Content = "Hotovo";
            asyncDelegatButton.IsEnabled = true;
        }
```

Definují se dva delegáti na delegování stringů. Nejdůležitější je delegát metoda (Func<>), do kterého si uložíme metodu Funkce a pomocí metody BeginInvoke() ji asynchronně spustíme se vstupními parametry 0 a 100 a callback metodami Hotovo a metoda. Pak už se jen přepíše popisek a vypne tlačítko.

Metoda Funkce deleguje stav cyklu pomocí delegáta delegatUpdateAD, který deleguje string do metody UpdateAD, která se stará o změnu popisku a stavu progressBaru.

Metoda Hotovo má jako parametr objekt IAsyncResult, z tohoto parametru dostaneme původního delegáta a na něm zavoláme metodu EndInvoke(). Ta se provede, když vlákno dokončí svou funkci. A int x se deleguje jako string do metody HotovoAD, která přepíše popisek a povolí tlačítko, pomocí delegáta delegatHotovoAD.

11.5. Task

Úloha Task, běží paralelně, jako background vlákno ThreadPoolu.

```
public delegate void StringCallback(string str);
public StringCallback delegatUloha;
private async void taskButton_Click(object sender, RoutedEventArgs e)
        {
            taskButton.IsEnabled = false;
            delegatUloha = new StringCallback(UpdateTask);
            Uloha u = new Uloha(this);
            Task<int> task = Task.Run(() => u.TaskUloha());
            task.Wait(100);
            taskStatusLabel.Content = task.Status.ToString();
            await task;
            taskStatusLabel.Content = task.Status.ToString();
            taskButton.IsEnabled = true;
        }

private void UpdateTask(string str)
        {
            taskLabel.Content = str;
            taskProgressBar.Value = Convert.ToInt32(str);
        }
```

Metoda také využívá async a await, vytvoří se delegát, který posílá string, ten se s celou instancí třídy pošle do třídy Uloha a z této třídy se volá pomocí Task.Run() metoda TaskUloha(), která vrací int. Metodu wait používám, aby se správně přepsal popisek, a program čeká na návratovou hodnotu úlohy. Status úlohy se pak vypíše do popisku a obnoví se tlačítko.

Metoda TaskUloha() ve třídě Uloha s kontruktorem

```
MainWindow window;

public Uloha(MainWindow w)
        {
            window = w;
        }

public int TaskUloha()
        {
            int i = 0;
            string str;
            for (; i <= 100; i++)
            {
                str = i.ToString();
                Thread.Sleep(50);
                window.Dispatcher.Invoke(window.delegatUloha, new Object[] { str
                      });
            }
            return 0;
        }
```

Metoda opět provádí cyklus a při každém cyklu posílá svůj string str do delegované metody UpdateTask.

12. Závěr

V bakalářské práci byl, zachycen postupný vývoj .NET Frameworku až po nejnovější verzi 4.5. Nejdůležitější změny a novinky, týkající se 4.5 jsou rozebrány více do hloubky, a je naprogramovaná aplikace v jazyce C#, která ukazuje vylepšení asynchronního programování v nové verzi frameworku.

V aplikaci je dobře ukázáno, jak se dá v nové verzi řešit asynchronní zapisování do souboru. Pokud se aplikace zatíží při debuggování, díky novým vláknům, zapne se DispatcherTimer a provede se asynchronní zápis do souboru, práce nových vláken se zpomalí relativně souměrně, čas, který běží na hlavním vláknu, se zasekává nejvíce. Při spuštění aplikace v běžném provozu vše běží krásně plynule.

Asynchronní metody mi přijdou výhodnější než vytváření nových vláken, hlavně kvůli jednoduchosti kódu. Není nutné ošetřovat složitě proměnné, se kterými pracuje více vláken, je dobře zajištěna návratová hodnota a vstupní parametry metody, kterou chci vykonat asynchronně a pokud chci hlavnímu vláknu trochu ulehčit, stačí pomocí Task.Run() zátěž přenést na jiné vlákno.

Hlavně díky tomu mi vylepšení v oblasti paralelního programování přijde jako velké ulehčení práce pro programátora. Asynchronní metody jsou dalším krokem, ve snaze Microsoftu, zdokonalit asynchronní operace. Ve verzi 4.0 přišli s úlohami, teď asynchronní metody, no jsem zvědavý, s čím přijdou příště.

13. Seznam použitých zdrojů

1. MICROSOFT, Msdn.microsoft.com/: Overview of the .NET Framework [online]. 2014 [cit. 2014-2-22] Dostupné z: http://msdn.microsoft.com/en-us/library/zw4w595w%28v=vs.71%29.aspx

2. MICROSOFT, Msdn.microsoft.com/: .NET Framework Versions and Dependencies [online]. 2014 [cit. 2014-2-22] Dostupné z: http://msdn.microsoft.com/en-us/library/bb822049%28v=vs.110%29.aspx

3. TheBestCSharpProgrammerInTheWorld, Thebestcsharpprogrammerintheworld.com/: Base Class Library (BCL) [online]. 2010 [cit. 2014-2-22] Dostupné z: http://www.thebestcsharpprogrammerintheworld.com/fundamentals/Base-Class-Library.aspx

4. BĚHÁLEK, Marek. Programovací jazyk C# [online]. Ostrava : VŠB-TU Ostrava, 2007. 144s. Učební text. TU Ostrava, VŠB-FEI, katedra informatiky. Dostupné z WWW: <http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text.pdf>.

5. SearchSOA, Searchsoa.techtarget.net/: Common Language Infrastructure(CLI) [online]. 2007 [cit. 2014-2-23] Dostupné z: http://searchsoa.techtarget.com/definition/Common-Language-Infrastructure

6. MICROSOFT, Msdn.microsoft.com/: Common Language Runtime (CLR) [online]. 2014 [cit. 2014-2-23] Dostupné z: http://msdn.microsoft.com/cs-cz/library/8bs2ecf4(v=vs.110).aspx

7. FAJT, Stanislav. Architektura Microsoft .NET Framework – 1. díl. [online]. 2006 [cit. 2014-2-23]. Dostupné z: http://programujte.com/clanek/2006042604-architektura-microsoft-net-framework-1-dil/

8. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 1.1 [online]. 2014 [cit. 2014-2-23] Dostupné z: http://msdn.microsoft.com/en-us/library/9wtde3k4%28v=vs.80%29.aspx

9. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 2.0 [online]. 2014 [cit. 2014-2-24] Dostupné z: http://msdn.microsoft.com/en-us/library/t357fb32%28v=vs.80%29.aspx

10. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 3.0 [online]. 2014 [cit. 2014-2-28] Dostupné z: http://msdn.microsoft.com/en-us/library/ms171868%28v=vs.85%29.aspx

11. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 3.5 [online]. 2014 [cit. 2014-3-1] Dostupné z: http://msdn.microsoft.com/en-us/library/bb332048%28v=vs.90%29.aspx

12. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 3.5 SP1 [online]. 2014 [cit. 2014-3-2] Dostupné z: http://msdn.microsoft.com/en-us/library/cc713697%28v=vs.90%29.aspx

13. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 4.0 [online]. 2014 [cit. 2014-3-6] Dostupné z: http://msdn.microsoft.com/en-us/library/ms171868%28v=vs.100%29.aspx

14. MICROSOFT, Msdn.microsoft.com/: What's New in the .NET Framework Version 4.5 [online]. 2014 [cit. 2014-3-10] Dostupné z: http://msdn.microsoft.com/en-us/library/ms171868%28v=vs.110%29.aspx

15. MICROSOFT, Msdn.microsoft.com/: Reducing System Restarts During .NET Framework 4.5 Installations [online]. 2014 [cit. 2014-3-15] Dostupné z: http://msdn.microsoft.com/en-us/library/hh527997%28v=vs.110%29.aspx

16. MICROSOFT, Msdn.microsoft.com/: <gcAllowVeryLargeObjects> Element [online]. 2014 [cit. 2014-3-15] Dostupné z: http://msdn.microsoft.com/en-us/library/hh285054%28v=vs.110%29.aspx

17. MICROSOFT, Msdn.microsoft.com/: Regex.MatchTimeout Property [online]. 2014 [cit. 2014-3-15] Dostupné z: http://msdn.microsoft.com/en-us/library/system.text.regularexpressions.regex.matchtimeout%28v=vs.110%29.aspx

18. MICROSOFT, Msdn.microsoft.com/: CultureInfo Class [online]. 2014 [cit. 2014-3-16] Dostupné z: http://msdn.microsoft.com/en-us/library/system.globalization.cultureinfo%28v=vs.110%29.aspx

19. MICROSOFT, Msdn.microsoft.com/: SortVersion Class [online]. 2014 [cit. 2014-3-16] Dostupné z: http://msdn.microsoft.com/en-us/library/system.globalization.sortversion%28v=vs.110%29.aspx

20. MICROSOFT, Msdn.microsoft.com/: CustomReflectionContext Class [online]. 2014 [cit. 2014-3-16] Dostupné z: http://msdn.microsoft.com/en-us/library/system.reflection.context.customreflectioncontext%28v=vs.110%29.aspx

21. ITNETWORK, itnetwork.cz/: Asynchronní programování v C# .NET - Async a await [online]. 2014 [cit. 2014-11-23] Dostupné z: http://www.itnetwork.cz/c-sharp-tutorial-asynchronni-programovani-async-await