Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Datové služby ve Windows

Zdeněk Špinka

© 2012 ČZU v Praze

Místo této strany vložíte zadání diplomové práce.

(Do jedné vazby originál a do druhé kopii)

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Datové služby ve Windows" jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne

_____

Poděkování

Rád bych touto cestou poděkoval:
Vedoucímu své práce, Ing. Martinu Papíkovi, Ph.D. za vedení práce a jeho rady a doporučení.
Mému vedoucímu Petrovi Švihálkovi za to že mi umožnil pracovat na tomto projektu.
Mým kolegům, kteří mi pomáhali a zejména pak:
Ing. Ivaně Suché za pomoc při extrakci dat ze staré databáze,
Milanu Strnadovi a Bc. Michalovi Hoškovi za pomoc s MSSQL
Zdeňkovi Layerovi za komentáře k mému úsilí s powershellem.
V neposlední řadě také mým rodičům a přátelům za jejich podporu.

Datové služby ve Windows

Data services in Windows

Souhrn

Tato práce popisuje problematiku a řešení přidělování přístupových práv ke sdíleným datům uživatelům v prostředí nadnárodní firmy. Jako hlavní prostředek přidělení práv je použit systém Microsoft active directory a členství uživatelů ve skupinách. Práce popisuje proceduru schválení přístupu k datům, přidělení práv, popisuje strukturu, v jaké jsou data uložena, i automatický systém, jehož úkolem je sbírat informace o adresářích.

Summary

This thesis describes problem and solution of assigning access rights to shared data in the environment of international company. As main tool for assigning rights, the Microsoft Active Directory system is used, and the rights are granted through membership in groups. Thesis describes how are the data stored, procedure for approving access to data, and automatic system, which task is to gather information about data.

Klíčová slova: Microsoft Active Directory, sdílený adresář, přidělení přístupu, schvalovací proces, SAN, NAS, SMB, NTFS

Keywords: Microsoft Active Directory, shared folder, granting an access, approval procedure, SAN, NAS, SMB, NTFS.

Obsah

1. Úvod

Od doby vzniku informačních technologií nacházejí tyto technologie široké uplatnění nejen na univerzitách a v armádě, kde vznikly, ale i v prostředí obchodu. Informační technologie dává podnikům do ruky dosud nevídaný nástroj, který jim umožnuje vyřizovat veškerou vnitřní administrativu elektronicky, ale hlavně napříč celým světem. V době internetu nehrají vzdálenosti až tak velkou roli a je zcela normální, že pro společnost v Americe dělá účetnictví specializované oddělení například v Belgii. Se vzrůstajícími komunikačními možnostmi, a možnostmi ukládat větší množství dat, ale přichází informační exploze. Množství uložených dat se neustále zvětšuje a je čím dále obtížnější v nich udržet řád. Firmám a podnikům je vlastní snaha o snižování nákladů a centralizaci služeb, to vede ke snaze přesouvat datová úložiště na jedno místo a k vytváření datových skladů. Spolu s daty se často přesouvá i technická podpora hardwarové základny, to vede opětovně ke ztrátě znalostí o datech a jejich významu. Všechny tyto faktory, spolu s požadavky na zálohování a neustálou proměnnou vytváří obtížný problém správy dat.

2. Cíle práce a metodika

Cílem této práce je popsat problém správy sdílených dat, a navrhnout řešení problému ztráty informace o datech. Budou zde popsány technologie a nástroje používané ke skladování dat, stejně jako postupy a nástroje pro jejich správu, zejména pak adresářový systém firmy Microsoft, Active Directory, který je hlavním nástrojem používaným při přidělování práv pro přístup k datům. Budu zde analyzovat problém přidělování práv a ztráty znalosti o nich, zvláště při zřizování, zanikání a migraci nových adresářů a odchodech zodpovědných lidí. Budu se zabývat popisem jednotlivých součástí celého systému, zejména dedikovaných NAS(Network attached storage) serverů a Active directory systémem a zanalyzuji celý problém. Jedním z posuzovaných údajů bude i stránka finanční a úvaha nad efektivitou současného řešení. Na základě získaných údajů navrhnu algoritmus ke sběru dat o adresářích, přístupu k nim a jejich vlastnících. Součástí tohoto návrhu bude i implementace algoritmu spolu s databází pro uložení dat a procedura pro získávání dat, přidělování práv a přenosu zodpovědnosti za uložená data novým uživatelům. Speciální pozornost bude věnována specifickým problémům identifikovaným při vývoji a jejich příčinám. Vedlejším cílem celého projektu je příprava báze dat pro případnou automatizaci celého systému. Výsledná implementace bude posouzena z hlediska výkonu, zejména doby sklízení dat, a stability.

3. Přehled řešené problematiky

3.1. Základní specifikace

Problém, jehož řešením se zabývám, úzce souvisí s mou prací, a proto ji nejdříve popíšu a představím se. Pracuji jako správce serverů se systémem Windows (verze NT4, 2000, 2003, 2008, 2008R2), ve firmě DHL IT Services. K povinnosti mé a mých kolegů patří i přidělování přístupových práv na sdílené adresáře pro jednotlivé uživatele. Je to dáno jednak bezpečnostní politikou a fyzickým umístěním jednotlivých datových skladišť, za druhé způsobem jakým jsou práva přidělována, pomocí členství ve skupinách v Microsoft active directory, jehož správa, na určité úrovni také patří k mým povinnostem. Z bezpečnostních důvodů není možné tuto povinnost přenést na helpdesk či jiný tým nižší instance, což by výrazně snížilo náklady. Zároveň je tento úkol značně rutinní a nevyžaduje příliš velké znalosti a schopnosti. Vyžaduje však poměrně velké množství, v našem případě dobře placené, práce. Navíc se jedná o jednu z nejčastějších žádostí, se kterou se setkáváme. Navíc firma má tendenci postupně celou svou strukturu podpory centralizovat a migrovat jak podporu, tak poskytované služby, i s hardwarem do centrály v Praze. Postupně tedy přebíráme práci původních oddělení z jednotlivých zemí, často bez jakékoliv dokumentace, v různorodé podobě, postrádající jakoukoliv standardizaci. Původní oddělení jsou rušena a zaměstnanci propouštěni a to vede k nárůstu problémů pocházejících ze ztráty znalostí o datech. Snažili jsme se samozřejmě všechny přenesené datové struktury standardizovat, abychom si usnadnili práci, to ale často naráželo na problémy s trošku jinak popsaným kontraktem podpory a trošku jinačími požadavky ze strany našich zákazníků. Brzo vyvstala potřeba vytvořit nástroj, který by znalosti o datech shromažďoval, což je podstatou této práce.

3.2. Analýza požadavků a reálná situace v DHL

i) Historie

Situace, kterou se pokouším řešit v DHL je poněkud komplikovaná. Výsledkem akvizice několika různých společností a jejich připojení do DHL je infrastruktura mírně nejednotná, a přestože existuje snaha ji sjednotit, tj. převést na stejný hardware, a podporovat ji podle stejného modelu, ještě stále je možné narazit na nestandartní řešení. To co zde budu popisovat je obvyklé řešení.

DHL je dopravní společnost, je rozdělena na velké množství různých poboček, ať už se jedná o sklady či kanceláře. Nejtypičtější službou, kterou každá pobočka potřebuje je schopnost ukládat data na úložišti a tisknout dokumenty. Původně měla každá pobočka File&Print server. To se ale ukázalo být neúnosné po přenesení podpory do Prahy. Po přenesení podpory byla do datacentra v Praze přesunuta i infrastruktura. Jako centrální úložiště byly zvoleny NAS servery. Zároveň byla vytvořena i DFS(Distributed file system) struktura, zjednodušující správu. Uživatelé se stále dotazují jen na určité položky v DFS, které se ale po změnách provedených na serverech mohou nacházet na zcela jiném místě, samozřejmě za předpokladu že DFS odkaz je udržován aktuální. Samotné NAS servery jsou virtuální, je to praktické, protože je potřeba čas od času změnit velikosti jednotlivých disků, a to se dělá o něco snadněji na virtuálním stroji.

ii) Strategie přidělování práv

Samotná strategie přidělování práv je založena na A G DL P strategii[1]. Na samotný sdílený adresář jsou přiřazeny tři doménově lokální skupiny. Jejich jméno je dáno jmennou konvencí, a popisuje, že se jedná o doménově lokální skupinu (D), která slouží ke sdílení souborů (F), a to buďto s čtecími, zápisovými, anebo úplnými právy (R,C,F), následuje zkratka země a stanice a popis adresáře.

Příklad:

DFCCZCHNO-LOGNL_Groups_Nijmegen3.

[1] Více ke způsobu přidělování práv je v další podkapitole Active Directory

Je skupina pro zápisová práva na adresář patřící oddělení v Nijmegenu v divizi ogistics, v Holandsku a nachází se v Čechách na Chodově. Tato skupina bude mít vnořenou globální skupinu, kde počáteční písmeno G znamená globální:

GFCCZCHNO-LOGNL_Groups_Nijmegen3.

Do ní budou přiřazeni uživatelé, popřípadě další globální skupiny, náležící skupinám uživatelů, například jednotlivým oddělením.

Celá struktura může být ale o několik stupňů komplikovanější. Nový uživatel, náš ilustrativní příklad, se stane členem oddělení a spolu s tím jsou mu přidělena práva skrze vnořené skupiny na sílené adresáře patřící oddělení. Skrze skupiny na třetí úrovni, vnořené v těch vnořených dostane přístupy k aplikacím a skrze pátou úroveň ke sdíleným diskům těchto aplikací. Struktura sama o sobě je složitá. Kdyby byla neměnná, šlo by ji zdokumentovat a mít manuál,ale bohužel, není. Se změnami technologií a možností a se změnami ve struktuře firmy jsou vyžadovány změny, které jsou postupně vrstveny na sebe. Po dokončení projektu je projekt postupně zapomenut, jak se zúčastnění lidé přesouvají od projektu, až nikdo pořádně neví, jak celá nově vytvořená struktura funguje. Teoreticky má každý sdílený adresář svého vlastníka, někoho kdo má právo schválit přidání dalšího uživatele do skupiny zajišťující přístup. Jméno takového člověka je pak uvedeno ve skupině, která mu patří, v poli poznámka. Lidé ale také často mění svou funkci, nebo rovnou odcházejí z firmy a jen málokdy je stanovena náhrada. Pokud by šlo o malé množství údajů v lokální firmě s malou pobočkou, dalo by se vypátrat, kdo za co zodpovídá a komu náleží rozhodovací právo. Bohužel struktura nadnárodní firmy je natolik složitá a rozsáhlá, že ani nevím koho se mám zeptat, abych dostal odpověď. Ne všechna data jsou tak důležitá, aby bylo třeba situaci důkladně vyšetřit, ale existují výjimky, Hlavně v případě dat personálních oddělení, finančních záznamů anebo adresářů patřících vedoucím pracovníkům. Například neúmyslné poskytnutí přístupu k pracovním smlouvám je přinejmenším velmi nešťastné.

iii) Struktura podpory

Podpora práv na sdílených adresářích má naásledující předpoklad. Ten kdo ji poskytuje, musí být schopen vyšetřit, jakým způsobem jsou práva přidělena, tedy musí mít práva k přihlášení se na servery, kde data jsou, což bývají nejen centrální servery ale i servery v jednotlivých podporovaných zemích. Vyžaduje to také určité nutné znalosti a zkušenosti s touto problematikou a v neposlední řadě práva na AD(Active direcotry) objekty, skupiny, které práva poskytují, aby bylo možné někomu práva přidat. Jednou z možností je že tuto službu poskytuje helpdesk, to předpokládá poměrně vysoká práva a zkušenosti helpdesku. Vzhledem k tomu, že v naší firmě jsou povinnosti helpdesku hlavně zajištění komunikace s uživateli napříč Evropou, často v jejich rodných jazycích, nejsou kladeny příliš velké nároky na jejich technickou práci a ta je předána výš. Jako team lidí starajících se o AD, o většinu windowsových serverů a majících technické znalosti a zkušenosti jsme pro tuto službu logickou volbou. Role helpdesku je získat od uživatele nezbytné informace ke splnění jeho požadavku, přeložit je do angličtiny a vytvořit v elektronickém systému žádost. Bohužel dost často dochází k problémům v komunikaci a k nám přichází žádosti se špatnými, anebo neúplnými informacemi. Vzhledem k tomu že je nás mnohem méně, než agentů na helpdesku, a že nemluvíme jazyky jednotlivých zemí, nemůžeme si informace opatřovat přímo, a proto posíláme žádosti zpět k doplnění a snažíme se vysvětlit helpdesku co vlastně potřebujeme.

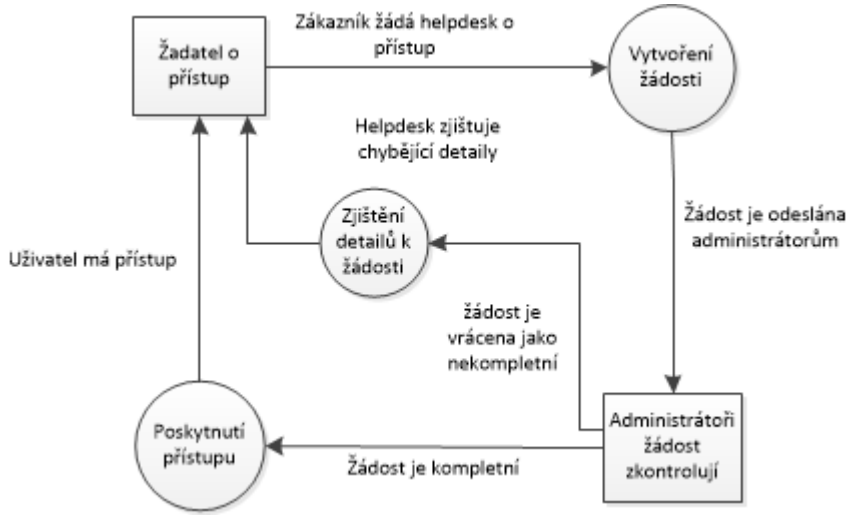


Diagram 3-1: Schéma putování žádosti

To se ne vždy povede, a proto dochází k frustraci všech zúčastněných stran, nás, protože nikdo nechápe, co vlastně chceme vědět, helpdesku protože mu stále vracíme ty samé žádosti, a hlavně koncového uživatelé, kterému pořád někdo telefonuje a nedokáže mu vysvětlit proč na práva na adresář, který nutně potřebuje k práci, čeká už čtyři dny a zatím si stále ještě musí kopírovat soubory od kolegů. Frustrace koncového uživatele, tedy našeho zákazníka, je pro nás to nejhorší. Účelem je tedy vytvořit systém, který umožní helpdesku lépe kontrolovat přijaté údaje, ideálně zároveň během hovoru se zákazníkem. Každá takováto žádost o přístup musí obsahovat lokaci kam má přístup být, a jeden ze základních tří druhů přístupů, které uživatelům obvykle přidělujeme, pro čtení, pro zápis, anebo plný přístup. Protože způsobů jak uživateli přístup přidělit, a skupin do kterých je možné uživatele dát, může být víc, je doporučeným vstupním údajem též vzorový uživatel. Žadateli je pak poskytnut ten samý druh přístupu.

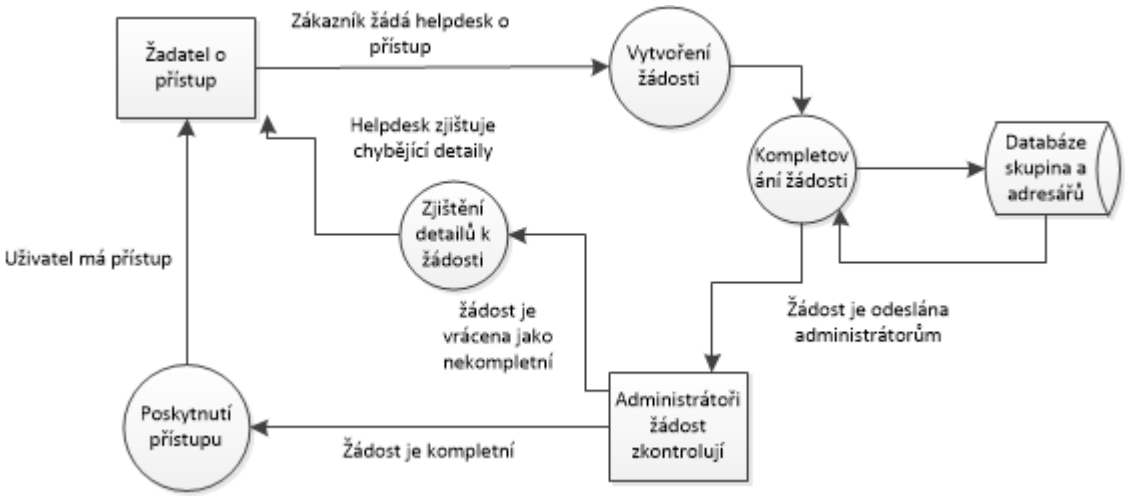


Diagram 3-2 Změna procesu

Nejlepší by bylo, aby takovýto systém umožňoval vytvoření automatické žádosti v našem systému. To je sice teoreticky možné, avšak vzhledem k momentálním změnám a výměně systému za nový, je nepravděpodobné, že by se podařilo oba systémy nějak propojit a zároveň udržet dlouhodobě kompatibilní. Systém musí být přístupný pro více uživatelů najednou. Systém musí být snadno ovladatelný, jeden ze způsobů žádání o přístup je email a systém by tedy měl být přístupný i pro uživatele a umožnit jim zjistit potřebné informace, takové, které by je samotné

nenapadlo zjistit. Jedno z možných řešení je automatické odesílání emailu, anebo automatické vygenerování textu, který uživatel do emailu vloží.

Specifikace řešení

Specifikace vycházející z existující struktury a technologií se kterými je nutné spolupracovat:

- Sbírat údaje o přístupových právech, jak z adresářů tak z active directory.
- Bude ukládat data ve srozumitelné podobě, takové, která umožnuje pozdější využití při případné automatizaci celého procesu, ideálně databázi.
- Bude moci běžet opakovaně a aktualizovat už existující databázi dat.
- Jeho použití se předpokládá hlavně na serverech v datacentru, měl by být ale použitelný i na servery, prostupnost sítě v tomto případě může být faktorem.
- Měl by být modifikovatelný kvůli možnému přidávání dalších funkcionalit, a snadno přenositelný, nezávislý na konkrétním serveru kde běží.
- Měl by běžet na serveru s operačním systémem Microsoft Windows 2008 R2
- Bude pracovat s Active directory ve verzi 2003, anebo vyšší

Bude vykonávat tři základní funkce, sběr dat, uchovávání dat a jejich vyhodnocování a zpřístupnění, každou z těchto rolí bude vykonávat jiná část zvlášť. Největšími omezeními při výběru použitých technologií je za prvé prostředí, které ve firmě existuje, a za druhé, použité technologie. Pokouším se udržet co největší míru kompatibility, a proto budu volit nástroje a programovací či skriptovací jazyky náležící k rodině firmy Microsoft.

- Část sběru dat.

  Bude realizována programem, ale kvůli dodatečným změnám spíše skriptem z rodiny Microsoft. Poběží periodicky kvůli aktualizaci dat. V úvahu připadají jazyky zaměřené na .NET, jako jsou například Powershell, v případě skriptování, a nebo C#, v případě programování.

- Část uchovávání dat.

  Bude realizována databází, kvůli kompatibilitě s rodinou produktů Microsoft, a protože databáze nemůže být čistě pasivní soubor, naopak, musí být přístupná neustále a z různých zdrojů, je jasnou volbou MSSQL. Vzhledem k tomu, že databáze nebude příliš velká, a ani příliš komplikovaná, stačí MSSQL v licenci express, která je sice velikostně omezená, ale je zdarma.

- Část zpřístupnění dat

  Účelem celého projektu je vytvořit systém, který umožní spravovat přístup k datům. Účelem poslední části bude tuto správu realizovat a využívat data z předchozích dvou bodů. Tato funkce bude realizována pomocí webového portálu, který umožní lidskému operátorovi získávat data z databáze, na základě určitých specifikací. Webový portál interagující s databází volím proto, že je to snadný způsob jak získat data a celý tento projekt zpřístupnit všem uživatelům. Obvyklé dotazy kladené portálu budou směřovat na požadované členství ve skupinách či na to kam má konkrétní uživatel přístup. Vzhledem k tomu, že celý projekt poběží na Windows 2008 R2, poběží celý webový portál na serveru IIS. Vzhledem ke kompatibilitě s ostatnímu částmi, často využívajícími .NET, bude i portál psaný v jazyce který ho využívá, ASP.NET

iv) Cena neefektivní komunikace.

Výše již byl popsán základní způsob odesílání žádosti, ale na to abych mohl ekonomicky zhodnotit tento projekt je potřeba zajít o něco hlouběji. Naše firma pro vytváření a evidování žádostí používá[2] systém *HP Openview Servicedesk*. Popsat celý systém by pravděpodobně vydalo víc než jen na jednu kapitolu, ve stručnosti jde o databázový systém, který v sobě obsahuje údaje žádostí všeho druhu, stejně jako údaje o volajících uživatelích, údaje o řešitelských teamech o serverech a mnoho dalšího, a zobrazuje je v podobě formulářů. Žádá-li něco uživatel, je na to

[2] V době vydání práce bude pravděpodobně vhodné říct používala

vytvořen formulář, který je odeslán na řešitelský team. V mnoha případech tento team není zdaleka sám a tiket putuje dále. Protože celková struktura podpory je složitá, málokdo ji zná celou a tak je jednou z povinností helpdesku přerozdělovat tikety na správné teamy a případně na požádání kontaktovat uživatele a vyžádat si od něj dodatečné informace[3]. Základní dělení těchto žádostí je na žádosti o změnu a incidenty, incidenty se vyznačují tím, že něco přestalo fungovat, žádosti o změnu prostě žádají změnu a za jejich úplnost zodpovídá zcela helpdesk. Žádost o úpravu přístupových práv je typickou žádostí o změnu. Typická je situace kdy někdo z administrátorů nějakou dobu zkoumá určitý problém, jen aby zjistil, že nemá dostatečné podklady k jeho vyřešení a vrátil ho zpět k doplnění[4]. Až se tiket vrátí, bude jej dost možná analyzovat někdo jiný a i on vynaloží toto množství práce. Jedná se řádově o několik málo desítek minut, ale vzhledem k velkému množství těchto žádostí a velké chybovosti, je množství ztracené práce nezanedbatelné. Navíc dochází k prodlevám a to vadí zákazníkům, kteří nedostávají službu, za kterou si zaplatili.

Celý projekt vyplynul z požadavků na náš administrátorský tým. Na konci roku 2011 a na začátku roku 2012 se prudce zvýšil počet žádostí o přístupy a předpokládalo se, že nároky budou nadále stoupat. Spolu se vzrůstajícím počtem žádostí vzrůstal i počet chyb a s tím opět nároky na administrátory. Jak zůstávaly žádosti nevyřízené, začaly se hromadit, až jsme zjistili, že v opravách a vracení nedostatečně vyplněných žádostí ztrácíme příliš mnoho času. Zde se pokusím vyčíslit cenu tohoto neefektivně vynaloženého úsilí.

Metodika:

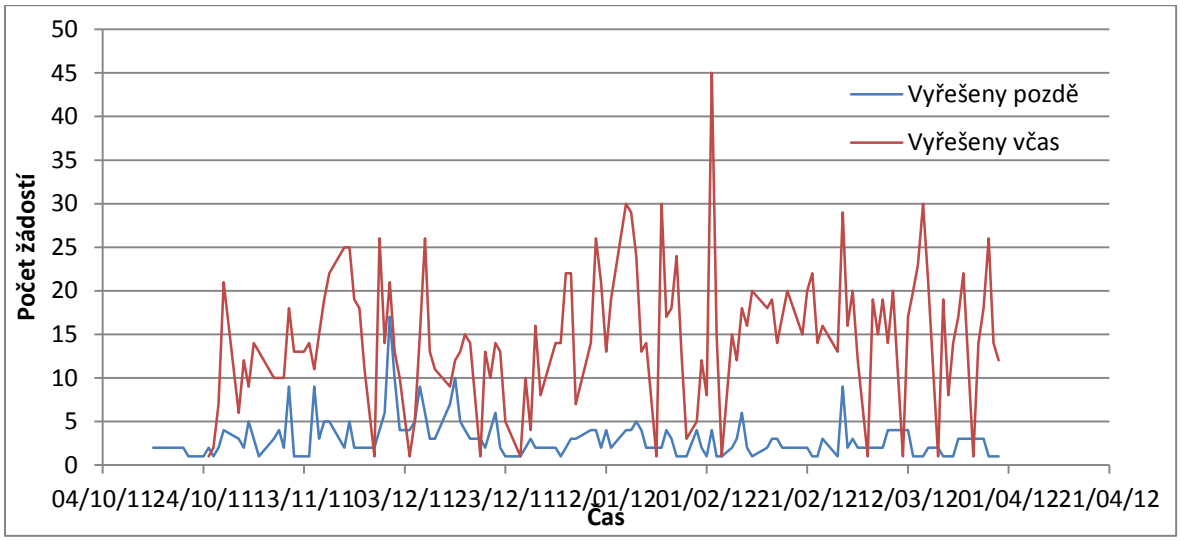
Data, která budu k vyčíslení používat, pochází z našeho starého a tehdy používaného systému žádosti HPSD, budu používat data z doby, kdy došlo k formulaci nároků na projekt[5] a krátce předtím, od prvního října 2011 do konce března 2012, protože tam by měly být nejvýraznější, a potřeba vytvoření nového systému se zdála největší. Ze všech žádostí jsou pro mě relevantní žádosti uzavřené

[3] Za určitých specifických podmínek.

[4] Což se může i opakovat

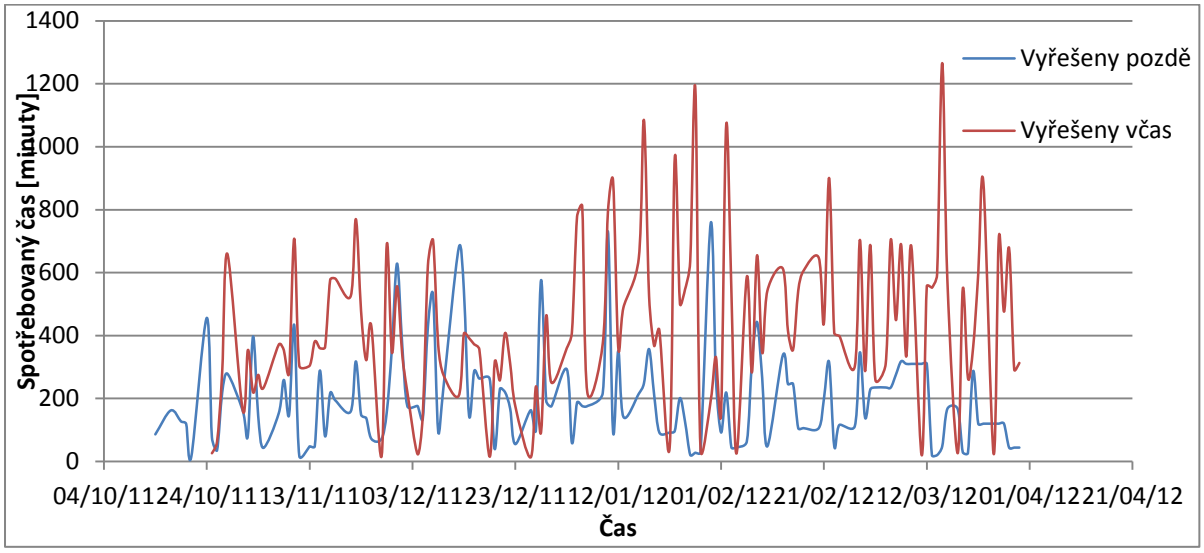
[5] Došlo k tomu v lednu 2012

s kódem „žádost o přístup“. Nepovinnou součástí každé žádosti bylo i pole, které nám umožnovalo identifikovat tikety ve kterých jsou nedostatečné informace a proto byly poslány zpět, bohužel zdaleka ne vždy bylo toto pole vyplněno a proto není množství těchto tiketů zcela vypovídající, je to však to nejlepší co je k dispozici. Tikety je rovněž nutné rozlišovat na ty, které byly vyřešeny včas a které ne. Množství tiketů nevyřešených včas a množství tiketů s nedostatečnou informací, a také spojení těchto dvou informací, je klíčovým ukazatelem. Další měřenou veličinou bude čas spotřebovaný na řešení, takto spotřebovaný čas totiž lze ocenit platem administrátora, který tento tiket řešil. Předpokládaný plat administrátora, ze kterého budu odvozovat cenu prodlevy, je zhruba 355 korun na hodinu.



Graf 3-1 Počet žádostí

Počet špatně vyplněných žádostí se nejeví nijak dramatický, v extrémních případech sice dosáhne na polovinu celkového počtu, jinak se ale drží nízko. Bohužel k vyřešení těchto žádostí je obvykle potřeba o dost větší množství času, a úsilí a několik průchodů skrze celý komunikační řetězec, jak ukazuje následující graf.



Graf 3-2 Vynaložené úsilí

Totéž vyplývá i z následující tabulky, která shrnuje počet žádostí, nákladů i času na ně spotřebované a z toho vyplývající finanční náklady. Je však nutné si uvědomit, že se jedná čistě o náklady za práci administrátora, náklady za hardware a infrastrukturu obecně, nejsou zahrnuty v součtu. Navíc se jedná čistě o náklady za provozování služby, nikoliv částku, která je účtována zákazníkovi.

	tikety pozdě	tikety včas	celkem
počet tiketů	339	1825	2164
celkový spotřebovaný čas [min]	22456	54332	76788
průměrný čas na tiket [min]	66.24	29.77	35.48
průměrná cena na tiket [Kč]	391.93	176.14	209.95
celková cena za období [Kč]	132 864.67	321 464.33	454 329

Zdaleka ne všechny tikety, které jsou zde prošlé, je možné připsat na vrub neefektivní komunikaci a nedostatečným informacím, bohužel administrátorské teamy, včetně toho našeho jsou v označování takovýchto tiketů značně nedůsledné a takto získané výsledky se nedají použít. Celkový počet lze jen odhadnout, a reálný odhad odpovídá zhruba třetině z tohoto počtu. Těžko lze také odhadnout efekt, který bude mít zavedení automatického systému na počet žádostí a o kolik se sníží nutné množství práce na ně. Pokud by byl počet nutné práce snížen na

polovinu u chybně zadaných tiketů, pak tento systém ušetří jednu šestinu celkových nákladů na vadné tikety, tedy zhruba 22144 korun, tedy 3690 korun měsíčně. Pravděpodobný přínos bude ale pravděpodobně větší, a projeví se i na zefektivnění práce na tiketech, které jsou momentálně vytvořeny správně. Vedlejším výstupem bude navíc i databáze mapující přístupy pro uživatele a usnadňující správu.

Následuje popis technologií používaných v naší firmě a technologií které budou použity při realizaci tohoto projektu.

3.3. Active Directory

S příchodem informačních systémů se objevila i potřeba jejich údržby a úprav. Tato potřeba postupně vzrůstala s rozšířením informačních systémů a s jejich vzájemnou propojeností. Postupně se objevila snaha o hromadnou správu zdrojů a identit, protože myšlenka samostatného nastavování na všech zúčastněných počítačích brzy přestala být reálná. Právě tyto potřeby vedly firmu Novell v roce 1993, k vytvoření adresářového systému Novell Directory services. Tento systém umožnuje centralizovanou správu uživatelských účtů politik a počítačů a systém Microsoft Active directory z něj vychází. Oba tyto systémy jsou v souladu s normou CCITT X. 500s (1988/1993)/ISO. (1)

i) Struktura Active Directory

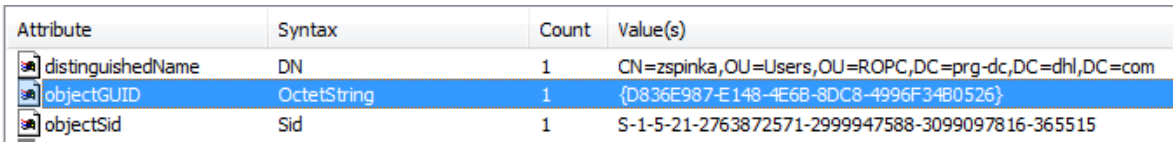
Active Directory je hierarchicky uspořádaná struktura mající charakter stromu. Její základní jednotkou je doména, ta může být sama podřazena jiné doméně, anebo být sama nejvýše postavena. S podřazenými a nadřazenými doménami tvoří les[6]. Mohou existovat i domény v lesech jiných než tento, které ale s touto doménou spolupracují, tedy mají zapnutou mezi doménovou důvěru, i když nejsou nutně součástí téhož lesa. Domény v témže lese mají typicky zapnutou oboustrannou důvěru, mezi doménová důvěra umožnuje přenos práv a vzájemnou autentizaci mezi jednotlivými doménami, byť s určitými omezeními. Doména samotná je rozdělena na organizační jednotky (organization unit, OU), adresáře připomínající kontejnery, ve kterých jsou uloženy jednotlivé objekty domény. Na organizační

[6] Oficiálním termínem je les, přestože z hlediska teorie grafů se jedná spíše o strom

jednotky, respektive na objekty v nich, se pak aplikují jednotlivé politiky. Existují i politiky, jejichž platnost je dána na celé doméně, například politika popisující požadavky na uživatelská hesla. (2)

ii) Objekty v Active directory

Active directory obsahuje různé typy objektů, uživatelské účty, politiky, tiskárny, skupiny aj. Každý objekt, nehledě na svůj typ, má definované své vlastnosti, jako je například jméno a každý objekt musí vždy mít 2 vlastnosti, bez kterých nemůže existovat. Svoje distinguished name a svoje GUID. GUID je unikátní v doméně a jednoznačně identifikuje objekt. V jedné doméně AD nemůžou být 2 objekty se stejným GUID, distinguished name popisuje objekt a jeho umístění v AD a doménu ve které se nachází.. Umístění je vlastnost, která určuje, jaké politiky budou uplatněny na objekt, protože politiky bývají aplikovány na celé OU.



Attribute	Syntax	Count	Value(s)
distinguishedName	DN	1	CN=zspinka,OU=Users,OU=ROPC,DC=prg-dc,DC=dhl,DC=com
objectGUID	OctetString	1	{D836E987-E148-4E6B-8DC8-4996F34B0526}
objectSid	Sid	1	S-1-5-21-2763872571-2999947588-3099097816-365515

Obrázek 3-3 GUID

iii) Security Principals

AD rozlišuje objekty na dva druhy. První druh jsou zdroje, může se jednat o tiskové fronty, aplikace, sdílené adresáře či něco jiného. Druhý druh objektů jsou takzvané security principals, jedná se o 3 typy objektů: uživatelské účty, účty počítačů a skupiny. Tyto objekty mají kromě GUID i svoje SID, *security identifier* (viz obrázek 3.1.3), které je unikátní nejen v doméně, ale mělo by být unikátní na celém světě. SID se skládá ze 48 bitového identifikačního čísla autority a 32 bitových hodnot podautorit a relativního identifikačního čísla. Příklad SID:

- S-15-21-2763872571-2999947588-3099097816-365515
- Stupeň revize: 15
- identifikační číslo autority: 21
- Doménový identifikátor: 2763872571-2999947588-3099097816
- Relativní identifikátor: 365515

S značí že se jedná o SID, identifikační číslo autority je předem definované, stejně jako stupeň revize. Doménový identifikátor, skládající se z jednotlivých podautorit je ale velmi pečlivě náhodně generován a jeho délka zaručuje, že je velmi nepravděpodobné, že někde na světě budou existovat domény se stejným číslem. Navíc existuje globální index všech domén, kde jsou nové domény nově registrovány. Relativní identifikátor je přiřazen určitému objektu. Existují předem definované identifikátory pro specifické objekty (doménoví administrátoři, účet Everyone atd.). Každý účet uživatele, každý účet počítače a každá skupina mají svoje SID. Z těchto čísel je poté, při autentizaci účtu,vytvořen Kerberos tiket, který popisuje práva účtu. (3) (4)

iv) Skupiny a členství v nich

Členství ve skupinách je klíčovou součástí přidělování práv v doménových systémech. Platí, že práva a politiky přiřazené na delegační skupinu budou poskytnuty, případně aplikovány na její členy, zároveň platí, že nejen účty ale i skupiny mohou být členy skupin. Těmto skupinám se pak říká vnořené (*nested*). Skupiny jsou rozdělené na globální, univerzální a doménově lokální, podle toho kde se nachází a kdo může být jejich členem. Další rozdělení skupin je na delegační a distribuční. Vzhledem k tomu, že distribuční skupiny slouží pouze k distribuci zpráv a nedelegují žádná práva, nás nezajímají, odteď když hovořím o skupině, mám na mysli delegační skupinu, pokud není řečeno jinak. Vlastnosti skupin závisí i na stupni funkcionality domény. Podle zadání se pohybujeme v doméně ve verzi 2003, native.

- Druhy skupin:
- Universal.

  Může být členem: domain local a nebo universal skupin z kterékoliv domény.

Může mít členy: uživatelské účty, global a universal skupiny z kterékoliv domény v lese.

- Domain local.
  Může mít členy: uživatelské a počítačové účty a globální a univerzální skupiny z jakékoliv domény v lese, anebo důvěryhodné domény. Domain local skupina z té samé domény.
  Může být členem pouze domain local skupin ve vlastní doméně
- Global
  Může mít členy: uživatelské a počítačové účty, globální skupiny z té samé domény.
  Může být členem: Univerzálních a domain local skupin, v jakékoliv důvěřující doméně a globálních skupin ve vlastní doméně.

v) Strategie přidělování práv skupinám v doménové struktuře.

Strategií na přidělování práv je několik, já zde zmíním jen 3 základní. Nejjednodušší strategie je nazývána A G P. Spočívá v přiřazení uživatelských účtů (A)[7] do globálních skupin (G) a přiřazení práv skupinám. Uživatelé potom dostanou svoje práva skrze dědění. Tento způsob má tu nevýhodu, že globální skupiny nejsou cachované na serverech, což znamená, že pokaždé, když uživatel žádá přístup k nějakému zdroji, je třeba jeho členství znovu ověřit. Další nevýhodou je že tento způsob lze jen těžko použít v systémech s více doménami. Výhodou tohoto přístupu je snadnost jeho údržby a provozování. Strategie A G DL P[8] je o něco složitější. Na rozdíl od předchozí strategie jsou práva přiřazena doménově lokálním skupinám, jejichž členy jsou poté globální skupiny. To znamená, že systém je o něco složitější vytvořit, ale jakmile je jednou vytvořen pořádně, jeho údržba je snazší, protože zásahů do AD není potřeba tolik. Poslední strategií, kterou jmenuji je A G U DL P[9]. Práva jsou přiřazena DL skupině. Jejím členem je příslušná univerzální skupina, která se může nacházet v kterékoliv

[7] Účty jsou v angličtině Accounts

[8] DL doménově lokální skupina

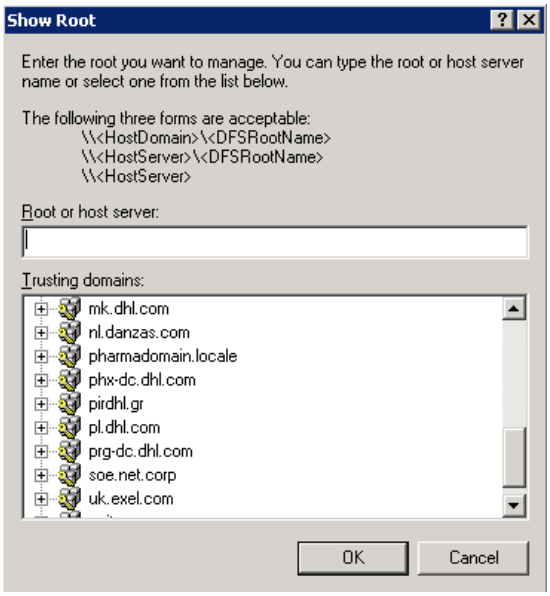
[9] U je univerzální skupina

doméně v lese, jejími členy jsou pak globální skupiny v jednotlivých doménách, naplněné uživateli.

3.4. Distributed file system

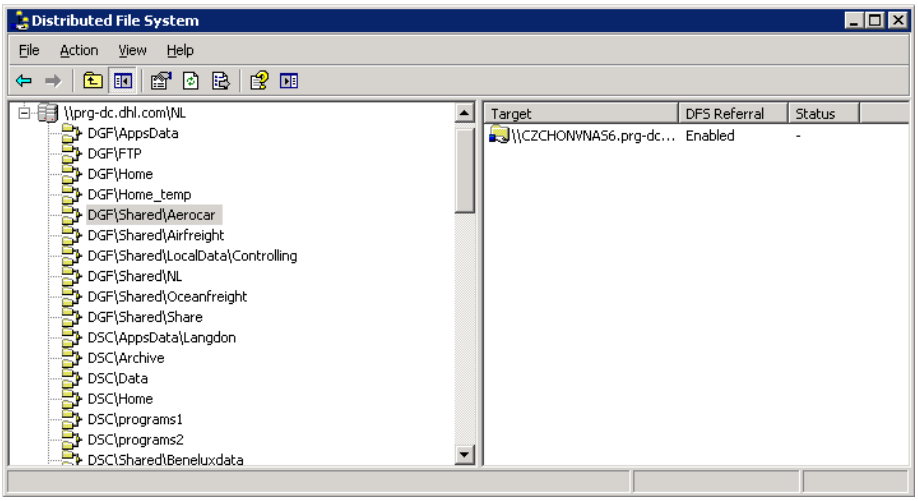
Distribuovaný souborový systému je služba implementovaná v systému Windows, která umožnuje správu sdílených adresářů podobným způsobem jako sdílené adresáře umožnují správu souborů. Poprvé se podobné služby objevily zároveň s příchodem NAS úložišť[10], kolem osmdesátých let minulého století. Vytváří strom virtuálních odkazů, které je možné procházet stejným způsobem jako například adresáře, nehledě na skutečné umístění dat. To vytváří pohodlnou situaci jak pro administrátora, tak pro uživatele. Uživatel přistupuje k jednotlivým datům na základě virtuální adresy a administrátor může data libovolně přenášet mezi disky, či úložišti a libovolně s nimi manipulovat, aniž by o tom uživatel věděl, anebo byl nějak ovlivněn. DFS existuje ve dvou verzích, za prvé, může existovat v lokálním jmenném prostoru počítače. Pak je to jen tento počítač, který je schopný využít služeb DFS. Anebo za druhé, může existovat jako služba na doméně Microsoft active directory, a být dostupná jako služba pro všechny počítače v lese domén. Druhá aplikace je ve velkých firmách z pochopitelných důvodů obvyklejší. Virtuální cesta, kterou uživatel v tomto případě používá, vypadá: \\<jméno domény>\<kořenDFS>.

[10] Network Attached Storage bude popsán v následující podkapitole



Obrázek 3-4 Stromy DFS

Role doménových řadičů v DFS je taková že pouze překládají odkazy. Uživatel zažádá o přístup k nějakému souboru v DFS, jeho uživatelská stanice, klient, komunikací pomocí protokolu CIFS obdrží odkaz na skutečné umístění souboru a na to už je přistoupeno přímo, obvykle pomocí NCP a nebo NFS (5) (6)



Obrázek 3-5 Kořeny DFS

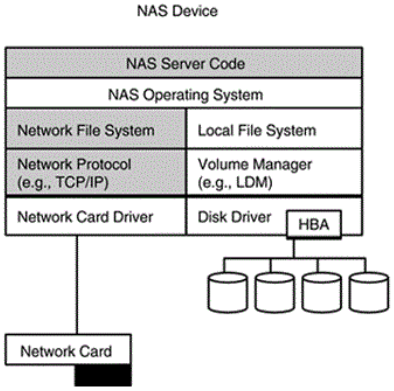
DFS se skládá z několika součástí, na serverové straně to je služba systému (Dfssvc.exe) a ovladače (Dfs.sys). Tato služba zodpovídá za exportování DFS topologie udržování je buďto v registrech, v případě lokálního použití DFS anebo

v Active Directory. Ovladač se stará o vyhledávání v DFS, když obdrží žádost od klienta. Na straně klienta je DFS implementováno dalším ovladačem (Dfsc.sys), který se stará o komunikaci s DFS serverem pomocí SMB přesměrování. Odtud také pochází jeden z limitů systému, lze jej použít jen na souborových systémech, které umožňují SMB. Další ovladač (Ntlanman.dll) pak realizuje přímo na klientovi samotnou službu. DFS obsahuje i DFS replikace (DFSR), díky ní může být jeden sdílený adresář uložen na více místech současně. DFSR se stará o replikace dat mezi úložišti. Zároveň slouží i k replikaci adresáře SYSVOL, který obsahuje přihlašovací skripty a soubory pro Group Policy. Vždy když dojde ke změně na jednom z úložišť, je změna propagována do všech ostatních. Propagována je pouze změna jako taková, nikoliv nový adresář či soubor, tím je vliv na zatížení sítě co nejmenší. Základním stavebním kamenem DFS je takzvaný *replica set*, což jsou dva, nebo víc, adresáře, respektive adresářové stromy, které se mezi sebou replikují, a to v závislosti na nastaveném časování a topologii, která definuje vztahy mezi jednotlivými domain controllery (DC). DFSR má bohužel svoje omezení, vzhledem k tomu že se, při detekci změn spoléhá na záznam změn souborového systému, lze jej použít jen na souborovém systému NTFS. DFSR jako takové nemá omezen počet úložišť. Teoreticky se jeden *replica set* může skládat z tisíců sdílených adresářů na tisících různých počítačů (kterým se pak říká *replica group*), propojených v různých topologiích (hvězda, kruh, řetěz atd.). Počítače navíc mohou být součástí více než jedné *replica group*. DFSR jako taková je implementována jako služba systému Windows (Dfsr.exe) a používá RPC autentizaci ke komunikaci mezi svými různými instancemi na různých počítačích. Vzhledem k tomu že Active directory má k dispozici i vlastní replikační nástroje (FRS, file replication service), které používá k replikaci SYSVOL, DFSR používá Active directory API ke stažení FRS konfigurace. DFSR má i WMI rozhraní pro management a konfiguraci. (6) (5) (4)

3.5. NAS

Network attached storage (NAS), v překladu síťové úložiště vzniklo jako koncept spolu s architekturou client-server v osmdesátých letech. Bylo výsledkem narůstající poptávky po úložných prostorech a zároveň dosažení limitů pro jednotlivé souborové

servery. Jejich limit byl dán limitem jejich řadičů a obvykle umožňoval připojit jen sedm medií. Síťové úložiště je řešením tohoto problému. Jedná se o samostatné zařízení, může být specializované, anebo jít o nějaký server s připojenými médii, který umožnuje přístup po síti ke svým diskovým oddílům a sdíleným adresářům.



. **Obrázek 3-6: Vnitřní struktura NAS**

Poskytnutí přístupu k NAS začíná síťovou žádostí na úrovni TCP/IP, dochází k autentizaci. Další je žádost o zápis do souborů či jejich čtení, zaslaná pomocí jednoho z protokolů síťových souborových systémů NFS nebo SMB/CIFS. Jakmile systém NASu obdrží tento požadavek, využije služby lokálního souborového systému, aby operaci provedl. Je v podstatě jedno, jestli NAS sám jako takový funguje na základě UNIXu, Windows, anebo svého vlastního operačního systému. Dalším řešením, které se historicky objevilo je SAN, Storage area Networks (*překlad*). Toto řešení se objevilo spolu s nástupem technologie Fiber channel (FC), která se pokouší kombinovat vlastnosti kanálu, a sítě. Nevýhodou klasické sítě je logika její komunikace a fakt, že většina z ní se odehrává na relativně vysokých úrovních. V případě FC existuje snaha rozšířit možnosti hardwaru a zároveň změnit logiku komunikace tak aby klesla na nižší úroveň. Výhodou je vysoká přenosová rychlost těchto zařízení, nevýhodou je vysoká cena při budováni infrastruktury. Z našeho pohledu je nejdůležitější, že zatímco NAS zpřístupňuje diskové oddíly a sdílené adresáře, SAN zpřístupňuje disky jako takové.

3.6. .NET

.NET byl vytvořen firmou Microsoft a je úzce svázán s Windows. Nejedná se jen o jeden produkt, jde o kombinaci několika věcí, a to knihoven funkcí .NETu, .NETové

jazyky, jako je například Visual Basic, C#, C++, ASP.NET engine, na kterém běží pro něj napsané webové stránky, editor na vývoj programů Visual Studio, a hlavně *The Common Language Runtime* (CLR), prostředí zajištující běh všech .NETových programů a starající se o paměť, optimalizaci, zabezpečení, komunikaci se systémem a mnoho dalšího.

i) The Common Language Runtime

Je asi nejdůležitější součástí .NETu, bez něj by nic nefungovalo. Obsahuje v sobě Compiler a Loader, a také optimalizační nástroje pro .NETové jazyky. Dále se stará o paměť, managment a garbage collector, a zajištuje zabezpečení. Když je napsán nějaký program v .NETovém jazyce. Je compilerem přeložen do jazyka nižší úrovně, takzvaného *Common Intermidiate Language* (CIL), výstupem je buďto spustitelný (exe) anebo knihovní soubor (dll). Je jedno který z jazyků .NETu je použit, všechny jsou obousměrně přeložitelné a všechny skončí přeloženy do stejně vypadajícího kódu CIL. K jeho spuštění je potřeba JIT (*just-in-time*) Compiler, který CIL interpretuje, a překládá do kódu nativního pro počítač, na kterém běží. Výsledkem tohoto přístupu je nezávislost takto napsaných programů na architektuře, na kterou budou použity. Pokud je na cílovém počítači CLR, ve správné verzi a se správnými knihovnami, pak tam dotyčný program poběží. (7)

ii) .NETové knihovny

Je sada předdefinovaných nástrojů v podobě tříd. Nejsou určené pro nějaký specifický účel, pokrývají potřeby pro vytváření desktopových aplikací, webových aplikací, malých skriptů na spravování a povšechně pokrývají potřeby vývojářů nehledě na obor. Jsou univerzální právě díky přenositelnosti CLR a použitelné v jakékoliv .NETovém jazyce. Pokud budu v této práci psát o .NETu mám na mysli použití jeho knihoven.

3.7. Web

Web začal vznikat na počátku devadesátých let, kdy Timothy John Brens-Lee navrhl jeho základ, jako hypertextový dokument s odkazy na jiné dokumenty a zobrazovaný klienty na základě klient-server architektury. Web měl podle něj mít schopnost uvědomit čtenáře o případném novém obsahu a umožnit čtenářům vkládat další příspěvky a dokumenty, čímž se smaže rozdíl mezi čtenářem a autorem. Tehdy byly položeny základy způsobu fungování webu, které se příliš nezměnily, a část jich přetrvává dodnes. Webová stránka byla tehdy statická a připomínala naformátovaný dokument textu, chyběly jí jakékoliv interaktivní prvky. Zaškrtávací políčka textové řádky a podobné komponenty přibyly až s HTML 2.0 v roce 1995, stále ale platilo, a do určité míry platí i dodnes, že stránka je zobrazována na požádání od klienta. Získat aktualizovaný obsah tedy vyžaduje načíst celou stránku znovu. Tehdejší stránky bychom stěží mohli nazvat aplikacemi, ty se objevily až později, a s nimi dva způsoby jak realizovat na webových stránkách aplikační logiku. (7)

iii) Klientská aplikace

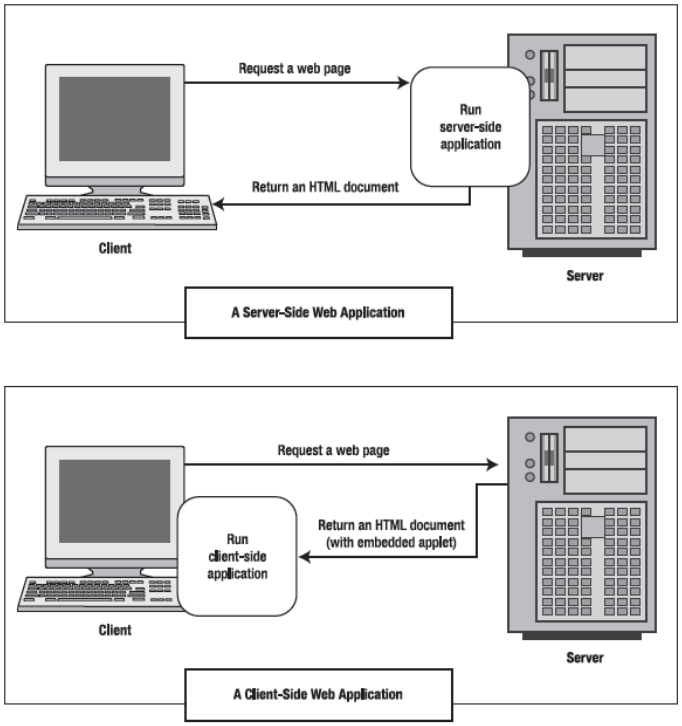
Přináší s sebou velké množství problémů. V závislosti na použité technologii je potřeba mít nainstalovaný příslušný software, ať už se jedná o Fash, Silverlight či Javu. Navíc je potřeba, aby stránky fungovaly ve všech prohlížečích, a zobrazovaly se stejně. To klade zvýšené nároky na programátora. Dalším aspektem je bezpečnost, je třeba si uvědomit, že takto je klientskému počítači poslán v podstatě kód celé aplikace, a pokud obsahuje nějaké bezpečnostní slabiny, bude snadné je najít. Přesto pokud jsou všechny tyto požadavky splněny, jde o funkční řešení, které se používá v mnoha situacích, například v případě internetových her. Hlavní výhodou je úspora výkonu na straně serveru a pro určité aplikace, jako například výše zmíněné hry,se takto místo výkonu na serveru, užívá výkonu na klientovi. (7)

.

iv) Serverová aplikace

Aplikace běžící na straně serveru funguje přesně opačným způsobem, místo aby poslala klientskému počítači kód, pošle mu pouze vygenerovanou stránku. Tím odpadá velká část problému s kompatibilitou mezi prohlížeči a s bezpečností, takto vygenerovaná stránka neobsahuje citlivé údaje a kusy kódu. Je to robustní a asi i

nejčastější řešení na internetu. Běžné použití je u menších internetových aplikací například bankovnictví a jiných, při kterých je potřeba pracovat s nějakými centrálními zdroji, typicky databází. Největší nevýhodou je opět to že běží na serveru a tím ho zatěžuje, server je potřeba dimenzovat vzhledem k předpokládanému počtu uživatelů. (7)



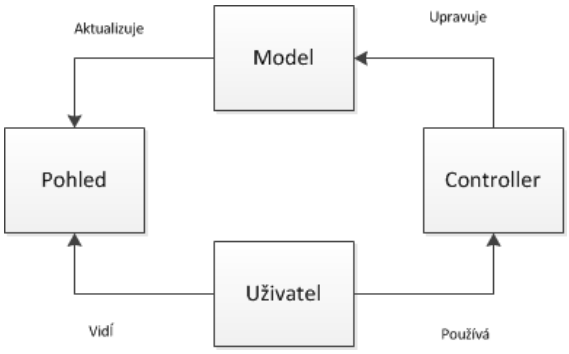
Obrázek 3-7 Klientská a serverová aplikace

Obě tyto koncepce ač velmi odlišné mají společné to, že při opětovné žádosti o stránku je načtena celá[11]. To znamená stažení velkého množství dat, v případě klientské aplikace, pokud nejsou už stažena a není možné je znovu použít, anebo nové vygenerování obsahu v případě serverové aplikace. To nehrálo v počátcích webu a do nedávné minulosti příliš velkou roli, dnes jsou ale stánky dělány mnohem větší, obvykle se složitou grafikou. To znamená, že i samotné nové načtení stránky muže způsobovat problémy, z důvodů zahlcení sítě. Vznikla tedy potřeba tento problém odstranit a s ní se objevil třetí koncept. (7)

[11]Stránky nejsou jen serverové a nebo klientské, obyklé je spojení těchto technologií.

v) MVC model

MVC je univerzální model který se nevztahuje jen na web. Odděluje od sebe data aplikace, pohled na aktuální data a řídící logiku. V angličtině jsou to výrazy *model, view* a *controller*, odtud MVC model. Starší architektura, takzvaný model 1 rozlišuje jen datový model a řídící logiku a neodděluje pohled. MVC model funguje zhruba takto: Uživatel provede nějakou akci. To zjistí controller z objektu uživatelského rozhraní, podle této akce zaktualizuje model. Pohled použije nově aktualizovaná data z modelu, pokud je to třeba, zcela nezávisle na modelu jako takovém, který je na pohledu nezávislý. Tento model byl poprvé použit v roce 1979 v jazyce Small talk, v souvislosti s Webem se typicky objevuje v ASP.NET MVC, což je zpracování založené na .NETu a v AJAXU, založeném na Javě. (7)



Obrázek 3-8 MVC model

3.8. ASP. NET

Je první interpreter serverových skriptů pro webové stránky od firmy Microsoft. Původně byl vydán jako doplněk pro IIS (Internet Information Services) server, což je prostředí používané k provozu webových stránek, ale dnes už je dodáván jako součást Windows. ASP umožnuje vkládat do HTML kódu webové stránky části některého z interpretovaných jazyků z rodiny jazyků .NETu. Ten je poté interpretován při zobrazení stránky. Původní verze ASP předcházela .NETu, ale další verze se už staly jeho součástí, mluví se tedy o klasickém ASP a ASP.NET. V této práci budu používat ASP. NET ve verzi 3. ASP je sám nejenom prostředím, ale v omezené míře i webovým jazykem. Je to jazyk objekty podporující, ovšem nikoliv objektově orientovaný, má

tedy několik předdefinovaných tříd, se kterými může pracovat, ale neumožnuje vytvářet nové a samozřejmě neumožnuje dědičnost, neumožnuje polymorfismus a tak dále. Toto omezení neplatí na jazyky používané jako součást stránek, ty naopak objektově orientované bývají a jejich schopnost využívat knihoven .NETu a tímto způsobem úzce spolupracovat s operačním systémem je jejich nejvýhodnější vlastností. Nevýhodou ASP je jeho nepřenositelnost na jiný operační systém. Podmínkou jeho fungování je totiž .NET a IIS. (7)

3.9. Powershell

Po dlouhou dobu byl „čistý“ přístup k serveru, typicky pouze prostřednictvím příkazové řádky, nedostupný pro Windows. Naopak vyskytoval se snad na všech ostatních operačních systémech, typicky na systémech založených na Unixu. Tento nedostatek se Microsoft pokusil dohnat v roce 2006 uvedením skriptovacího jazyka Powershell, který by byl objektově orientovaný a plně integrovaný do .NETu a stal se příkazovým shellem. Powershell původně nebyl pevnou součástí Windows, ale pro Windows 2003, tehdy nejaktuálnější, byl doinstalovatelný. Integrace přišla až s Windows 2008, bohužel tehdy v podobě už zaostalé verze 1, která neumožnovala ovládat celý systém. Postupný vývoj powershell, a to že s ním produkty Microsoftu počítaly, vedl k zajímavým paradoxním situacím. Bez ovládacího shellu bylo nutnou součástí každého produktu Microsoftu grafické rozhraní, pomocí kterého by šel produkt ovládat. Některé z těchto produktů s powershellem počítali natolik, že nebylo možné některé věci nastavit jinak než pomocí powershellu, to se stalo například u MSExchange, ovšem ještě v době před plnou integrací powershellu do operačního systému. Teoreticky tedy mohla existovat v některých ohledech nenastavitelná instance MSExchange, běžící na Windows 2003. Plné integrace jsme se dočkali až v roce 2009, s vydáním Microsoft Windows 2008 R2, kde byl každý komponent systému plně ovladatelný Powershellem a Powershell byl do systému plně integrován. To umožnilo vydání ´Windows 2008 Server Core, prvních Windows které jsou přístupné bez grafického rozhraní a tedy mnohem méně náročné na provoz. Powershell má ještě jedno specifické použití. Lze jej použít k vytvoření runspace pro aplikaci. Aplikační proces běží jako podprocess powershellu a má k dispozici Powershellové nástroje,

pipeline a commandlety. Skrze tento rodičovský proces je pak aplikace ovládána. Příkladem tohoto použití je například MS Exchange 2007. Momentální nejnovější verze powershellu je verze 3.0. Je dodávána jako součást Windows 8 a Windows Server 2012. (8)

3.10. C#

Je objektově a komponentově orientovaný programovací jazyk vyvinutý firmou Microsoft. Původní snahou při vývoji bylo napsat ekvivalent jazyku Java. Většina vývojářů ale programovala v C++, a tak se zrodil jazyk který syntaxí sice C++ připomíná, ale svou logikou a způsobem vkládání jednotlivých komponentů připomíná spíše Javu. Jazyk sám byl vyvinut paralelně s .NETem a v roce 2000. .NETové knihovny jsou jeho nedílnou součástí a bez nich je nemožné ho použít. Kód napsaný v C# je přeložen do *Common Intermediate Language* a stejně jako každý jiný .NETový jazyk, je poté interpretován pomocí CLR, to mu zajišťuje vysokou přenositelnost napříč platformami. V dnešní době se C# stal téměř univerzálním jazykem pro programování pro Windows a všechny nové produkty společnosti Microsoft jsou psány z větší části v něm. První verze C# byla vydána v roce 2002 a schválena v rámci ISO/IEC 23270:2003, poslední normovaná verze, je verze 2.0 v rámci ISO/IEC 23270:2006. Nejnovější verze C# je 5.0, která ale není normovaná. (7)

4. Vlastní řešení

4.1. Vybudování zázemí

Už od počátku bylo jasné, že tento projekt, pokud má být úspěšně dokončen, nemůže běžet občas na stolním počítači, ale musí sbírat data a aktualizovat databázi periodicky, a samotná databáze musí být stále přístupná. To si vyžadovalo server, na kterém bude celý projekt hostován. Od svého vedoucího jsem dostal svolení využít server náležící našemu teamu, operující pod systémem Microsoft Windows 2008 R2, a to jak ke spouštění samotného skriptu, tak k hostování databáze. Vzhledem k tomu, že samotná databáze je poměrně jednoduchá (viz návrh databáze) postačí pro ni MSSQL server

Express, který je zdarma. MSSQL volím kvůli kompatibilitě s powershellem, který budu používat. Zároveň operační systém Windows 2008 R2, je příhodný právě díky verzi Powershellu a .NETu, která je součástí systému. Přesto bylo nutné na server nainstalovat několik balíčků do powershellu, pro práci s SQL serverem, konktrétně *SQL Server provider and Database engine cmdlets* a *SQL Server Master Data cmdlets.*

4.2. Návrh Databáze

i) Součástí práce je i návrh databáze pro uložení získaných údajů. Uvažoval jsem nad několika možnostmi, jednou z nich bylo i uložení dat do csv souboru, ale předpokládaná velikost souboru, množství získaných údajů, a nutnost mít údaje stále k dispozici, stejně jako obtížnost vyhledávání v příliš velkém csv souboru vedla k tomu, že jsem toto řešení nakonec zavrhl, a rozhodl se vytvořit pro uložení údajů databázi. Ta se navíc může hodit pro další rozvoj projektu.

Je potřeba uložit celkem tři druhy údajů.

- Sdílený adresář a plnou cestu k němu.
- Skupinu poskytující přístup k adresáři, její distinguished name, a druh přístupu.
- Majitele adresáře, emailový kontakt a jméno.

Tyto prvky jsou mezi sebou řetězově provázány, a mají N:N vazbu. Sdílený adresář může mít přiřazeno více skupin než jen jednu, vlastně se to očekává, vzhledem k DHL standardům. Skupiny mezi sebou mohou být vnořené ale i vnořené skupiny budou poskytovat práva na tentýž adresář, proto každá bude v databázi mít vazbu na původní adresář. Krom toho může jedna skupina poskytovat přístup na více adresářů. Jedna skupina může mít také více než jednoho vlastníka a jeden vlastník může vlastnit i více než jednu skupinu. V 1NF vypadá návrh tedy takto:

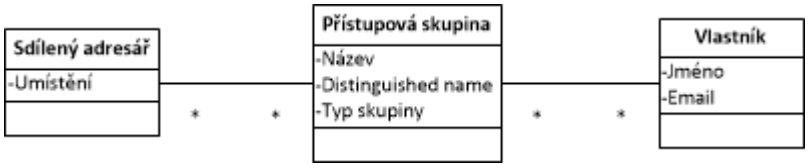
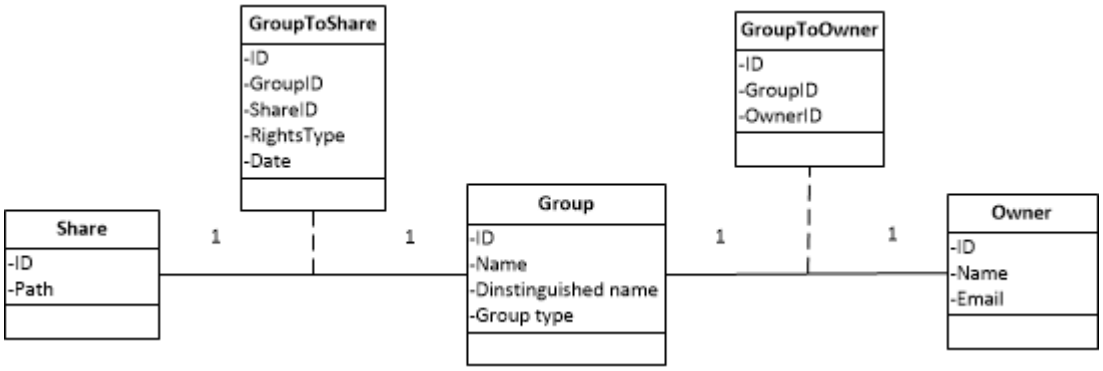


Diagram 4-1 1NF databáze

Toto je jen základní návrh, je třeba přidat klíče a dekomponovat vazby. V 3NF vpadá tedy databáze takto:



. **Diagram 4-2 3NF databáze**

Přibyly 2 tabulky, které dekomponují vazby, a spolu s nimi primární a cizí klíče. Dále byla nadefinována omezení, vynucující mazání dekomponujících tabulek, tak aby nemohla existovat tabulka, která má jen jednu nebo žádnou vazbu. Z pohledu databáze je 5 tabulek ještě stále velmi jednoduchá databáze. Bohužel algoritmus jejího vytvoření to poněkud zkomplikovalo.

4.3. Vývoj algoritmu

Bylo třeba vytvořit algoritmus, který by splnil specifikace uvedené v úvodu do řešené problematiky. Na první pohled se zdála úloha být jednoduchá, obsahující pouze cyklické prohledávání a záznamy do databáze. Rozdělil jsem ji tedy na několik částí a postupně se pokusil nastudovat ty, co neumím. Skript musí umět:

- Připojit se na vzdálený adresář a získat informace o přístupových právech. Prohledávat všechny vnořené adresáře a získávat tytéž informace.
- Připojit se k AD a získat odtamtud informace o jednotlivých skupinách Prohledat mnohdy nesourodé poznámky ke skupinám a získat z nich identitu vlastníka. Získat jeho email. Totéž udělat pro všechny vnořené skupiny.
- Připojit se na databázi, ověřovat jestli je třeba záznamy vytvořit, tedy jestli skupiny, vlastníci anebo sdílené adresáře existují. Vytváření nových záznamů případně rozpoznávání existujících.

První bod je jednoduchý, powershell sám o sobě obsahuje cmdlety které umožnují prohledávat souborový systém přímo, vzhledem k tomu, že powershell používá .NET, je maximální hloubka vnoření omezena délkou cesty. .NETové funkce umožnují totiž přistoupit na cestu dlouhou maximálně 259 znaků. Toto omezení se zdá být velmi zvláštní, vzhledem k tomu, že .NET stojí na funkcích win32API, které samotné dovolují přistoupit na cestu dlouhou 32767 znaků. Bohužel win32API není pro programátory a vývojáře obecně zdaleka tak přátelské jako .NET. Jakákoliv dokumentace se špatně vyhledává a jeho jednotlivé funkce se, naprosto nečekaně, mění. Zajistit funkčnost a stabilitu nějakého projektu je v tomto prostředí ještě obtížnější než ho realizovat, a tak jsem se rozhodl, že vzhledem k tomu že mou zodpovědností je monitorovat adresáře do třetího vnoření, prostě toto omezení připustím a ošetřím a nebudu se ho pokoušet obejít.

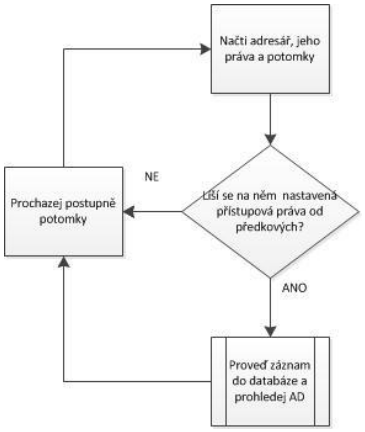
Druhý bod je technicky o něco složitější, ale vzhledem ke své práci mám s danou problematikou již nějaké zkušenosti. Obecně při přečtení práv na adresáři může dojít k jedné ze tří situací. Na adresáři se nachází Security principal z vlastní domény, nebo se na adresáři nachází SP z nevlastní, ale známé a důvěřované domény, anebo se tam nachází SP z neznámé a nedůvěřované domény. První případ je standartní případ, druhý je nestandartní, ale možný, nastává v případech, kdy na sdílený adresář přistupují uživatelé z jiných domén, a tedy jiných částí korporace. Poslední případ nastává v případě, že na sdíleném adresáři zbydou nastavena nějaká práva po SP co již přestal existovat, anebo pokud se jedná o nastavení pomocí SID history. V prvních dvou případech získám jméno skupiny i domény a ke získání detailů o něm použiji funkce .NETu. V třetím případě je k adresáři přiřazenou jenom SID a to je potřeba přeložit. I na to obsahuje .NET funkce, ale vzhledem k tomu, že se jedná o neznámou doménu, je její úspěšnost poměrně malá. V případě že se nepodaří SP identifikovat, případně pokud se nepodaří získat z příslušné domény nutné detaily, provede skript zápis do souboru, který zaznamenává průběh.

Třetí část, práce s databází, byla nejsložitější. V powershellu je mnoho způsobů jak pracovat s databází, já zvolil ten, který se mi zdá nejčistší, tedy transakční odesílání SQL příkazů databázi. Samotná práce s databází s sebou nese tu komplikaci, že se jedná o jednu strukturu provázanou podmínkami, které zaručují, že neexistují žádné volné vazby, že každá skupina má přiřazen adresář ke kterému patří a každý majitel

skupinu kterou vlastní, to je nutné zahrnout do samotného algoritmu prohledávání adresářů a skupin.

i) Algoritmus prohledávání souborů

Na první pohled se úloha jeví velmi jednoduše a nabízí jednoduché řešení v podobě rekurzivní konstrukce naznačené na následujícím diagramu. Bohužel tento postup funguje pouze na menší množství dat. V našem případě znamená rekurzivní přístup zaznamenání do paměti všech načtených vlastností z adresáře, včetně všech plných cest k datům, načtených přístupových práv a seznamu všech potomků. Krom toho i práv předka. Brzy po spuštění zabere takto napsaný skript většinu paměti a zcela samovolně se zhroutí. Po zjištění tohoto problém jsem, na základě předchozích zkušeností, algoritmus upravil na iterativní, vytvořil třídu, která bude uchovávat takto získaná data. Algoritmus je bude postupně ukládat do pole těchto tříd. Tím jsem převedl celý problém na o něco komplikovanější prohledávání do šířky. To sice problém s uloženými informacemi v paměti poněkud zjednodušilo, ale procházení pole a hledání konkrétních objektů, podle zaznamenaných indexů předků a potomků situaci stále zpomaluje. Je možné buďto změnit způsob uložení dat v paměti, nejspíše do podoby vyváženého stromu a tím usnadnit procházení, nebo změnit strategii prohledávání na prohledávání do hloubky, adekvátně upravit způsob uložení a tím snížit nejen pomalost procházení paměti, ale i celkové množství uložených informací. Následující diagramy ilustrují oba algoritmy, červená barva označuje adresář, jehož údaje jsou aktuálně uložena v paměti.



. **Diagram 4-3 Rekurzivní algoritmus**

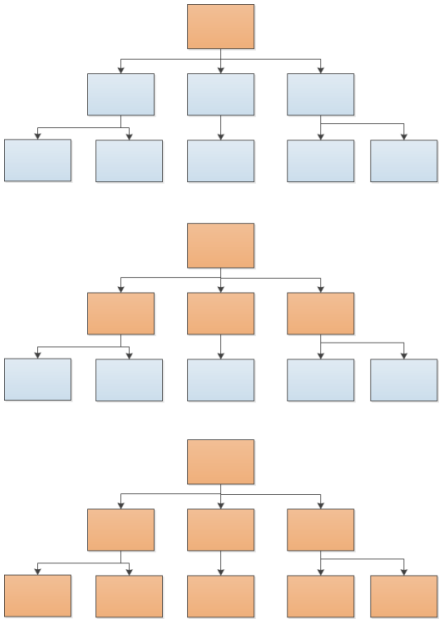


Diagram 4-4 Iterace se spojovým seznamem

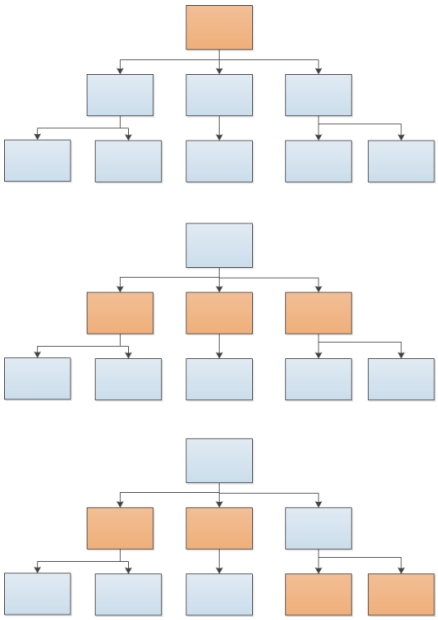


Diagram 4-5 Iterace se zásobníkem

První verze pracuje s polem tak, že vždycky zjistí vlastnosti adresáře, vyhodnotí ho, a poté ho expanduje. Do pole uloží všechny jeho potomky a jejich vlastnosti a pozici jejich předka i jeho potomků zaznamená do hash tabulky, a poté se posune k dalšímu prvku v poli na vyhodnocení. Toto chování by se v případě jazyků nižší úrovně realizovalo nejspíše spojovým seznamem. V powershellu funguje pole velmi podobně. Tento přístup vede k tomu, že jsou postupně vyhodnocovány prvky stále nižší úrovně, jako první jdou všechny prvky na první úrovni, dále všechny prvky na druhé úrovni a tak dále. Vzhledem k tomu že počet prvků se s klesající úrovní má tendenci zvyšovat, obsazená paměť pozvolna narůstá, k uvolnění začne docházet až po dosažení nejspodnější úrovně stromu. Tento způsob sice také zahlcuje paměť, ale množství uložených dat je mnohem menší než v případě rekurze, přesto procházení pole a hledání konkrétních prvků podle toho jak po sobě následují v souborové struktuře je zdlouhavé. Nejefektivnější se jeví třetí způsob. Místo spojového seznamu lze k uložení seznamu adresářů použít zásobník. V tomto případě odpadá nutnost vytvářet komplikovanou třídu k uložení informací o právě vyhodnocovaném prvku, protože najednou je vyhodnocován jen jeden prvek. Informace o něm jsou načítány přímo z něj a ukládány přímo do databáze. Z vrcholu zásobníku je vyzvednut prvek, přesněji plná cesta k nějakému adresáři.

Jsou nalezeny všechny jeho podadresáře a na nich se nacházející přístupová práva jsou srovnána s právy na aktuálním adresáři, a pokud se liší, jsou vloženy na vrchol zásobníku. Práva jsou na aktuálním prvku vyhodnocena, stejným způsobem, opět přes zásobník, je prozkoumáno i členství skupin umístěných na adresáři, ty jsou rozděleny podle druhu přístupu, který je pro snadnou správu rozdělen na přístup pro čtení, přístup pro zápis a plný přístup. V případě, že je zjištěna nesrovnalost, která je proti standardům DHL, je zaznamenána do logu o průběhu skriptu, aby mohla být později vyřešena.

ii) Prohledávání Active Directory

Prohledávání active directory je relativně blízké prohledávání stromové struktury souborového systému. Úlohou je opět nalézt vlastnosti nadřazeného prvku a poté provést totéž na všechny podřazené prvky. Jsou zde ale jisté rozdíly. Active Directory by sice měla být stromová struktura, ale to není vždy pravda. Může se stát, že nějaký prvek je dítětem svého dítěte. Pokud by algoritmus prohledával AD stejným způsobem, dojde k jeho zacyklení. Naštěstí je množství členu AD skupin výrazně menší a struktura je výrazně jednodušší než u souborového systému, a tudíž jde indexovat všechny členy a zabránit tak zacyklení. Samotný algoritmus prohledávání AD používá, stejně jako algoritmus prohledáváni souborového systému zásobník a prohledávání do hloubky. Vlastníci skupin mohou být uloženi v textu v popisu skupin. Hledané vlastnosti tedy jsou *members, note* a *description*. Z *note* a z *description* je načten text a pomocí regulárních výrazů zpracován k nalezení jména či emailu vlastníka. Bohužel protože systém byl dělán pro lidské operátory, neexistuje žádná pevně daná syntaxe, vyhledávám tedy obvykle používaný výraz *owner* a text co následuje za ním, z něj se pokouším odfiltrovat buďto jméno, anebo email a ověřit jej proti AD. Takto získané údaje, pokud možno oba, uložím do databáze. Jedním z budoucích projektů bude i standardizace popisů skupin, aby byly snáze čitelné jak pro člověka, tak pro stroj. Další komplikací při prohledávání může být členství ve skupinách napříč různými doménami. Platí, že členové domény jsou uloženy pouze na domain controllerech té domény do které skupina patří. Teoreticky je tedy třeba provádět skoky mezi jednotlivými doménami a střídavě se tázat jednotlivých domain kontrolérů. Protože vyhledávání

toho správného domain kontroloru podle názvu domény nějakou dobu trvá, vytváří si skript záznamy o doménách a adresách nejbližších, tedy nejrychleji odpovídajících, domain controlerů. Další možností jak může být přidělen přístup je skrze atribut SID history, kdy skupina má ještě druhé SID z jiné domény, které odpovídá jiné skupině a má skrze něj i její přístupová práva. Tento způsob, má své nevýhody, například prodlužování Kerberos tiketů a nepřehlednost, a je používán jen zřídka. Vzhledem k tomu že není považován za standard, není ze strany skriptu podporován. Skript nevyhledává tyto atributy a neanalyzuje je.

iii) Zápis do databáze

Pro zápis do databáze má powershell k dispozici funkce .NETu, který je s MSSQL plně kompatibilní. Používal jsem metody následujících tříd:

- System.Data.SqlClient.SqlConnection
- System.Data.SqlClient.Sqlcommand
- System.Data.SqlClient.SqlAdapter

Příklad vytvoření připojení do databáze:

```
$global:conn = New-Object
System.Data.SqlClient.SqlConnection("Server=**************;Database=AccessManager;
integrated security=SSPI")
$global:conn.Open()
$global:cmd = New-Object System.Data.SqlClient.SqlCommand
$global:cmd.connection = $global:conn
$global:SqlAdapter = New-Object System.Data.SqlClient.SqlDataAdapter
$global:SqlAdapter.SelectCommand = $global:cmd
```

Na začátku skriptu se naváže spojení s databází a to je udržováno po celou dobu. Jeho vytvoření trvá nějakou dobu, a je tedy nevýhodné ho vytvářet při každém záznamu do databáze. Skrze toto spojení probíhá komunikace v podobě textových příkazů v jazyce SQL, a protože jednotlivé příkazy jsou si pro jednotlivé tabulky velmi podobné, vytvořil jsem pro každou tabulku speciální funkci, která se stará o samotný zápis i určitou vyhodnocující logiku. Rozhoduje například mezi příkazy UPDATE a INSERT, na základě toho, jestli taková data v databázi již jsou a zda potřebují pozměnit či vymazat. Zároveň upravují vstupy v podobě názvů adresářů a skupin, kde se mohou vyskytovat znaky klíčové pro jazyk SQL. Následující diagram zevrubně popisuje celý algoritmus sběru dat. Je ochuzen o detaily

některých částí, jako jsou například algoritmy zápisu do databáze a ošetřování výjimek, které by ho příliš komplikovaly, předpokládá se že prostě fungují.

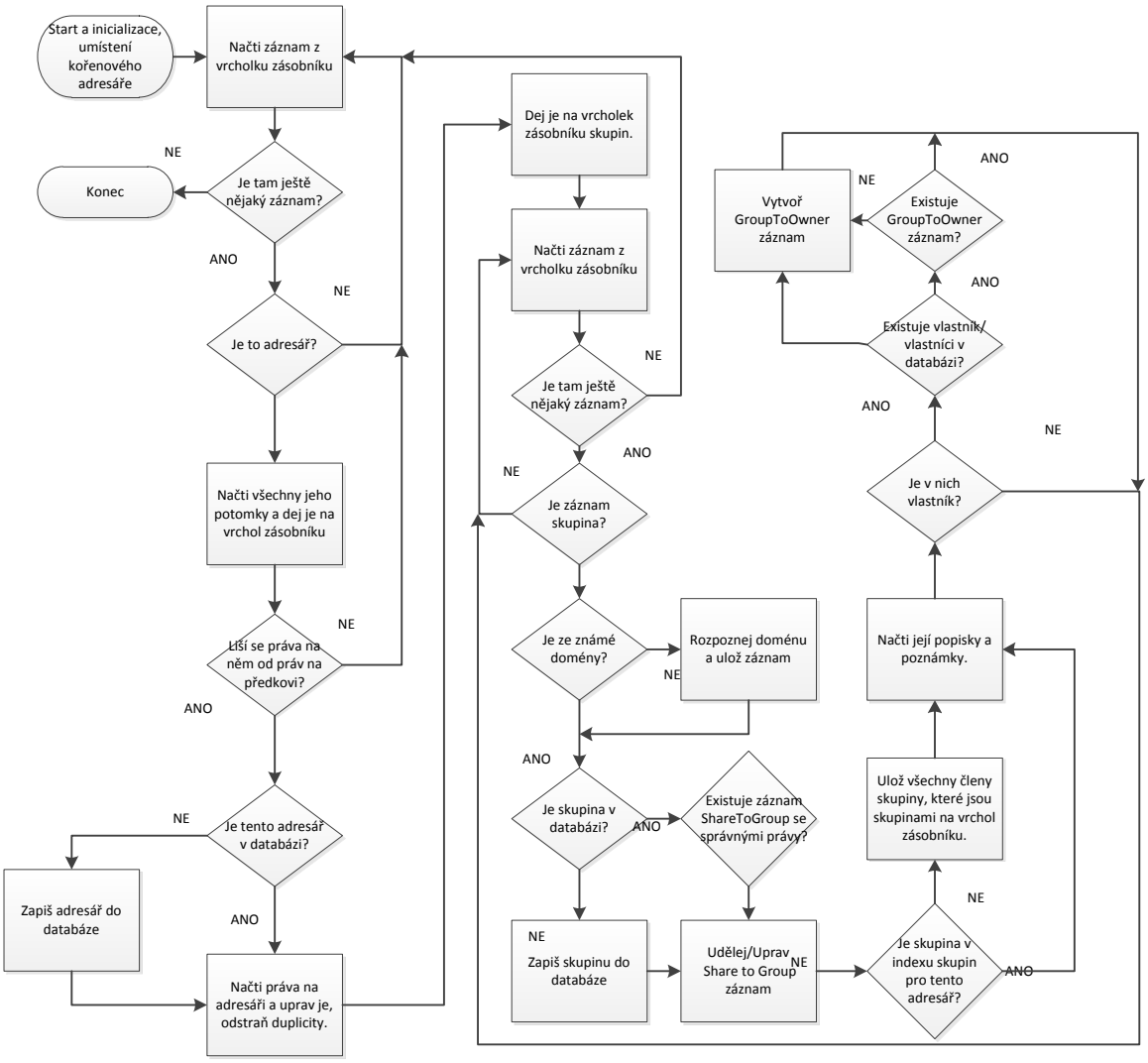


Diagram 4-1 Schéma prohledávacího algoritmu

4.4. Test sběru dat

První věc, co jsem potřeboval otestovat, byla rychlost algoritmu při sběru dat. Celý skript poběží na virtuálním serveru s následujícími parametry:

Systém: Windows 2008 R2 Enterprise Edition

Procesory: 2x Intel Xeon X7350 2.93 GHz

RAM :4GB

Síťová karta: Virtuální 1x 1Gbps.

Účel a metodika testu:

Účelem testu je vyzkoušet výkon a zhodnotit různé strategie při sběru dat, a najít nejlepší z nich. Při předpokládaném počtu objektů k prohledání sto milionů je důležité aby proledávání proběhlo dostatečně rychle, protože data se mohou v průběho sběru měnit. Tyto testy budou srovnávat různá řešení této problematiky. Je několik možností jak data sbírat, a to, který způsob je nejvýhodnější pravděpodobně záleží na místě jejich uložení. Pokud budou data uložena na počítači s jedním diskem, bude největším problémem rychlost jeho disku, zde použití několika vláken nebude velým přínosem, naopak několik vláken si může konkurovat a celý postup zpomalit. Naopak, jsou-li data uložena na diskovém poli, skládajícím se z mnoha disků a umožňujícím mnoho přístupů najednou, je jednovláknový přistup pomalý. Přestože skript je primárně učen k použití v tom samém datacentru, použití na větší vzdálenost může být potřeba. Pro takový případ je třeba otestovat vliv sítě na rychlost prohledávání. Zároveň, aby bylo možné zjistit nároky na vlastní, nepříliš výkonný server, poběží po celou dobu spuštění monitorování jeho zatížení. Všechny testy poběží v pozdních večerních hodinách, nejlépe mezi devátou hodinou a půlnocí, abych se vyhnul možnému zatížení pracovního dne, a zároveň zálohovacím procedurám, které jsou v provozu v brzkých ranních hodinách. Ve více vláknových testech poběží vlákna vždy s nejnižší možnou prioritou zaprvé, aby si nekonkurovala vzájemně, zadruhé aby nepřetížila celý server.

i) Test1

Je prováděn na produkčních datech, tedy datech reálných zákazníků

Je prováděn na diskovém poli v datacetnru, vliv sítě je tedy minimální a zároveň předpokládám výhodnost vícevláknového přístupu.

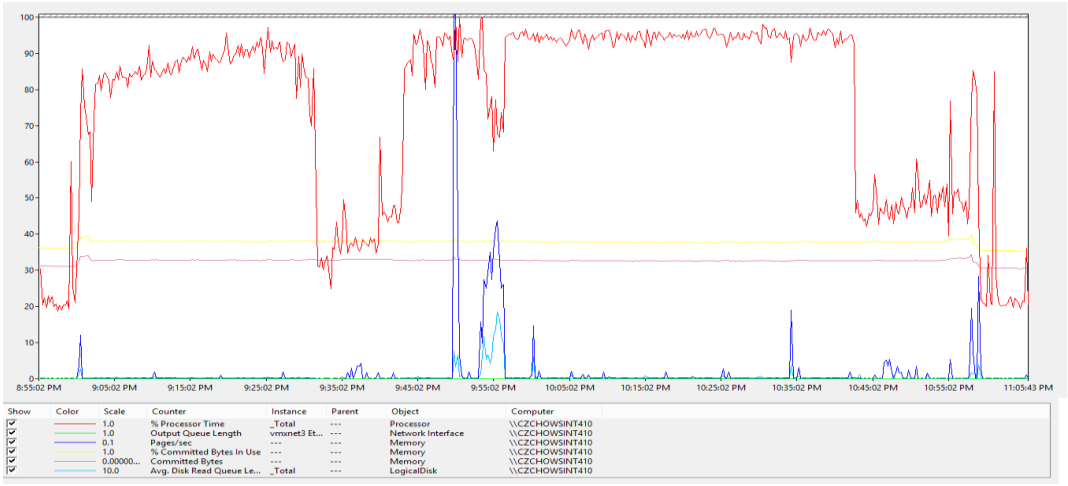
Test 1 jednovláknový	
zahájení	23:00:14
konec	23:50:26
Doba trvání	00:50:11
Počet prohledaných AD objektů	78291
Počet prohledaných souborů	75360
Počet prohledaných adresářů	8519

Zatížení serveru:

Graf 4-1Test 1, jednovláknový

Test 1 více vláknový			
vlákna	1	2	3
zahájení	21:00:25	21:00:25	21:00:26
konec	21:01:51	22:58:55	21:01:45
Doba trvání	00:01:25	01:58:29	00:01:19
Počet prohledaných AD objektů	119	80176	176
Počet prohledaných souborů	2365	74001	1939
Počet prohledaných adresářů	153	8473	123

Zatížení serveru v mnohovláknovém testu.



Graf 4-2 Test 1, více vláken

Zhodnocení testu:

Test ukázal, že prohledávání je nenáročné na paměť, tato část problematiky byla dobře ošetřena, ale je náročný na výkon procesoru, a to natolik, že i když běží vlákna při nejnižší možné prioritě, konkurují si.

ii) Test 2

Předchozí test nebyl příliš průkazný, protože reálná data jsou nesouměrně rozvrstvena a některá vlákna, v našem případě dvě ze tří, končí dříve než ostatní. Na diskovém poli, jsem tedy vytvořil, ideální testovací prostředí, rovnoměrně rozvrstvených dat, jedná se o testovací diskové pole, takže na denní době nezáleží, test nebude ovlivněn živým provozem. V testovacím adresáři se nachází vždy 10 adresářů. V každém z těchto adresářů je dalších deset adresářů a dvacet souborů s krátkým textovým řetězcem. V těchto adresářích jsou další adresáře a soubory atd. Celkem jsou čtyři stupně vnoření, tedy 222200 souborů a 11110 adresářů. Na každém stupni vnoření je navíc v bezpečnostním nastavení nastavena jiná skupina. Test poběží dvakrát, jednou s jedním a podruhé s deseti vlákny.

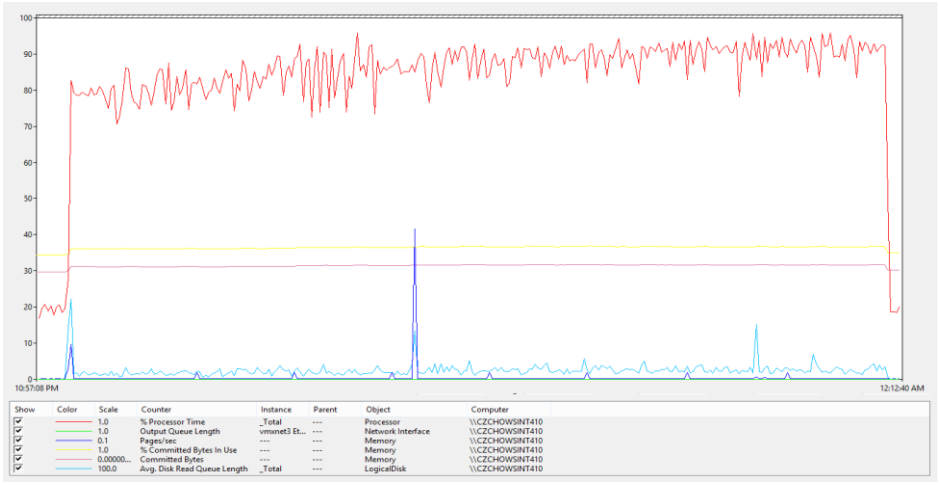
Test 2

Test 2 jednovláknový	
zahájení	23:00:12
konec	00:11:23
Doba trvání	01:11:11
Počet prohledaných AD objektů	33330
Počet prohledaných souborů	222200
Počet prohledaných adresářů	11110

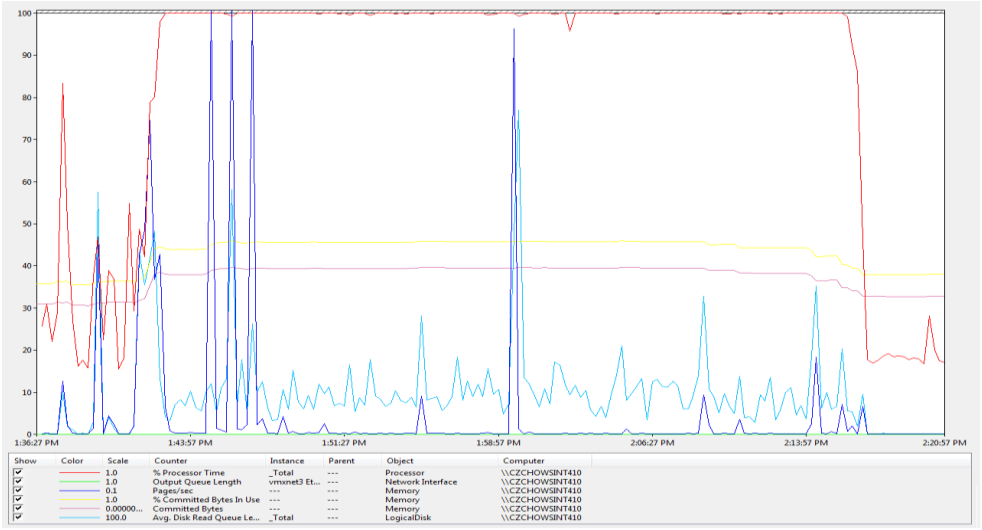
Test 2 více vláknový		
vlákna	Nejpomalejší	Nejrychlejší
zahájení	13:42:20	13:42:20
konec	14:16:39	14:08:56
Doba trvání	00:34:18	00:26:36
Počet prohledaných AD objektů	3330	3330

Počet prohledaných souborů	22220	22220
Počet prohledaných adresářů	1110	1110

Zatížení serveru:



Graf 4-3 Test 2, jedno vlákno



Graf 4-3 Test 2, více vláken

Zhodnocení testu:

Test běžel podle očekávání a prohledal celý souborový systém. Odchylky v délce běhu jednotlivých vláken, jsou způsobeny tím, že vlákna běží s nejmenší prioritou. Jsou tedy jinými vlákny patřícími jiným procesům odsouvána z procesoru a hladoví. Odchylka 8 minut mezi nejrychlejším a nejpomalejším vláknem je poměrně vysoká, vzhledem k celkové délce běhu. Krátké výkonové špičky v zatížení paměti serveru, jsou s nejvyšší pravděpodobností způsobeny jinými procesy. Rozdělením úlohy na deset podvláken se podařilo celkovou prohledávací dobu zkrátit na polovinu.

iii) Test 3

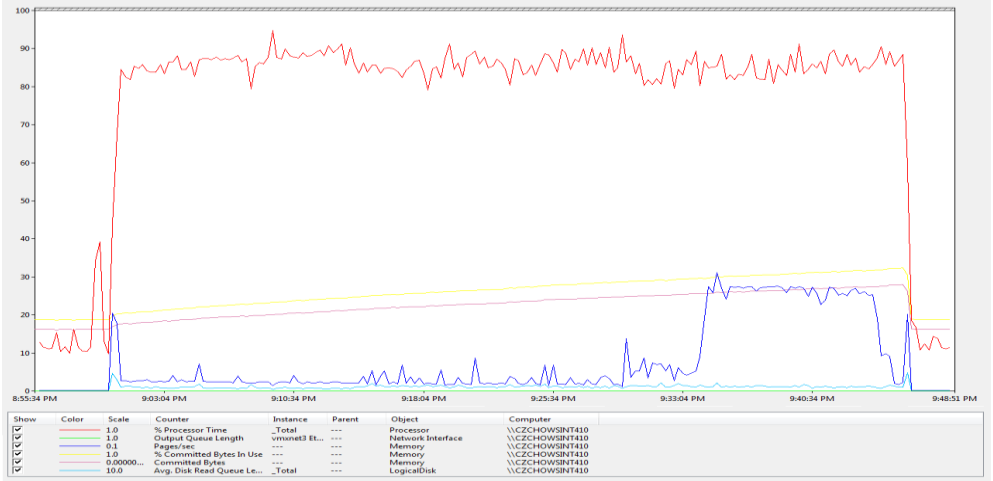
Test 3 provádím, abych si ověřil základní předpoklad, že mnohovláknový sběr dat nemá smysl na diskovém systému, který není schopen pojmout více vstupů najednou, bude zajímavé srovnat finální výsledky obou verzí testu a hledat případné zpomalení.

Test 3

Test 3 jednovláknový	
zahájení	21:00:16
konec	21:46:05
Doba trvání	00:45:49
Počet prohledaných AD objektů	55550
Počet prohledaných souborů	222200
Počet prohledaných adresářů	11110

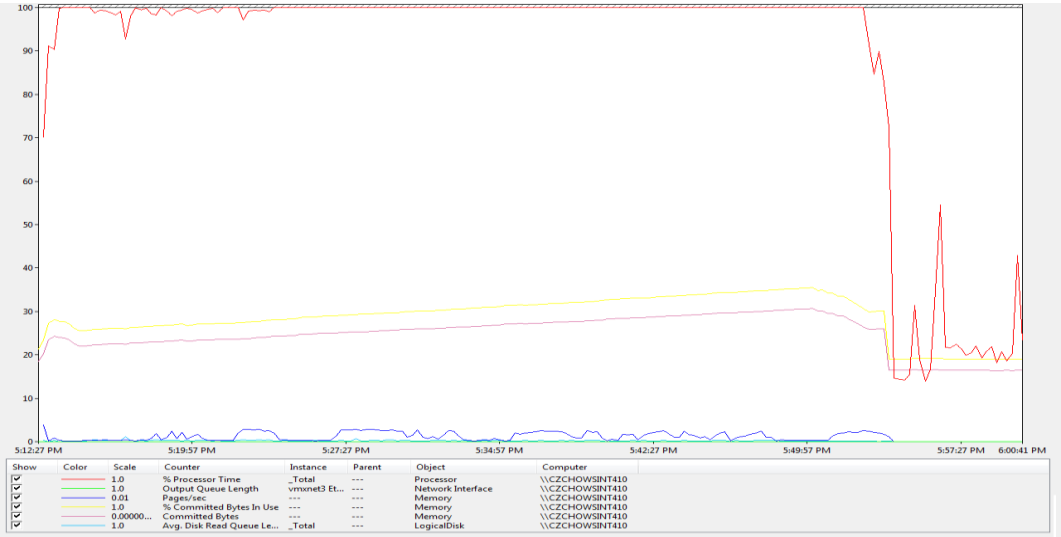
Test 3 více vláknový		
vlákna	Nejpomalejší	Nejrychlejší
zahájení	17:13:03	17:13:05
konec	17:53:52	17:51:16
Doba trvání	00:40:48	00:38:10
Počet prohledaných AD objektů	5550	5550
Počet prohledaných souborů	22220	22220
Počet prohledaných adresářů	1110	1110

Zatížení serveru.



Graf 4-4 Test 3, 1 vlákno

Zatížení serveru



Graf 4-5 Test 3, vícevláknový

Test 3 Zhodnocení.

Hypotézu se nepodařilo potvrdit, přestože byl mnohovláknový test pomalejší, než na diskovém poli uzpůsobeným k větším zátěžím. Ještě stále bylo výhodné použít mnohovláknovou verzi programu. Přesto nelze říct, že by omezení disku nehrálo žádnou roli, zpomalení se projevilo. Svou roli může hrát i fakt, že server, na kterém, a ze kterého celý pokus běžel, je virtuální a jeho disky jsou jen virtuálně přidělené místo na SAN, ke kterému by ale neměl být schopen přistupovat nijak zvýhodněně.

Celkové zhodnocení testů

Kombinací mnohovláknových a jednovláknových testů v různém prostředí se podařilo zjistit, že mnohovláknové použití se vyplatí téměř vždy, i na serverech s jedním virtuálním diskem běží rychleji mnohovláknové sbírání dat než jednovláknové. Celá část sbírání dat se ukázala být překvapivě náročná na procesor, a proto by běžících vláken nemělo být příliš. Logika spouštění postupně podle prohledávaných oblastí je sice realizovatelná, protože vlákna mají nejmenší priority, mohla by ale selhat, proto by spouštění mělo být doplněno o monitorování aktuálního stavu procesoru a postupného spouštění jednotlivých vláken. Jinak se sběr dat ukázal být překvapivě kolísavý mezi živými a testovacími daty. Nejpomalejší průběh byl na živých datech a dosahoval zhruba čtyři prvky souborového systému za vteřinu. Naopak nejrychlejší průběh

získaný mnohovláknovm přístupem na testovacích datech dosahuje 97,4 souborů za vteřinu. Při odhadovaném množství dat na sto milionů souborů se nabízí úvaha, zda jsou tyto rychlosti dostatečné pro reálné nasazení. Bohužel další zrychlení se nezdá být reálné, protože hlavní důvod pomalosti skriptu není v něm samotným, ale v prostředí které prohledává. Dalšího zrychlení je možné dosáhnout úpravou vstupních dat, například prohledávání adresářové struktury do menší hloubky. Zaručovaná hloubka ke správě jsou pouze tři stupně, skript ale počítá s potřebou hlubšího prohledávání.

4.5. Návrh Webového portálu

i) Základní specifikace požadavků

Úkolem webového portálu bude používat data získaná při prohledávání a uložená do databáze. Portál z nich bude vyhodnocovat ideální přiřazení uživatelských skupin pro jednotlivé uživatele. Od přesného vyhodnocení této informace očekávám zefektivnění a zrychlení celkového procesu přidělování práv, stejně jako zdokonalení zabezpečení celého procesu. Portál je teoreticky přístupný kterémukoliv uživateli v doméně, protože kdokoliv může žádat o přístup, portál tedy musí být schopen uživatele identifikovat. Samozřejmostí je připojení se na databázi nalezených údajů. Vzhledem k tomu, že častým problémem je dodání přesné adresy požadovaných sdílen adresářů od uživatelů, popřípadě od heldpesku, je nutné výběr adresy uživateli co nejvíce ulehčit. Výběr požadované adresy bude tedy doplněn funkcí našeptávače, který bude uživatelům radit ty správné adresy. Uživatel musí mít možnost vybrat více než jednu adresu. Ke každé vybrané adrese musí mít možnost přidat požadovaný druh přístupu, pro čtení, či pro zápis[12], ideálně, zadat uživatele, který poslouží jako šablona. Portál musí být schopen zadané parametry vyhodnotit, a na základě dat z databáze zjistit, které ze skupin je potřeba uživateli přidělit aby příslušná práva získal. Výstupem portálu bude seznam skupin, které je potřeba uživateli, případně uživatelům, pokud jich žadatel zadá více, přidělit. Možným budoucím rozvojem portálu je napojení se na evidenční

[12] Po špatných zkušenostech,k kdy uživatelé sebrali administrátorům přistupová práva nepřidělujeme uživatelům práva, která umožnují přstup modifikovat.

systém žádostí, až bude dokončen, a automatické vyváření nových žádostí na základě údajů z portálu a databáze v něm.

ii) Interakce a použití

Nejčastějším uživatelem webového portálu a celého tohoto systému by měli být sami uživatelé, žádající o přístup. S pomocí tohoto portálu mohou vytvořit přesnou žádost, kterou poté přepošlou helpdesku, pravděpodobně emailem. Ten pak vytvoří oficiální žádost v evidenčním systému. Pokud uživatelé nebudou schopni, anebo nebudou chtít, zabývat se celou věcí takto, poslouží tento portál operátorům helpdesku, kterým pomůže získat všechny nutné údaje a tím zabraní stárnutí žádosti pro nedostatek údajů. Administrátorovou rolí je jednak přijetí žádosti a rozhodnutí o přidělení práv. Jeho další zodpovědností je správa portálu jako takového, dohlížení na sběr dat a určování nových oblastí k prohledání. Souvislosti shrnuje kontextový diagram:

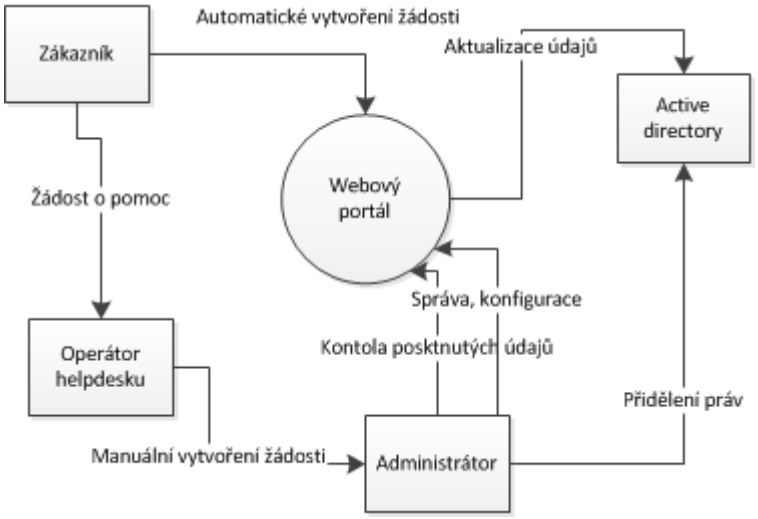


Diagram 4-4 Kontext webového portálu

iii) Proces vytvoření žádosti v portálu

Je několik způsobů jak lze zadat žádost o přidělení přístupu, všechny probíhají na základě uživatele kontaktujícího helpdesk. Je i několik způsobů jak může tato žádost probíhat, a několik hlavních členů této interakce. Vzájemné interakce mezi členy popisuje následující DFD diagram.

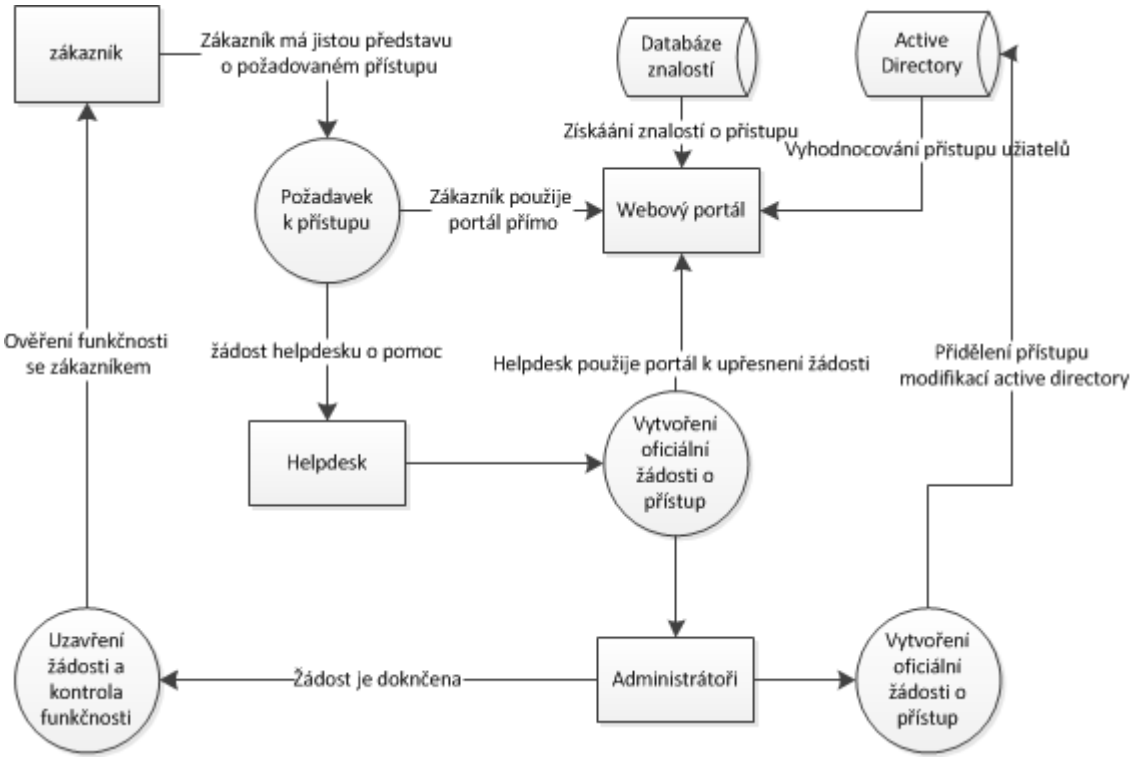


Diagram 4-5 DFD žádosti o přístup

Diagram popisuje všechny běžné kanály, kudy žádost přichází. Existují ještě kanály vedlejší, uživatelé se občas obrací na administrátory přímo, ale tento kanál není oficiální a tudíž není v diagramu zahrnut. Ve všech případech slouží webový portál ke kompletování žádosti. Svou funkcí umožnuje obejít složité zjišťování správného způsobu přidělení práv a zároveň zaručit, že příslušná žádost bude mít všechny podstatné náležitosti a nebude potřeba ji vracet k doplnění. Následující diagram popisuje, jak by měla vypadat žádost o přístup s použitím portálu.

Diagram 4-6 Logika webového portálu

Samotný proces vytvoření portálu je jednoduchý a bude zahrnovat pouze několik webových stránek. Zákazník se identifikuje svými přihlašovacími údaji do domény, postupně zadává adresy, a druhy přístupu, a také komu má tento přístup patřit. Ideálním vstupem je i šablonový uživatel, který sám už přístup má, tato informace zpřesní hledání průniku skupin. Po zadání všech přístupů jsou nalezeny skupiny, které mají na příslušné adresy přístup správného druhu. Pokud existuje jedna skupina co má přístup na víc lokací, je upřednostněna ta. Pokud je zadán šablonový uživatel, úplnou přednost mají skupiny, jejichž členem je šablonový uživatel. Z těchto skupin je vytvořen výpis, který shrnuje, kam mají být přidáni kteří uživatelé. Možným rozšířením je úplná, anebo částečná automatizace procesu. Úplná automatizace by zahrnovala automatickou úpravu active directory. Od této části bylo upuštěno z důvodů bezpečnostních. Administrátoři by ztratili přehled, kdo dostal kam přístup. Částečná automatizace by představovala automatické vytváření žádostí v systému žádostí. Tento vývoj byl původní součástí projektu, avšak vzhledem k tomu, že v současné době prodělává firma komplikovaný přechod ze systému *HP openview servicedesk* na *Global service now* a projekt ještě zdaleka není dokončen, je vývoj jakékoliv interakce s tímto systémem nereálný.

iv) Jednotlivé Webové stránky

Samotný webový portál se bude skládat pouze ze tří webových stránek, které budou postupně plnit jednotlivé funkce portálu.

Diagram 4-7 Přechody mezi stránkami

- Přihlašovací stránka

Je jen vstupní stránkou, bude sloužit k tomu, aby získala přihlašovací údaje uživatele, pod těmito přihlašovacími údaji poběží i vnitřní skripty portálu.

- Zadávací stránka

Je hlavní funkční stránkou portálu. Zde může žadatel postupně zadávat komu a jaká práva chce přidělit. Bude zde postupně se zvětšující formulář, který umožní načtení adres požadovaných přístupů a uživatelů. Teoreticky nemusí existovat limit pro množství zadaných údajů, ale z důvodů přehlednosti množství údajů omezeno bude. Vzhledem k tomu, že nepředpokládám přesnou znalost zadávaných cest u uživatelů, bude tato část vybavena našeptávačem. Na této stránce bude rovněž existovat velká část skrytých skriptů pracujících s active directory a sloužících k ověřování identit, rozpoznávání účtů a vyhodnocování členství ve skupinách.

- Stránka s výsledky

Slouží k zobrazení výsledků, které uživatel dále použije. Spojení mezi zadávací stránkou a stránkou s výsledky musí být oboustranné a musí zachovávat zadané údaje pro případ, že by uživatel chtěl zadání změnit. V případě rozšíření funkcí portálu zde budou možnosti na automatické vytvoření žádosti.

- Celkové zobrazení a styl.

  Firemními barvami jsou červená a žlutá a pro firemní stránky je běžný určitý standardizovaný design, který je určen centrálními sdílenými kaskádovými styly. V rámci zachování firemní kultury tedy styly využiji, i když nejsem jejich autorem.

v) Skryté funkcionality portálu

Pro přesné zjištění skupin kam je potřeba člověka přidat je potřeba ze skupin které mají na dané místo přístup vybrat tu správnou. To je jednoduché, pokud je součástí zadání i šablonový uživatel, pokud není, je třeba minimalizovat množství práv, poskytnutých navíc. Samotné získání jmen skupin, kterých je uživatel členem je věcí opakovaného tázaní se na členy skupin do active directory a v podstatě prohledáváním do hloubky. Počet členů skupiny sice není teoreticky omezen, ale omezena je velikost Kerberos tiketu, který v sobě členství ve skupinách, a to i vnořených, nese. Standartní doménová velikost je 12000 KB, což je i náš případ. Množství skupin, kterých může být uživatel členem, je tedy zhruba 300, v závislosti na jejich typu. Členství ve větším množství skupin nemá smysl. Tuto možnost, stejně jako možnost kruhového členství ve skupinách je ale stejně potřeba ošetřit. Z celkového množství skupin jsou na základě seznamu vyřazeny skupiny, které patří administrátorům, a do kterých se běžní uživatelé nepřidávají. Výběr těch správných skupin bude záležet na jejich vlastnostech a také na tom do kolika a jakých adresářů poskytují přístup. To předpokládá opakované dotazování jak do databáze znalostí, tak do active directory. Celý proces filtrace a výběru skupin probíhá zhruba takto:

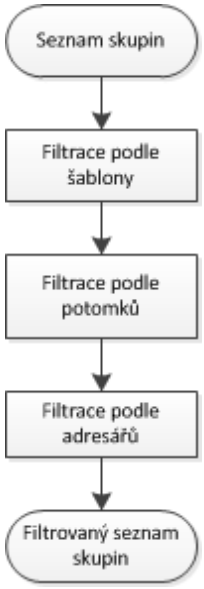


Diagram 4-8 Proces filtrování skupin

- Filtrace podle šablony

Je jednoduchým množinovým průnikem s množinou skupin šablonového uživatele. Pokud je k dispozici šablonový uživatel, pak přidáme jen nějaké ze skupin, jichž je členem, pokud ho nemáme, bude tato část přeskočena.

- Filtrace podle potomků

Jakákoliv skupina, která je členem, nebo také potomkem skupiny jiné, dědí i její práva. Je však nežádoucí, aby byl uživatel přidáván do celé řady skupin v dědičnosti. Pokud dvě po sobě jdoucí skupiny neobsahují žádná přístupová práva navíc, je skupina rodiče vyřazena ze seznamu. Vzhledem ke standardu přidělování práv, používaném v DHL, se upřednostňují globální skupiny před doménově lokálními.

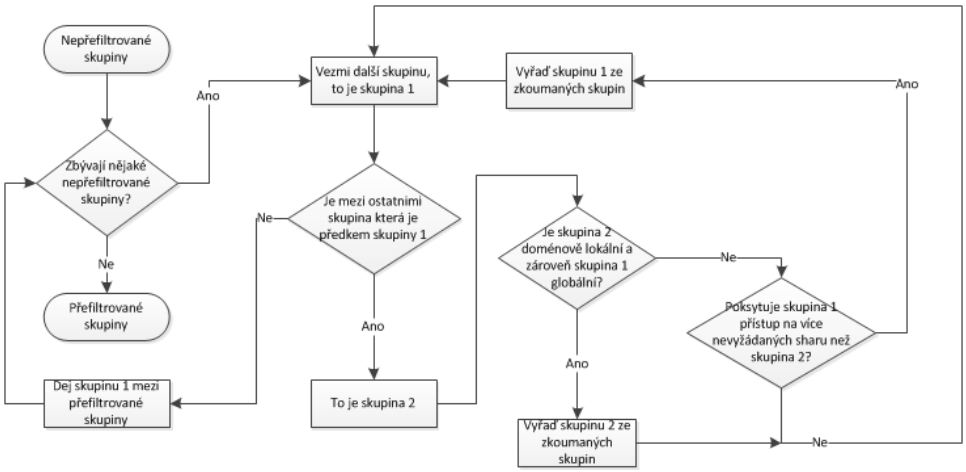


Diagram 4-9 Diagram filtrování podle skupin

- Filtrace podle adresářů

Dost často se stává, že nějaká skupina přiděluje přístup na více žádaných adresářů najednou, je žádoucí použít menší množství takovýchto skupin, místo přiřazovaní skupin za každý sdílený adresář

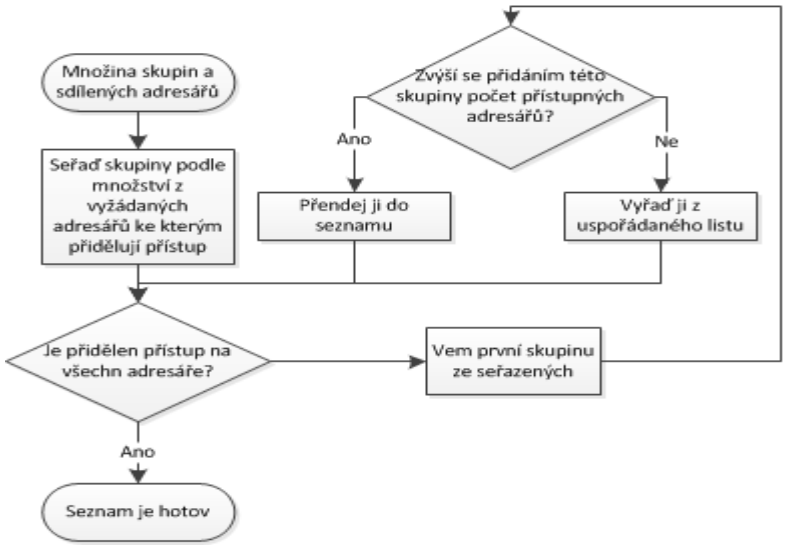


Diagram 4-10 Filtrace podle adresářů

vi) Možná budoucí rozšíření portálu

Předpokládaným rozšířením portálu je rozšíření na automatické vytváření žádosti v evidenčním systému. Toto rozšíření bude již brzy možné realizovat, Servicenow je nyní ve stabilním provozu a prodělává již jen málo úprav, přesto se jimi nebudu

konkrétně zabývat v této práci a ponechám jej pro další rozvoj. Momentálně se zdá že Servicenow bude umožnovat vytváření nových žádosti na základě webservices. Další částí, kterou je možné vytvořit, je možnost automaticky odesílat emaily s žádostí o přístup vlastníkům adresářů. Ve své momentální podobě na to tento projekt pamatuje. Existují vyhledávací funkce pro nalezení vlastníků a v databázi tabulka pro jejich uložení. Bohužel se tato část střetává s kvalitou nalezených dat. Je velmi obtížné přimět uživatele k tomu, aby poskytli při vytváření sdíleného adresáře jméno zodpovědné osoby, a je ještě obtížnější toto jméno udržovat aktuální. Tato část byla nicméně zavedena do aktuálního projektu, a možnost odeslat žádost bude přidána, jakmile se kvalita dat zlepší. K tomu by měl částečně přispět i tento projekt, který může zároveň posloužit jako nástroj pro vynucování určité kvality dat. Poslední možností do budoucna je přidat portálu možnost vytvářet žádosti o nové sdílené složky, jejich automatické vytváření, a zanášení do databáze.

vii) Implementace

Vývoj poslední součásti, webového portálu trval zhruba 14 dní. Použitá verze MVC modelu je nakonec MVC 3, a ke zobrazování webových stránek je částečně použit i *Razor engine*, specifický interpretovaný jazyk pro zobrazování stránek, určený pro použití s C# a ASP. Net. Celý webový portál se také spoléhá na autentizaci vůči doméně a je proto nepoužitelný z vnějšího prostředí. Jako priklad MVC modelu uvádím příklad zmíněné autentizace.

Příklad, autentizace:

Model:

```
namespace TestMVCEmptyTemplate.Controllers
{
    public class LoginModel
    {
        [Required]
        public string Username { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

Controller:

```
        public ActionResult index(LoginModel model)
```

```
        {
            //  kod co neco dela
            if (ModelState.IsValid)
            {
                if (Membership.ValidateUser(model.Username,model.Password))
                { //here you have cookie
                    FormsAuthentication.SetAuthCookie(model.Username, false);
                    return RedirectToAction("index", "Share");
                }
                else
                {
                    ModelState.AddModelError("", "Invalid username or password");
                }
            }
            //konec kodu co neco dela
            return View();
        }
```

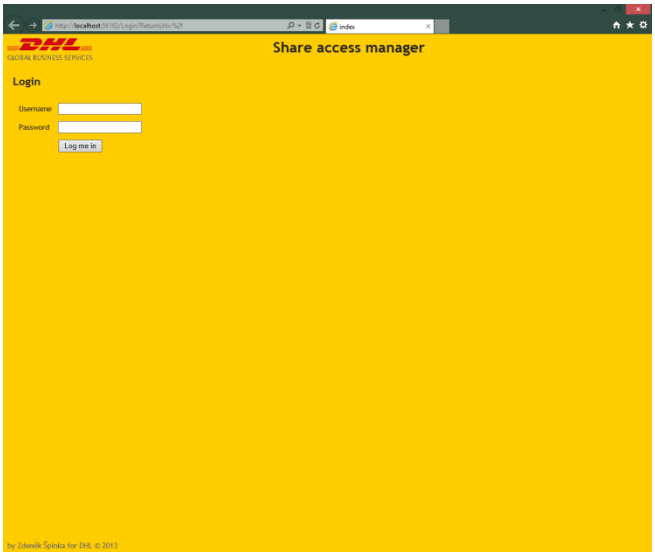
Nastavení providera:

```
<connectionStrings>
    <add name="ADService" connectionString="LDAP://prg-dc.dhl.com/DC=prg-dc,DC=dhl,DC=com"
/>
  </connectionStrings>
<providers>
        <add name="AspNetActiveDirectoryMembershipProvider"
type="System.Web.Security.ActiveDirectoryMembershipProvider"
connectionStringName="ADService" attributeMapUsername="sAMAccountName" />
</providers>
```

View, samotná stránka:

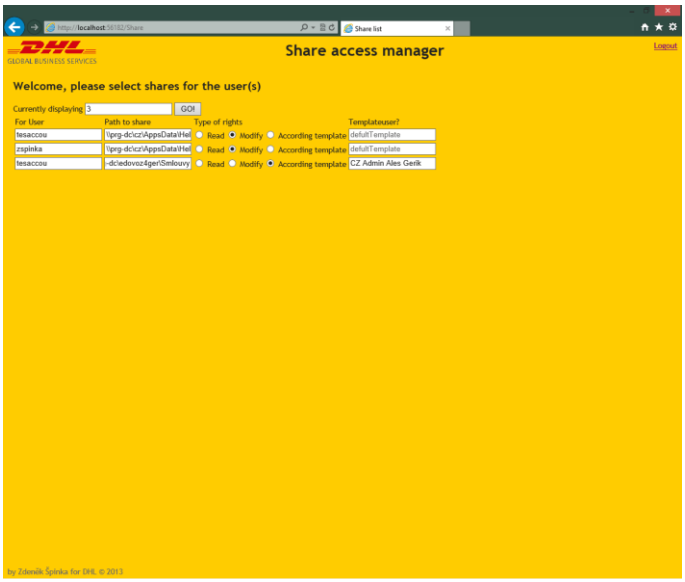
```
@using (Html.BeginForm())
{
    <table cellspacing="10">
    <tr> <td>@Html.LabelFor(m => m.Username)</td> <td>@Html.TextBoxFor(m => m.Username, new
{ style = "width:150px" }) </td></tr>
    <tr><td>@Html.LabelFor(m => m.Password)</td><td>@Html.PasswordFor(m => m.Password, new
{ style = "width:150px" })</td></tr>
    <tr><td colspan="2" align="center"><input type="submit" value='Log me in' /> </td></tr>
    </table>

    @Html.ValidationSummary()

}
```

A celkový pohled na věc



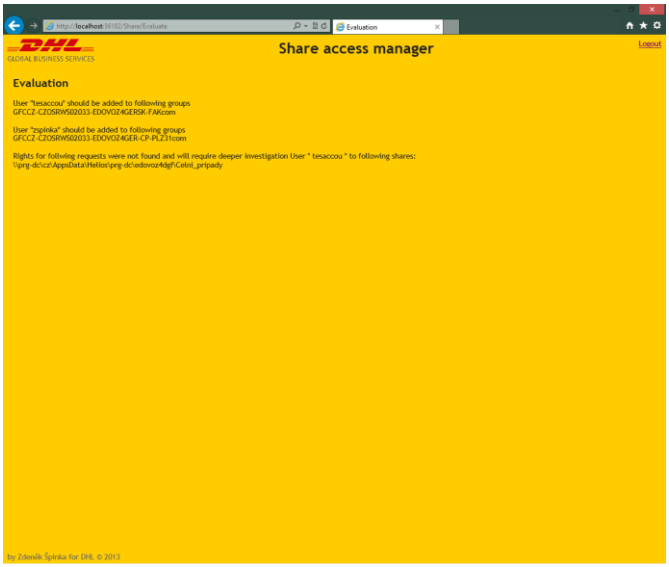
Obrázek 4-3 Přihlašovací stránka

Přihlašovací stránka zobrazená o kus výše ověří přihlašovací údaje uživatele vůči doméně, to mimo jiné umožnuje, při znalosti správných údajů, i přihlášení se pod jménem někoho jiného, a vytvoření žádosti jeho jménem.



Obrázek 4-4 Přidělování práv

Stránka pro zdávání údajů umožnuje zadat žádost až o dvacet různých adresářů ke každému zažádat o přidělovaná práva buďto pro čtení, nebo pro zápis, anebo stejný druh práv, která už nějaký uživatel má.



Obrázek 4-5 Vyhodnocení

Poslední stránka zobrazuje vyhodnocení do jakých supin a jakým způsobem mají být práva přidělena, v rámci vyhodnocení je prováděna i určitá selekce a skupiny

pro administrátory, či systémové prostředky, nejsou zobrazovány. Pokud by došlo k dalšímu rozvoji projektu, bude tato stránka modifikována, například na automatické vytváření žádostí.

viii) Aktuální stav

V této chvíli je vývoj ukončen a projekt má postupně přejít do testovací fáze. To však závisí na tom, zda-li se podaří najít vhodné, funkční produkční prostředí na kterém by mohl celý projekt, a to včetně databáze běžet. Na výsledku testování bude zároveň záviset další vývoj, zejména zda bude webový portál napojen na ServiceNow. Možnost, že by práva přiděloval automaticky, byla zavržena z bezpečnostních důvodů hned na počátku. Poté co bude systém uveden do provozu, je třeba ještě rozhodnout, na která data bude napojen. Data jsou na NAS rozdělena podle zemí a divizí společnosti, a zavedení tohoto systému do provozu si, kromě rozpočtu na vybudování serveru, může vyžádat souhlas vlastníků dat.

5. Závěr

Účelem celého projektu bylo vyvinout systém na správu dat. Důvodem pro jeho vývoj byla vysoká neefektivita při přidělování uživatelských práv jednotlivým uživatelům. žádosti byly často nejasné, vraceny k doplnění a to vedlo skrze prodlevy k nespokojenosti uživatelů a plýtvání náklady v podobě placeného času administrátorů. Skládá se ze tří součástí. První součástí jsou skripty určené ke sběru dat, běžící na serveru jako *scheduled task*. Ty sbírají data o přístupových právech ve vybraných lokacích a zároveň hlásí odchylky od standardů, jako jsou například plná práva pro uživatele v situacích kdy, by je měli mít jen administrátoři. Všechny takto získaná data jsou uložena do databáze, MS SQL 2008 R2 Express. Express verzi používám proto, že nepotřebuje licenci, a databáze je natolik malá a jednoduchá, že se tam vejde. Rozhraním pro užívání tato získaných údajů je webový portál, který umožnuje uživatelům, helpdesku případně administrátorům analyzovat. Systém má vnitřní algoritmus, podle kterého rozhodne, do kterých skupin má uživatel být přidán a tyto skupiny doporučí. Aby byl systém žádostí efektivnější než předchozí varianta manuálního vytváření žádostí, je nedílnou součástí webového portálu i našeptávač, který nedovolí žadateli vybrat neexistující sdílený adresář anebo uživatele, a tím zabrání vytváření neurčitých a nekompletních žádostí. Systém zcela neeliminuje nutnost manuálního přidělování práv. V případě komplikovaně přidělených práv systém nemusí být schopen rozhodnout. V případě žádosti o práva v neprohledaných oblastech je asistence administrátora stále nutná. Napojení na systém žádostí k jejich úplnému automatickému vytváření nebylo naprogramováno, nejednalo se o původní součást projektu, hlavně z důvodu výměny systému žádosti za jiný, je s ním ale počítáno jako s možným rozšířením a systém je k tomu uzpůsoben. V této chvíli byl vývoj systému ukončen a čeká se na jeho nasazení na serveru a spuštění testování, to vše bude záviset na manažerském rozhodnutí a také na penězích, vybudování produkčního prostředí není zcela levná záležitost, a je zcela mimo kompetence mě jako administrátora.

6. Seznam použitých zdrojů

1. **PŘICHYSTAL, Ing. Oldřich.** Adresářová služba Novell eDirectory. *Adresářová služba Novell eDirectory.* [Online] Novell, 2011. [Citace: 25. 6 2012.] http://www.novell.cz/cs/aktuality/technicke-clanky/adresarova-sluzba-novell-edirectory.html.

2. **Microsoft.** *Managing a Microsoft Windows server 2003 enviromemt.* Redmont, Washington : Microsoft, 2005.

3. —. Security Identifier Structure. *Security Identifier Structure.* [Online] Microsoft, 2012. [Citace: 2. 7 2012.] http://technet.microsoft.com/en-us/library/cc962011.

4. **RUSSINOVICH, Mark E. , SOLOMON, David A.** *Windows Internals, Fifth edition.* Redmont, Washington : Microsoft Press, 2009. ISBN: 978-0-7356-2530-3.

5. **NAIK, Dilip C.** *Inside Windows Storage.* Boston : Pearson Education, Inc., 2003. ISBN: 0-321-12698-X.

6. **Microsoft.** DFS Management. *DFS Management.* [Online] Microsoft, 22. 8 2005. [Citace: 25. 05 2012.] [2012-5-30]. http://technet.microsoft.com/en-us/library/cc756883%28v=ws.10%29.

7. **MACDONALD, Matthew.** *Beginning ASP.NET in C# 2010.* New York : Springer Science + Bussiness Media, LLC., 2010. ISBN-13: 978-1-4302-2609-3.

8. **MALINA, Patrik.** *Powershell 2.0.* Brno : Computer Press, a.s., 2010. ISBN: 978-80-251-2732-2.

9. **FREEMAN, Adam.** *Pro ASP.NET MVC4.* místo neznámé : apress, 2012. ISBN13: 978-1-4302-4236-9.

10. **MACDONALD, Matthew.** *Begining ASP.NET 4 in C# 2010.* místo neznámé : apress, 2010. ISBN13: 978-1-4302-2608-6.

11. **MACDONALD Matthew, Freeman Adam.** *Pro ASP.NET 4 in C# 2010.* místo neznámé : apress, 2010. ISBN13: 978-1-4302-2529-4.

12. **KHORSRAVI, Shahram.** *Professional IIS 7 and ASP.NET Integrated Programming.* Indianopolis : Wile Publishing Inc., 2008. ISBN: 978-0-470-15253-9.

13. **CRANE, Dave, PASCARELLO, Eric a FARREN, James.** *Ajax in Acion.* Greenwitch : Manning Publications Co., 2006. ISBN: 1-93394-61-3.

14. **MUELLER, John Paul.** *Mastering IIS 7 Implementation and Administration.* Indianopolia : Wiely Publishing Inc., 2007. ISBN: 978-0-470-17893-5.

15. **DARIE, Cristian, a další, a další.** *Building a Responsie Web Application.* Birmigham : Packt Publishing Ltd., 2006. ISBN: 1-904811-82-5.

16. **WILSSON, Ed a spol.** *Windows powershell 2.0 Best Practicies.* REdmont : Microsoft Press, 2010. Library of Congres controll number: 2009938599.

17. **SCHAFER, Ken a spol.** *Professional IIS 7.* Indianopolis : Wiley Publishing Inc., 2008. ISBN: 978-0-470-09782-3.

18. **Microsoft.** *Implementing and Administering Microsoft Internet Information Services (IIS) 6.0.* Redmont : Microsoft Press, 2005.

19. **MALINA, Patrik.** *Powershell podrobný průvodce skriptováním.* Brno : Computer Press a.s., 2007. ISBN: 978-80-251-1816-0.

20. **NIELSEN, Paul, WHITE, Mike a PARUI, Utam.** *Microsoft SQL Server 2008 Bible.* Redmont : Wiely Publishing Inc., 2009. ISBN: 978-0-470-25704-3.

7. Přílohy

Haverstor2.10.ps1, Skript na sběr dat

```
param ([string]$Where=$(Throw "I really need to have a path"),
[int]$MaxDepth=(Throw "Please provide required depth"),
[string]$jobname=$(Throw "What is name of the job?"),
[string]$logpath)

[string]$global:WhereParam=$Where #globalizing of input parameters
[int]$global:MaxDepthParam=$MaxDepth
[string]$global:jobname=$jobname

[System.Threading.Thread]::CurrentThread.Priority = 'Lowest'

$global:WhereParam = $global:WhereParam -replace "\\$" #cutting off end of path
$logpath = $logpath -replace "\\$"

function GetRights
{
        param($ancestor)
        try{
        $prava=Get-Acl $ancestor # -ErrorAction Stop
        }
        Catch [System.UnauthorizedAccessException]
        {
                $prava = "Acccess denied"
        }
        return $prava
}

function GetDomain
{ param([string]$instring)
        if($instring[2] -eq "=")      #distinguished
        {
                $instring=[Regex]::Matches($instring,  "DC=.*",  'IgnoreCase')  |  foreach-
object {$_.Value}
                #Write-Host $missing
                $tobefound=$instring -replace "DC="
                #Write-Host $tobefound
                $tobefound=$tobefound -replace ",","."
        }
        else #netbios
        {
                [string[]]$jmeno=$instring.split("\")
                $tobefound=$jmeno[0]
        }
        $type = [System.DirectoryServices.ActiveDirectory.DirectoryContextType]"Domain"
```

```
        $context                                =                                New-Object
System.DirectoryServices.ActiveDirectory.DirectoryContext($type,$tobefound)
        $domain = [System.DirectoryServices.ActiveDirectory.Domain]::GetDomain($context)
        $direntry=$domain.GetDirectoryEntry()
        $Root = $direntry #$Domain.GetDirectoryEntry()
        $IP=$domain.FindDomainController().IPAddress
        $DomainName=$tobefound
        return $direntry.distinguishedname
}

Function Compare-ACL{
    Param (
    $ReferenceObject,
    $DifferenceObject
    )
    #$ReferenceObjectACL = ((Get-ACL $ReferenceObject).AccessToString).Split("`n")
    #$DifferenceObjectACL = ((Get-ACL $DifferenceObject).AccessToString).Split("`n")
    #Compare-Object       -ReferenceObject       $ReferenceObjectACL       -DifferenceObject
$DifferenceObjectACL
        return    Compare-Object    -ReferenceObject    $ReferenceObject.Accesstostring    -
DifferenceObject $DifferenceObject.Accesstostring
}

function GetMembers
{
        param([string]$vstup)
        #We are expecting distinguished name, tho pint is get members groups or groups and
users
        $dom=[Regex]::Matches($vstup, "DC=.*", 'IgnoreCase') | foreach-object {$_.Value}
        [string]$filtr="(&(objectclass=group)(memberof="+$vstup+"))"
        [string]$entrypath="LDAP://" + $Global:DomainIndex[$Dom]
        $entry  = New-Object System.DirectoryServices.DirectoryEntry($entrypath)
        $objSearcher   =   New-Object   System.DirectoryServices.DirectorySearcher   ($entry,
"$filtr")
        $objSearcher.SearchScope = "Subtree"
        [string[]]$a=$objSearcher.FindAll()                |                foreach-object
{($_.Properties).Item("distinguishedName")}
        #$a
        return $a
}



function ShareSQL
{
        param($cesta)
        $cesta=$cesta -replace "'","''"
        $global:WhereParamCondition="      WHERE      Path      =`'$cesta`'"                #
$global:WhereParamCondition=" WHERE Path =`'$($Descendant.FullName)`'"
```

```
        $global:cmd.CommandText      ="select      *      from      AccessManager.dbo.Share
$global:WhereParamCondition"
        $global:SqlAdapter.SelectCommand = $global:cmd
        $DataSet = New-Object System.Data.DataSet
        $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in  the
database
        #loading whole share
        if ($returnedrows -eq 0)
        {

                #setting level of share
                $upravenacesta=$cesta  -replace  "^\\\\(\w*[-.]*)+\\"  #cutting  off  first
location so what remains is level 1
                $level=[regex]::Matches($upravenacesta,"\\").Count+1


                $global:cmd.CommandText      ="INSERT      INTO      AccessManager.dbo.SHARE
(Path,Pathlevel)  VALUES  (`'$cesta`',`'$level`')"  #$global:cmd.CommandText  ="INSERT  INTO
AccessManager.dbo.SHARE (Path) VALUES (`'$($Descendant.FullName)`')"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)

                $global:WhereParamCondition="         WHERE         Path         =`'$cesta`'"
#$global:WhereParamCondition=" WHERE Path =`'$($Descendant.FullName)`'"
                $global:cmd.CommandText     ="select     *     from     AccessManager.dbo.Share
$global:WhereParamCondition"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)
                $ShareID=$DataSet.Tables[0].Rows[0]["ID"]  #Here  we  have  stored  ID,  the
primary queue of just created record
        }
        else
        {
                $ShareID=$DataSet.Tables[0].Rows[0]["ID"]  #Here  we  have  stored  ID,  the
primary queue of found record
        }
        return $ShareID;
}

function OwnerSQL
{
        param($vstup)
         if (($vstup -ne $null)-and($vstup -ne ""))
                {
                [string]$vec=$vstup;

                #is it mail?
```

```
                if($OwnerNote -match  "@")
                {
                        $IsItMail=$true
                        $global:WhereParamCondition="mail"
                        $SecondValue="name"
                }
                else
                {
                        $IsItMail=$false
                        $global:WhereParamCondition="name"
                        $SecondValue="mail"
                }
                $global:cmd.CommandText  ="select  *  from  AccessManager.dbo.OWNER    WHERE
$global:WhereParamCondition =`'$vec`'"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in
the database
                if ($returnedrows -eq 0) #creating record in the database
                {
                        $global:cmd.CommandText    ="INSERT    INTO    AccessManager.dbo.OWNER
($global:WhereParamCondition,$SecondValue) VALUES (`'$vec`','unknown')"
                        $global:SqlAdapter.SelectCommand = $global:cmd
                        $DataSet = New-Object System.Data.DataSet
                        $returnedrows=$global:SqlAdapter.Fill($DataSet)

                        #again searching for it to get ID
                        $global:cmd.CommandText  ="select  *  from  AccessManager.dbo.OWNER
WHERE $global:WhereParamCondition =`'$vec`'"
                        $global:SqlAdapter.SelectCommand = $global:cmd
                        $DataSet = New-Object System.Data.DataSet
                        $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is
not in the database
                        $OwnerID+=$DataSet.Tables[0].Rows[0]["ID"]
                }
                else
                {
                        $OwnerID+=$DataSet.Tables[0].Rows[0]["ID"]
                }
                return $OwnerID
        }
        else
        {
                return $null
        }
}

function GroupSQL
{
```

```
        param($GroupPath,$GroupName)
        $global:WhereParamCondition="DistinguishName =`'$($GroupPath)`'"
        $global:cmd.CommandText    ="select    *    from    AccessManager.dbo.[Group]       WHERE
$global:WhereParamCondition"
        $global:SqlAdapter.SelectCommand = $global:cmd
        $DataSet = New-Object System.Data.DataSet
        $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in  the
database
        if ($returnedrows -eq 0) #creating record in the database
        {
                $global:cmd.CommandText      ="INSERT      INTO      AccessManager.dbo.[Group]
(Distinguishname,Name) VALUES (`'$GroupPath`',`'$GroupName`')"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)


                #again searching for it to get ID
                $global:cmd.CommandText  ="select  *  from  AccessManager.dbo.[Group]    WHERE
$global:WhereParamCondition"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in
the database
                $GroupID=$DataSet.Tables[0].Rows[0]["ID"]
        }
        else
        {
                $GroupID=$DataSet.Tables[0].Rows[0]["ID"]
                #Update existujici grupy, vpodstate neni co updatovat
        }
        return $GroupID
}

function GroupShareSQL
{
        param($GroupID,$ShareID,$TypeofAccess)

        if($GroupID -eq 2456){
                Write-Host "stop"
        }
        $global:WhereParamCondition="GROUPID =`'$($GroupID)`' and SHAREID =`'$($ShareID)`'"
        $global:cmd.CommandText  ="select  *  from  AccessManager.dbo.GroupToShare    WHERE
$global:WhereParamCondition"
        $global:SqlAdapter.SelectCommand = $global:cmd
        $DataSet = New-Object System.Data.DataSet
        $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in  the
database
        if($returnedrows -gt 0) #it IS in the database
```

```
        {
                #$AssociationGroupShareID=$DataSet.Tables[0].Rows[0]["ID"]
                $AssociatonType=$DataSet.Tables[0].Rows[0]["RightsType"]
                if($TypeofAccess -ne $AssociatonType) #type of rights is changed
                {
                        $global:WhereParamCondition="  WHERE  GROUPID  =`'$($GroupID)`'  and
SHAREID =`'$($ShareID)`'"
                        $global:cmd.CommandText  ="UPDATE  AccessManager.dbo.GroupToShare  SET
RightsType=`'$TypeofAccess`' $global:WhereParamCondition"
                        $global:SqlAdapter.SelectCommand = $global:cmd
                        $DataSet = New-Object System.Data.DataSet
                        $returnedrows=$global:SqlAdapter.Fill($DataSet) #if
                }
        }
        else #creating of new record
        {
                #$global:WhereParamCondition="  WHERE  GROUPID  =`'$($GroupID)`'  and  SHAREID
=`'$($ShareID)`'"
                $global:cmd.CommandText   ="INSERT   INTO      AccessManager.dbo.GroupToShare
(GroupID,ShareID,RightsType) VALUES (`'$($GroupID)`',`'$($ShareID)`',`'$TypeofAccess`')"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)  #if  0  then  share  is  not  in
the database
                #$AssociationGroupShareID=$DataSet.Tables[0].Rows[0]["ID"]
        }
}

function GroupOwnerSQL
{
        param($GroupID, $OwnerID)
        $global:WhereParamCondition="    WHERE    GROUPID    =`'$($GroupID)`'    and    OwnerID
=`'$($OwnerID)`'"
        $global:cmd.CommandText     ="select     *     from     AccessManager.dbo.GroupToOwner
$global:WhereParamCondition"
        $global:SqlAdapter.SelectCommand = $global:cmd
        $DataSet = New-Object System.Data.DataSet
        $returnedrows=$global:SqlAdapter.Fill($DataSet)

        if($returnedrows -eq 0) #if the association doesn't exist
        {
                #$global:WhereParamCondition="  WHERE  GROUPID  =`'$($GroupID)`'  and  OwnerID
=`'$($OwnerID)`'"
                $global:cmd.CommandText   ="INSERT   INTO      AccessManager.dbo.GroupToOwner
(GroupID,OwnerID) VALUES (`'$($GroupID)`',`'$OwnerID`')"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)
```

```
                #again getting ID of created record
                $global:WhereParamCondition="  WHERE  GROUPID  =`'$($GroupID)`'  and  OwnerID
=`'$($OwnerID)`'"
                $global:cmd.CommandText  ="select  *  from  AccessManager.dbo.GroupToOwner
$global:WhereParamCondition"
                $global:SqlAdapter.SelectCommand = $global:cmd
                $DataSet = New-Object System.Data.DataSet
                $returnedrows=$global:SqlAdapter.Fill($DataSet)
                $AssociationGroupOwnerID=$DataSet.Tables[0].Rows[0]["ID"]

        }
}

function SendToLog
{
        param([string]$Dologu)
        if(-not($Global:TheTextLog.contains($Dologu)))
        {
                $Global:TheTextLog+= ($Dologu + "`n")
        }
        Out-File  -FilePath  $global:logname  -Append    -Encoding  "Unicode"  -InputObject
$Dologu
}

function ACLupdater #updates rights on folder so they are more readable
{
        param($zaznam, [ref]$RefParsingHashTable)
        $tail=$zaznam
        $head=$zaznam -replace " Allow.*| Deny.*"
        $tail=$tail -replace $($head -replace '\\','\\')
        $ParssedDescendantStringrights=New-Object  System.Collections.ArrayList  #  an  empty
collection
        if($head -match "^S-\d-\d+-\d+-\d+-\d+-\d+.*") #unenumerated SID
        {
                $objUser= $null
                $Translated=$true
                $objSID = New-Object System.Security.Principal.SecurityIdentifier ("$head")
                try
                {
                        $objUser = $objSID.Translate( [System.Security.Principal.NTAccount])
                }
                catch
                {
                        if ($_.Exception -match ".*Some or all identity references could not
be translated.*")
                 {
                                $Translated=$false
                        }
                }
```

```
                if($Translated)
                {
                        $head=$objUser.Value
                }
        }
        #checking of what kind of records is there searching for susspitions ones: Creator
Owner or authenticated user
        if($head -like "*Authenticated Users*" )
        {
                SendToLog -Dologu $("Authenticated on " + $Descendant.FullName)
        }
        if($head -like "*CREATOR OWNER*")
        {
                SendToLog -Dologu $("Creator owner on " + $Descendant.FullName)
        }
        if(-not($RefParsingHashTable.Value.ContainsKey($head))) #the record is new
        {
                $ParssedDescendantStringrights.add($zaznam)                                    |
%{$RefParsingHashTable.Value.add($head,$_)}
                #$parsingcounter++;
                #$ParssedDescendantStringrights.add($zaznam)
        }
        else #the record is old
        {
                $position=$RefParsingHashTable.Value[$head];
                if(($ParssedDescendantStringrights[$position]         -match         ".*Allow
Modify.*|Allow        ReadAndExecute")-and($zaznam    -match    "Allow        FullControl")-
or(($ParssedDescendantStringrights[$position] -match "Allow  ReadAndExecute")-and($zaznam -
match "Allow  FullControl|Allow  Modify"))) # if the newer one has higher rights
                {
                        $ParssedDescendantStringrights.RemoveRange($position,1)
                        $RefParsingHashTable.Value.Remove($head)
                        $ParssedDescendantStringrights.add($zaznam)      |      foreach-object
{$RefParsingHashTable.Value.add($head,$_)}
                }
        }
        if(($ParssedDescendantStringrights  -ne  $null)  -and  ($ParssedDescendantStringrights
-ne ""))
        {
                return $ParssedDescendantStringrights
        }
}

function Groupchecker
{
        param($zaznam)
        $global:GroupCounter++
        if($zaznam -match "Testinggroup2" -and $Debug)
        {
```

```
                Write-Host "stop"
        }

        if(((($zaznam -notlike "*BUILTIN*")-and ($zaznam -notlike "*NT Autho*")-and ($zaznam
-notlike "*Deny*")) -or ($zaznam -eq "Everyone"))
        {
                #fixing descendant rigts
                [string]$TypeOfAccess=""
                if($zaznam -like "* FullControl")
                {
                        $TypeOfAccess="Full"
                        SendToLog -Dologu $("Full controll on " + $Descendant.FullName + "
by " + $(PreareACLRecord $zaznam))
                }
                elseif($zaznam -like "*ReadAndExecute*")
                {
                        $TypeOfAccess="Read"
                }
                else    #($zaznam -like "* Modify")
                {
                        $TypeOfAccess="Modify"
                }
                #Filtering rights description
                $zaznam=$zaznam -replace "\s+Allow.*|\s+Deny.*"
                if(($zaznam[2]  -ne  "=")-and  ($zaznam  -notmatch  "^S-\d-\d+-\d+-\d+-\d+-
\d+.*") -and ($zaznam -ne "Everyone")) #not Distingushed and not SID (so Netbios)
                {
                        $DomainNetbios=$zaznam.split("\")[0]
                        $GroupNetbios=$zaznam.split("\")[1]
                        $Netbios=$true  #if  the  netbios  is  true,  then  it  is  netbios  group
name, otherwise it is distinguished
                        if(-not($Global:DomainIndex.ContainsKey($DomainNetbios)))   #if   the
domain is unknown
                        {
                                $direntry=GetDomain $DomainNetbios
                                $Global:DomainIndex.add($DomainNetbios,$direntry)
                                $Global:DomainIndex.add($direntry,$direntry)  #  we  add  the
domain
                        }
                        if(-not($Global:GroupIndex.ContainsKey($GroupNetbios)))
                        {
                                #Ok lets turn it to distinguished (because it is unknown)
                                [string]$filtr="(cn="+ $GroupNetbios + ")"
                                [string]$entrypath="LDAP://"                                +
$Global:DomainIndex[$DomainNetbios]
                                $entry                                =               New-Object
System.DirectoryServices.DirectoryEntry($entrypath)
                                $objSearcher                    =                   New-Object
System.DirectoryServices.DirectorySearcher ($entry, "$filtr")
```

```
                                $objSearcher.SearchScope = "Subtree"
                                $objSearcher.PropertiesToLoad.Add("") | out-null
                                #[string]$GroupDisting=$objSearcher.FindOne()    |    foreach-
object {($_.Properties).Item("distinguishedName")}
                                $GroupDisting=$objSearcher.FindAll()      |      ForEach-Object
{$_.Path }
                                <#trap { if ($_.Exception -match ".*You cannot call a method
on a null-valued expression.*")
                            {   return $null}
                                continue
                        }#>
                                [ADSI]$GroupADSI=$GroupDisting           #checking    if    it
really is group
                                $GroupDisting=$GroupDisting -replace "LDAP://"
                                if ($GroupADSI.objectClass -match "group")
                                {
                                        $Global:GroupIndex.add($GroupNetbios,$GroupDisting)
                                }
                                #$GroupDisting=$GroupDisting -replace "LDAP://" -moved up
                        }
                        else #so the group is known, but needs to have additionall accesses
added
                        {
                                $GroupDisting=$Global:GroupIndex[$GroupNetbios]
                                [ADSI]$GroupADSI="LDAP://"+$GroupDisting
                        }
                }elseif($zaznam -eq "Everyone")
                {
                        $GroupDisting=$zaznam
                        $GroupADSI=$null
                        $GroupID= GroupSQL "Everyone" "Everyone"
                }
                elseif(($zaznam  -notmatch  "^S-\d-\d+-\d+-\d+-\d+-\d+.*")-and  ($zaznam  -ne
"Everyone")) #now we have distinguished
                {
                        $GroupDisting=$zaznam
                }
                elseif($zaznam -eq "CREATOR OWNER")
                {
                        $GroupDisting=$zaznam
                        $GroupADSI=$null
                        SendToLog -Dologu $("CREATOR OWNER on " + $Descendant.FullName)
                }
                else # it is failed  Unenumerated SID
                {
                        $GroupDisting=$zaznam
                        $GroupADSI=$null
                        SendToLog -Dologu $("Unenumerated SID on " + $Descendant.FullName)
                }
```

```
                if($zaznam[2] -eq "=") #only if it was distinugishd from beggining as
creating of ADSI object can take long
                {
                        [ADSI]$GroupADSI="LDAP://"+$GroupDisting        #checking    if    it
really is group
                }
                if ($GroupADSI.objectClass -match "group")
                {
                        $Global:GroupQueue.push($GroupDisting)  #we  need  to  enumerate  its
members even if it is knonw
                        if(-not($Global:GroupIndex.ContainsKey($GroupDisting)))   #the   group
is unknown, so we will add it to groups being listed
                        {
                                $Global:GroupIndex.add($GroupDisting,$GroupDisting)
                        }
                }

                #end of rights attached directly to the share
                #here we will start to work on enumerating groups in queue
                $global:memberslist=""
                while ($Global:GroupQueue.Count -gt 0) # if there is something in queue some
groups to add to database
                {
                        $GroupPATH=$Global:GroupQueue.pop()
                        if($GroupADSI.path -match $groupPath)
                        {
                                $Group=$GroupADSI
                        }
                        else
                        {
                                [ADSI]$Group="LDAP://"+$GroupPATH
                        }
                        $GroupName=$GroupPATH -replace ",DC=.*"
                        $GroupName= $GroupName -replace "CN="
                        $GroupName=$GroupName -replace ",OU=.*"
                        $GroupNote=$Group.properties.info;
                        $GroupDescription=$Group.properties.description;
                        #ok Let's parse group owner
                        [string]$OwnerNote=[Regex]::Matches($GroupNote,
"owner\S*\s*\w*\S*\w*", 'IgnoreCase') -replace "owner\S*\s*"
                        [string]$OwnerDescription=[Regex]::Matches($GroupDescription,
"owner\S*\s*\w*\S*\w*", 'IgnoreCase') -replace "owner\S*\s*"
                        # we have got the owner! oh yes! now we just need to put them into
database
                        #[String[]]$OwnerID
                        [String[]]$OwnerID=$null
                        #$OwnerID=OwnerSQL $OwnerNote $OwnerDescription
                        $OwnerID+=OwnerSQL $OwnerNote
                        $OwnerID+=OwnerSQL $OwnerDescription
```

```
                        #completed adding owner to database
                        #$members=$Group.members
                        $members=GetMembers $Group.distinguishedname

                        foreach($member in $members)
                        {
                                if($global:Debug)
                                {
                                        #Write-Host "Test test"
                                }
                                if(($member           -ne           $null)-and           (-not
$global:memberslist.contains($member)))
                                {
                                        $global:memberslist+="`n$member"
                                        $Global:GroupQueue.push($member)
                                }
                        }

                        #writing group into database
                        $GroupID= GroupSQL $GroupPATH $GroupName

                        #Creating association betwen share and group
                        GroupShareSQL $GroupID $ShareID $TypeofAccess

                        #creating association between Owner and group
                        foreach ($owner in $ownerID)
                        {
                                if(($owner -ne $null)-and($owner -ne ""))
                                {
                                        GroupOwnerSQL $GroupID $owner
                                }
                        }
                }
        }
}

function FolderChecker
{
        param($Descendant) #input is folder path of folder to be checked
        if(($Descendant.psiscontainer)-or($Descendant.Fullname    -eq    $Ancestor.Fullname))
#only folders are interesting
        {

        if(($descendant.fullname    -match    "\\\\prg-dc\\cz\\AppsData\\Helios\\t")    -and
($global:Debug -eq $false))
        {
            write-host "let the debug start"
                        $global:Debug=$true;
        }
```

```
                if($Descendant.psiscontainer)
                {
                        $global:foldercounter++
                }
                [string]$ExtensionAdress                                                =
$([string]$Descendant.Fullname).replace($global:WhereParam,"")
                $Depth = Select-string -InputObject $ExtensionAdress -AllMatches -Pattern
"\\" | % {$_.Matches.count } #detecting how deep are we
                $DescendantRights= GetRights $Descendant.FullName
                #Write-Host $Descendant.Fullname
                if($DescendantRights -eq "Acccess denied") #Handling lack of access, if we
can't get there, we won't touch it
                {
                        SendtoLog -Dologu $("Access denied on " + $Descendant.FullName)
                }
                else #if we have access on the rights on the folder
                {
                        if($Depth -lt $global:MaxDepthParam) #handling level of depth
                        {
                                $global:FolderQueue.push($Descendant)
                        }
                        if   (((Compare-ACL   $AncestorRights   $DescendantRights)-ne   $null)-
or($Descendant.Fullname -eq $global:WhereParam) ) #if the rights are not equal, then we
have to put it in the database and search groups and so on.
                        {
                                #Write-Host "Checking "$Descendant.Fullname
                                $ShareID=ShareSQL $Descendant.Fullname

                                #Analyzing groups on the share itself
                                $DescendantStringrights=$DescendantRights.Accesstostring
                                $DescendantStringrights=$DescendantStringrights.split("`n")
#spliting group record for specific share lines
                                #preparation of rights strings
                                $ParssedDescendantStringrights=New-Object
System.Collections.ArrayList  # an empty collection

        #$ParssedDescendantStringrights=[ref]$ParssedDescendantStringrights
                                $ParsingHashTable=@{}  #hashtable  for  parsing  of  security
records
                                $RefParsingHashTable=[ref]$ParsingHashTable
                                #$parsingcounter=0;

                                foreach($zaznam in $DescendantStringrights)
                                {
                                        $navratova=ACLupdater $zaznam $RefParsingHashTable
                                        if($navratova -ne $null)
                                        {
                                                $ParssedDescendantStringrights+=$navratova
                                        }
```

```
                                        <#else
                                        {
                                                Write-Host "WTF?"
                                        }#>
                                }
                                #TO DO, what about groups which are in database but no longer
on share?
                                Groupdeletor $ShareID $ParssedDescendantStringrights

                                foreach($zaznam   in   $ParssedDescendantStringrights)   #Here
starts enumerating of specific groups directly attached to share
                                {
                                        Groupchecker $zaznam
                                }
                        }
                }
        }
        else
        {
                $Global:FileCounter++
        }
}

function GroupDeletor
{ #this functions searches for groups in database attached on share and seeks any group
which anymore has no access.
        #first extracting from database
        param($ShareID, $Groups)

        $global:cmd.CommandText          ="SELECT          AccessManager.dbo.GroupToShare.ID,
AccessManager.dbo.[Group].Name              from              AccessManager.dbo.GroupToShare,
AccessManager.dbo.[Group]  where  AccessManager.dbo.GroupToShare.ShareID=`'$($ShareID)`'  and
AccessManager.dbo.[Group].ID=AccessManager.dbo.GroupToShare.GroupID"
        $global:SqlAdapter.SelectCommand = $global:cmd
        $DataSet = New-Object System.Data.DataSet
        $returnedrows=$global:SqlAdapter.Fill($DataSet)

        if($returnedrows -ne 0) #if the association doesn't exist
        {
                $groupIDs=@()
                $rows=$DataSet.Tables[0].Rows.count

                for($i=0;$i -lt $rows; $i++)
                {
                        $isitthere=$false
                        foreach($group in $groups)
                        {
                                if($DataSet.Tables[0].Rows[$i]["Name"]   -match   $($group   -
replace "\\","\\"))
```

```
                                {
                                        $isitthere=$true
                                }
                        }
                        if(-not $isitthere) #the group needs to be PURGED!
                        {
                                $groupIDs+=$DataSet.Tables[0].Rows[$i]["ID"]
                        }
                }
                foreach($id in $groupIDs) #Purging not connected groups
                {
                        $global:cmd.CommandText ="DELETE from AccessManager.dbo.GroupToShare
where ID=`'$($ID)`'"
                }
        }
}

function PreareACLRecord
{
        param([string]$zaznan)
        #this function needs to parse ACL record on folder, which is expected to be netbios
type
        $zaznam=$zaznam.split("\")[1]
        $zaznam=$zaznam -replace "( Allow| Deny).*"
        $zaznam=$zaznam.Trim()
        $zaznam
}

#defining global parameters
$Global:FolderQueue = New-Object System.Collections.Stack; #list of folders to check
$Global:GroupQueue = New-Object System.Collections.Stack; #list of folders to enumerate
$Global:GroupIndex=@{} #list of already enumeratd group (so we won't do it second time)
$Global:DomainIndex=@{} #list of known domains
[string]$Global:TheTextLog="" #what we send to log will be here as well....
$global:Debug=$false

#defining global parameters for log
$global:logname= $logpath + "\"+$(get-date | %{$global:jobname +"_"+ $_.year +"-"+ $_.month
+ "-" + $_.day + "-" + $_.hour +"."+ $_.minute +".txt"})

[System.IO.DirectoryInfo]$lokace=$global:WhereParam
$global:FolderQueue.push($lokace)

#Definition of SQL connection
$global:conn                                    =                                    New-Object
System.Data.SqlClient.SqlConnection("Server=czchowsint410,1525;Database=AccessManager;integ
rated security=SSPI")
$global:conn.Open()
$global:cmd = New-Object System.Data.SqlClient.SqlCommand
```

```
$global:cmd.connection = $global:conn
$global:SqlAdapter = New-Object System.Data.SqlClient.SqlDataAdapter
$global:SqlAdapter.SelectCommand = $global:cmd
$global:FileCounter=0
$global:FolderCounter=0
$global:GroupCounter=0
[string]$global:memberslist=""
$Counter=0

sendtolog -Dologu "Starting at"
$global:startdate=$(Get-Date)
sendtolog -Dologu $global:startdate

while ($global:FolderQueue.Count -gt 0) # if there is something in queue some folders to
check
{
        if ($true) #($Counter -gt 0)
        {
                $Ancestor=$global:FolderQueue.pop()
                $Descendants= Get-ChildItem $Ancestor.Fullname #geting child folders
        }
        else
        {
                $Ancestor= New-Object System.IO.DirectoryInfo ($global:WhereParam)
                #$Ancestor.fullname=$global:WhereParam
                $Descendants=@(New-Object System.IO.DirectoryInfo ($global:WhereParam))
        }
        $AncestorRights= GetRights $Ancestor.Fullname #getting acces rightsl
        foreach($Descendant in $Descendants)
        {
                Folderchecker $Descendant
        }
        $Counter++
}
$global:enddate=Get-Date
sendtolog -Dologu "Ended at $global:enddate"
$duration=$global:enddate.subtract($global:startdate)
sendtolog -Dologu $("Lasted: $duration")
SendToLog -Dologu $("Processed $global:Foldercounter folders, $global:Filecounter files,
and $global:GroupCounter AD Objects")
$conn.Close()
```

CSS soubory

Query.css

```
/*! jQuery UI - v1.10.1 - 2013-02-15
* http://jqueryui.com
* Includes: jquery.ui.core.css, jquery.ui.accordion.css, jquery.ui.autocomplete.css,
jquery.ui.button.css, jquery.ui.datepicker.css, jquery.ui.dialog.css, jquery.ui.menu.css,
jquery.ui.progressbar.css, jquery.ui.resizable.css, jquery.ui.selectable.css, jquery.ui.slider.css,
jquery.ui.spinner.css, jquery.ui.tabs.css, jquery.ui.tooltip.css
* Copyright (c) 2013 jQuery Foundation and other contributors Licensed MIT */

/* Layout helpers
----------------------------------*/
.ui-helper-hidden {
	display: none;
}
.ui-helper-hidden-accessible {
	border: 0;
	clip: rect(0 0 0 0);
	height: 1px;
	margin: -1px;
	overflow: hidden;
	padding: 0;
	position: absolute;
	width: 1px;
}
.ui-helper-reset {
	margin: 0;
	padding: 0;
	border: 0;
	outline: 0;
	line-height: 1.3;
	text-decoration: none;
	font-size: 100%;
	list-style: none;
}
.ui-helper-clearfix:before,
.ui-helper-clearfix:after {
	content: "";
	display: table;
	border-collapse: collapse;
}
.ui-helper-clearfix:after {
	clear: both;
}
.ui-helper-clearfix {
	min-height: 0; /* support: IE7 */
}
.ui-helper-zfix {
	width: 100%;
	height: 100%;
	top: 0;
	left: 0;
	position: absolute;
	opacity: 0;
	filter:Alpha(Opacity=0);
}

.ui-front {
	z-index: 100;
}


/* Interaction Cues
----------------------------------*/
.ui-state-disabled {
	cursor: default !important;
}
```

```
/* Icons
----------------------------------*/

/* states and images */
.ui-icon {
	display: block;
	text-indent: -99999px;
	overflow: hidden;
	background-repeat: no-repeat;
}


/* Misc visuals
----------------------------------*/

/* Overlays */
.ui-widget-overlay {
	position: fixed;
	top: 0;
	left: 0;
	width: 100%;
	height: 100%;
}
.ui-accordion .ui-accordion-header {
	display: block;
	cursor: pointer;
	position: relative;
	margin-top: 2px;
	padding: .5em .5em .5em .7em;
	min-height: 0; /* support: IE7 */
}
.ui-accordion .ui-accordion-icons {
	padding-left: 2.2em;
}
.ui-accordion .ui-accordion-noicons {
	padding-left: .7em;
}
.ui-accordion .ui-accordion-icons .ui-accordion-icons {
	padding-left: 2.2em;
}
.ui-accordion .ui-accordion-header .ui-accordion-header-icon {
	position: absolute;
	left: .5em;
	top: 50%;
	margin-top: -8px;
}
.ui-accordion .ui-accordion-content {
	padding: 1em 2.2em;
	border-top: 0;
	overflow: auto;
}
.ui-autocomplete {
	position: absolute;
	top: 0;
	left: 0;
	cursor: default;
}
.ui-button {
	display: inline-block;
	position: relative;
	padding: 0;
	line-height: normal;
	margin-right: .1em;
	cursor: pointer;
	vertical-align: middle;
	text-align: center;
	overflow: visible; /* removes extra width in IE */
}
.ui-button,
```

```
.ui-button:link,
.ui-button:visited,
.ui-button:hover,
.ui-button:active {
	text-decoration: none;
}
/* to make room for the icon, a width needs to be set here */
.ui-button-icon-only {
	width: 2.2em;
}
/* button elements seem to need a little more width */
button.ui-button-icon-only {
	width: 2.4em;
}
.ui-button-icons-only {
	width: 3.4em;
}
button.ui-button-icons-only {
	width: 3.7em;
}

/* button text element */
.ui-button .ui-button-text {
	display: block;
	line-height: normal;
}
.ui-button-text-only .ui-button-text {
	padding: .4em 1em;
}
.ui-button-icon-only .ui-button-text,
.ui-button-icons-only .ui-button-text {
	padding: .4em;
	text-indent: -9999999px;
}
.ui-button-text-icon-primary .ui-button-text,
.ui-button-text-icons .ui-button-text {
	padding: .4em 1em .4em 2.1em;
}
.ui-button-text-icon-secondary .ui-button-text,
.ui-button-text-icons .ui-button-text {
	padding: .4em 2.1em .4em 1em;
}
.ui-button-text-icons .ui-button-text {
	padding-left: 2.1em;
	padding-right: 2.1em;
}
/* no icon support for input elements, provide padding by default */
input.ui-button {
	padding: .4em 1em;
}

/* button icon element(s) */
.ui-button-icon-only .ui-icon,
.ui-button-text-icon-primary .ui-icon,
.ui-button-text-icon-secondary .ui-icon,
.ui-button-text-icons .ui-icon,
.ui-button-icons-only .ui-icon {
	position: absolute;
	top: 50%;
	margin-top: -8px;
}
.ui-button-icon-only .ui-icon {
	left: 50%;
	margin-left: -8px;
}
.ui-button-text-icon-primary .ui-button-icon-primary,
.ui-button-text-icons .ui-button-icon-primary,
.ui-button-icons-only .ui-button-icon-primary {
	left: .5em;
}
.ui-button-text-icon-secondary .ui-button-icon-secondary,
```

```
.ui-button-text-icons .ui-button-icon-secondary,
.ui-button-icons-only .ui-button-icon-secondary {
	right: .5em;
}

/* button sets */
.ui-buttonset {
	margin-right: 7px;
}
.ui-buttonset .ui-button {
	margin-left: 0;
	margin-right: -.3em;
}

/* workarounds */
/* reset extra padding in Firefox, see h5bp.com/l */
input.ui-button::-moz-focus-inner,
button.ui-button::-moz-focus-inner {
	border: 0;
	padding: 0;
}
.ui-datepicker {
	width: 17em;
	padding: .2em .2em 0;
	display: none;
}
.ui-datepicker .ui-datepicker-header {
	position: relative;
	padding: .2em 0;
}
.ui-datepicker .ui-datepicker-prev,
.ui-datepicker .ui-datepicker-next {
	position: absolute;
	top: 2px;
	width: 1.8em;
	height: 1.8em;
}
.ui-datepicker .ui-datepicker-prev-hover,
.ui-datepicker .ui-datepicker-next-hover {
	top: 1px;
}
.ui-datepicker .ui-datepicker-prev {
	left: 2px;
}
.ui-datepicker .ui-datepicker-next {
	right: 2px;
}
.ui-datepicker .ui-datepicker-prev-hover {
	left: 1px;
}
.ui-datepicker .ui-datepicker-next-hover {
	right: 1px;
}
.ui-datepicker .ui-datepicker-prev span,
.ui-datepicker .ui-datepicker-next span {
	display: block;
	position: absolute;
	left: 50%;
	margin-left: -8px;
	top: 50%;
	margin-top: -8px;
}
.ui-datepicker .ui-datepicker-title {
	margin: 0 2.3em;
	line-height: 1.8em;
	text-align: center;
}
.ui-datepicker .ui-datepicker-title select {
	font-size: 1em;
	margin: 1px 0;
}
```

```
.ui-datepicker select.ui-datepicker-month-year {
	width: 100%;
}
.ui-datepicker select.ui-datepicker-month,
.ui-datepicker select.ui-datepicker-year {
	width: 49%;
}
.ui-datepicker table {
	width: 100%;
	font-size: .9em;
	border-collapse: collapse;
	margin: 0 0 .4em;
}
.ui-datepicker th {
	padding: .7em .3em;
	text-align: center;
	font-weight: bold;
	border: 0;
}
.ui-datepicker td {
	border: 0;
	padding: 1px;
}
.ui-datepicker td span,
.ui-datepicker td a {
	display: block;
	padding: .2em;
	text-align: right;
	text-decoration: none;
}
.ui-datepicker .ui-datepicker-buttonpane {
	background-image: none;
	margin: .7em 0 0 0;
	padding: 0 .2em;
	border-left: 0;
	border-right: 0;
	border-bottom: 0;
}
.ui-datepicker .ui-datepicker-buttonpane button {
	float: right;
	margin: .5em .2em .4em;
	cursor: pointer;
	padding: .2em .6em .3em .6em;
	width: auto;
	overflow: visible;
}
.ui-datepicker .ui-datepicker-buttonpane button.ui-datepicker-current {
	float: left;
}

/* with multiple calendars */
.ui-datepicker.ui-datepicker-multi {
	width: auto;
}
.ui-datepicker-multi .ui-datepicker-group {
	float: left;
}
.ui-datepicker-multi .ui-datepicker-group table {
	width: 95%;
	margin: 0 auto .4em;
}
.ui-datepicker-multi-2 .ui-datepicker-group {
	width: 50%;
}
.ui-datepicker-multi-3 .ui-datepicker-group {
	width: 33.3%;
}
.ui-datepicker-multi-4 .ui-datepicker-group {
	width: 25%;
}
.ui-datepicker-multi .ui-datepicker-group-last .ui-datepicker-header,
```

```
.ui-datepicker-multi .ui-datepicker-group-middle .ui-datepicker-header {
	border-left-width: 0;
}
.ui-datepicker-multi .ui-datepicker-buttonpane {
	clear: left;
}
.ui-datepicker-row-break {
	clear: both;
	width: 100%;
	font-size: 0;
}

/* RTL support */
.ui-datepicker-rtl {
	direction: rtl;
}
.ui-datepicker-rtl .ui-datepicker-prev {
	right: 2px;
	left: auto;
}
.ui-datepicker-rtl .ui-datepicker-next {
	left: 2px;
	right: auto;
}
.ui-datepicker-rtl .ui-datepicker-prev:hover {
	right: 1px;
	left: auto;
}
.ui-datepicker-rtl .ui-datepicker-next:hover {
	left: 1px;
	right: auto;
}
.ui-datepicker-rtl .ui-datepicker-buttonpane {
	clear: right;
}
.ui-datepicker-rtl .ui-datepicker-buttonpane button {
	float: left;
}
.ui-datepicker-rtl .ui-datepicker-buttonpane button.ui-datepicker-current,
.ui-datepicker-rtl .ui-datepicker-group {
	float: right;
}
.ui-datepicker-rtl .ui-datepicker-group-last .ui-datepicker-header,
.ui-datepicker-rtl .ui-datepicker-group-middle .ui-datepicker-header {
	border-right-width: 0;
	border-left-width: 1px;
}
.ui-dialog {
	position: absolute;
	top: 0;
	left: 0;
	padding: .2em;
	outline: 0;
}
.ui-dialog .ui-dialog-titlebar {
	padding: .4em 1em;
	position: relative;
}
.ui-dialog .ui-dialog-title {
	float: left;
	margin: .1em 0;
	white-space: nowrap;
	width: 90%;
	overflow: hidden;
	text-overflow: ellipsis;
}
.ui-dialog .ui-dialog-titlebar-close {
	position: absolute;
	right: .3em;
	top: 50%;
	width: 21px;
```

```
        margin: -10px 0 0 0;
        padding: 1px;
        height: 20px;
}
.ui-dialog .ui-dialog-content {
        position: relative;
        border: 0;
        padding: .5em 1em;
        background: none;
        overflow: auto;
}
.ui-dialog .ui-dialog-buttonpane {
        text-align: left;
        border-width: 1px 0 0 0;
        background-image: none;
        margin-top: .5em;
        padding: .3em 1em .5em .4em;
}
.ui-dialog .ui-dialog-buttonpane .ui-dialog-buttonset {
        float: right;
}
.ui-dialog .ui-dialog-buttonpane button {
        margin: .5em .4em .5em 0;
        cursor: pointer;
}
.ui-dialog .ui-resizable-se {
        width: 12px;
        height: 12px;
        right: -5px;
        bottom: -5px;
        background-position: 16px 16px;
}
.ui-draggable .ui-dialog-titlebar {
        cursor: move;
}
.ui-menu {
        list-style: none;
        padding: 2px;
        margin: 0;
        display: block;
        outline: none;
}
.ui-menu .ui-menu {
        margin-top: -3px;
        position: absolute;
}
.ui-menu .ui-menu-item {
        margin: 0;
        padding: 0;
        width: 100%;
}
.ui-menu .ui-menu-divider {
        margin: 5px -2px 5px -2px;
        height: 0;
        font-size: 0;
        line-height: 0;
        border-width: 1px 0 0 0;
}
.ui-menu .ui-menu-item a {
        text-decoration: none;
        display: block;
        padding: 2px .4em;
        line-height: 1.5;
        min-height: 0; /* support: IE7 */
        font-weight: normal;
}
.ui-menu .ui-menu-item a.ui-state-focus,
.ui-menu .ui-menu-item a.ui-state-active {
        font-weight: normal;
        margin: -1px;
}
```

```
.ui-menu .ui-state-disabled {
	font-weight: normal;
	margin: .4em 0 .2em;
	line-height: 1.5;
}
.ui-menu .ui-state-disabled a {
	cursor: default;
}

/* icon support */
.ui-menu-icons {
	position: relative;
}
.ui-menu-icons .ui-menu-item a {
	position: relative;
	padding-left: 2em;
}

/* left-aligned */
.ui-menu .ui-icon {
	position: absolute;
	top: .2em;
	left: .2em;
}

/* right-aligned */
.ui-menu .ui-menu-icon {
	position: static;
	float: right;
}
.ui-progressbar {
	height: 2em;
	text-align: left;
	overflow: hidden;
}
.ui-progressbar .ui-progressbar-value {
	margin: -1px;
	height: 100%;
}
.ui-progressbar .ui-progressbar-overlay {
	background: url("images/animated-overlay.gif");
	height: 100%;
	filter: alpha(opacity=25);
	opacity: 0.25;
}
.ui-progressbar-indeterminate .ui-progressbar-value {
	background-image: none;
}
.ui-resizable {
	position: relative;
}
.ui-resizable-handle {
	position: absolute;
	font-size: 0.1px;
	display: block;
}
.ui-resizable-disabled .ui-resizable-handle,
.ui-resizable-autohide .ui-resizable-handle {
	display: none;
}
.ui-resizable-n {
	cursor: n-resize;
	height: 7px;
	width: 100%;
	top: -5px;
	left: 0;
}
.ui-resizable-s {
	cursor: s-resize;
	height: 7px;
```

```
        width: 100%;
        bottom: -5px;
        left: 0;
}
.ui-resizable-e {
        cursor: e-resize;
        width: 7px;
        right: -5px;
        top: 0;
        height: 100%;
}
.ui-resizable-w {
        cursor: w-resize;
        width: 7px;
        left: -5px;
        top: 0;
        height: 100%;
}
.ui-resizable-se {
        cursor: se-resize;
        width: 12px;
        height: 12px;
        right: 1px;
        bottom: 1px;
}
.ui-resizable-sw {
        cursor: sw-resize;
        width: 9px;
        height: 9px;
        left: -5px;
        bottom: -5px;
}
.ui-resizable-nw {
        cursor: nw-resize;
        width: 9px;
        height: 9px;
        left: -5px;
        top: -5px;
}
.ui-resizable-ne {
        cursor: ne-resize;
        width: 9px;
        height: 9px;
        right: -5px;
        top: -5px;
}
.ui-selectable-helper {
        position: absolute;
        z-index: 100;
        border: 1px dotted black;
}
.ui-slider {
        position: relative;
        text-align: left;
}
.ui-slider .ui-slider-handle {
        position: absolute;
        z-index: 2;
        width: 1.2em;
        height: 1.2em;
        cursor: default;
}
.ui-slider .ui-slider-range {
        position: absolute;
        z-index: 1;
        font-size: .7em;
        display: block;
        border: 0;
        background-position: 0 0;
}
```

```
/* For IE8 - See #6727 */
.ui-slider.ui-state-disabled .ui-slider-handle,
.ui-slider.ui-state-disabled .ui-slider-range {
        filter: inherit;
}

.ui-slider-horizontal {
        height: .8em;
}
.ui-slider-horizontal .ui-slider-handle {
        top: -.3em;
        margin-left: -.6em;
}
.ui-slider-horizontal .ui-slider-range {
        top: 0;
        height: 100%;
}
.ui-slider-horizontal .ui-slider-range-min {
        left: 0;
}
.ui-slider-horizontal .ui-slider-range-max {
        right: 0;
}

.ui-slider-vertical {
        width: .8em;
        height: 100px;
}
.ui-slider-vertical .ui-slider-handle {
        left: -.3em;
        margin-left: 0;
        margin-bottom: -.6em;
}
.ui-slider-vertical .ui-slider-range {
        left: 0;
        width: 100%;
}
.ui-slider-vertical .ui-slider-range-min {
        bottom: 0;
}
.ui-slider-vertical .ui-slider-range-max {
        top: 0;
}
.ui-spinner {
        position: relative;
        display: inline-block;
        overflow: hidden;
        padding: 0;
        vertical-align: middle;
}
.ui-spinner-input {
        border: none;
        background: none;
        color: inherit;
        padding: 0;
        margin: .2em 0;
        vertical-align: middle;
        margin-left: .4em;
        margin-right: 22px;
}
.ui-spinner-button {
        width: 16px;
        height: 50%;
        font-size: .5em;
        padding: 0;
        margin: 0;
        text-align: center;
        position: absolute;
        cursor: default;
        display: block;
        overflow: hidden;
```

```
        right: 0;
}
/* more specificity required here to overide default borders */
.ui-spinner a.ui-spinner-button {
        border-top: none;
        border-bottom: none;
        border-right: none;
}
/* vertical centre icon */
.ui-spinner .ui-icon {
        position: absolute;
        margin-top: -8px;
        top: 50%;
        left: 0;
}
.ui-spinner-up {
        top: 0;
}
.ui-spinner-down {
        bottom: 0;
}

/* TR overrides */
.ui-spinner .ui-icon-triangle-1-s {
        /* need to fix icons sprite */
        background-position: -65px -16px;
}
.ui-tabs {
        position: relative;/* position: relative prevents IE scroll bug (element with position:
relative inside container with overflow: auto appear as "fixed") */
        padding: .2em;
}
.ui-tabs .ui-tabs-nav {
        margin: 0;
        padding: .2em .2em 0;
}
.ui-tabs .ui-tabs-nav li {
        list-style: none;
        float: left;
        position: relative;
        top: 0;
        margin: 1px .2em 0 0;
        border-bottom: 0;
        padding: 0;
        white-space: nowrap;
}
.ui-tabs .ui-tabs-nav li a {
        float: left;
        padding: .5em 1em;
        text-decoration: none;
}
.ui-tabs .ui-tabs-nav li.ui-tabs-active {
        margin-bottom: -1px;
        padding-bottom: 1px;
}
.ui-tabs .ui-tabs-nav li.ui-tabs-active a,
.ui-tabs .ui-tabs-nav li.ui-state-disabled a,
.ui-tabs .ui-tabs-nav li.ui-tabs-loading a {
        cursor: text;
}
.ui-tabs .ui-tabs-nav li a, /* first selector in group seems obsolete, but required to overcome bug
in Opera applying cursor: text overall if defined elsewhere... */
.ui-tabs-collapsible .ui-tabs-nav li.ui-tabs-active a {
        cursor: pointer;
}
.ui-tabs .ui-tabs-panel {
        display: block;
        border-width: 0;
        padding: 1em 1.4em;
        background: none;
}
```

```
.ui-tooltip {
        padding: 8px;
        position: absolute;
        z-index: 9999;
        max-width: 300px;
        -webkit-box-shadow: 0 0 5px #aaa;
        box-shadow: 0 0 5px #aaa;
}
body .ui-tooltip {
        border-width: 2px;
}

/* Component containers
----------------------------------*/
.ui-widget {
        font-family: Verdana,Arial,sans-serif/*{ffDefault}*/;
        font-size: 1.1em/*{fsDefault}*/;
}
.ui-widget .ui-widget {
        font-size: 1em;
}
.ui-widget input,
.ui-widget select,
.ui-widget textarea,
.ui-widget button {
        font-family: Verdana,Arial,sans-serif/*{ffDefault}*/;
        font-size: 1em;
}
.ui-widget-content {
        border: 1px solid #aaaaaa/*{borderColorContent}*/;
        background: #ffffff/*{bgColorContent}*/ url(images/ui-
bg_flat_75_ffffff_40x100.png)/*{bgImgUrlContent}*/ 50%/*{bgContentXPos}*/ 50%/*{bgContentYPos}*/
repeat-x/*{bgContentRepeat}*/;
        color: #222222/*{fcContent}*/;
}
.ui-widget-content a {
        color: #222222/*{fcContent}*/;
}
.ui-widget-header {
        border: 1px solid #aaaaaa/*{borderColorHeader}*/;
        background: #cccccc/*{bgColorHeader}*/ url(images/ui-bg_highlight-
soft_75_cccccc_1x100.png)/*{bgImgUrlHeader}*/ 50%/*{bgHeaderXPos}*/ 50%/*{bgHeaderYPos}*/ repeat-
x/*{bgHeaderRepeat}*/;
        color: #222222/*{fcHeader}*/;
        font-weight: bold;
}
.ui-widget-header a {
        color: #222222/*{fcHeader}*/;
}

/* Interaction states
----------------------------------*/
.ui-state-default,
.ui-widget-content .ui-state-default,
.ui-widget-header .ui-state-default {
        border: 1px solid #d3d3d3/*{borderColorDefault}*/;
        background: #e6e6e6/*{bgColorDefault}*/ url(images/ui-
bg_glass_75_e6e6e6_1x400.png)/*{bgImgUrlDefault}*/ 50%/*{bgDefaultXPos}*/ 50%/*{bgDefaultYPos}*/
repeat-x/*{bgDefaultRepeat}*/;
        font-weight: normal/*{fwDefault}*/;
        color: #555555/*{fcDefault}*/;
}
.ui-state-default a,
.ui-state-default a:link,
.ui-state-default a:visited {
        color: #555555/*{fcDefault}*/;
        text-decoration: none;
}
.ui-state-hover,
.ui-widget-content .ui-state-hover,
.ui-widget-header .ui-state-hover,
```

```
.ui-state-focus,
.ui-widget-content .ui-state-focus,
.ui-widget-header .ui-state-focus {
        border: 1px solid #999999/*{borderColorHover}*/;
        background: #dadada/*{bgColorHover}*/ url(images/ui-
bg_glass_75_dadada_1x400.png)/*{bgImgUrlHover}*/ 50%/*{bgHoverXPos}*/ 50%/*{bgHoverYPos}*/ repeat-
x/*{bgHoverRepeat}*/;
        font-weight: normal/*{fwDefault}*/;
        color: #212121/*{fcHover}*/;
}
.ui-state-hover a,
.ui-state-hover a:hover,
.ui-state-hover a:link,
.ui-state-hover a:visited {
        color: #212121/*{fcHover}*/;
        text-decoration: none;
}
.ui-state-active,
.ui-widget-content .ui-state-active,
.ui-widget-header .ui-state-active {
        border: 1px solid #aaaaaa/*{borderColorActive}*/;
        background: #ffffff/*{bgColorActive}*/ url(images/ui-
bg_glass_65_ffffff_1x400.png)/*{bgImgUrlActive}*/ 50%/*{bgActiveXPos}*/ 50%/*{bgActiveYPos}*/
repeat-x/*{bgActiveRepeat}*/;
        font-weight: normal/*{fwDefault}*/;
        color: #212121/*{fcActive}*/;
}
.ui-state-active a,
.ui-state-active a:link,
.ui-state-active a:visited {
        color: #212121/*{fcActive}*/;
        text-decoration: none;
}

/* Interaction Cues
----------------------------------*/
.ui-state-highlight,
.ui-widget-content .ui-state-highlight,
.ui-widget-header .ui-state-highlight {
        border: 1px solid #fcefa1/*{borderColorHighlight}*/;
        background: #fbf9ee/*{bgColorHighlight}*/ url(images/ui-
bg_glass_55_fbf9ee_1x400.png)/*{bgImgUrlHighlight}*/ 50%/*{bgHighlightXPos}*/
50%/*{bgHighlightYPos}*/ repeat-x/*{bgHighlightRepeat}*/;
        color: #363636/*{fcHighlight}*/;
}
.ui-state-highlight a,
.ui-widget-content .ui-state-highlight a,
.ui-widget-header .ui-state-highlight a {
        color: #363636/*{fcHighlight}*/;
}
.ui-state-error,
.ui-widget-content .ui-state-error,
.ui-widget-header .ui-state-error {
        border: 1px solid #cd0a0a/*{borderColorError}*/;
        background: #fef1ec/*{bgColorError}*/ url(images/ui-
bg_glass_95_fef1ec_1x400.png)/*{bgImgUrlError}*/ 50%/*{bgErrorXPos}*/ 50%/*{bgErrorYPos}*/ repeat-
x/*{bgErrorRepeat}*/;
        color: #cd0a0a/*{fcError}*/;
}
.ui-state-error a,
.ui-widget-content .ui-state-error a,
.ui-widget-header .ui-state-error a {
        color: #cd0a0a/*{fcError}*/;
}
.ui-state-error-text,
.ui-widget-content .ui-state-error-text,
.ui-widget-header .ui-state-error-text {
        color: #cd0a0a/*{fcError}*/;
}
.ui-priority-primary,
.ui-widget-content .ui-priority-primary,
```

```
.ui-widget-header .ui-priority-primary {
	font-weight: bold;
}
.ui-priority-secondary,
.ui-widget-content .ui-priority-secondary,
.ui-widget-header .ui-priority-secondary {
	opacity: .7;
	filter:Alpha(Opacity=70);
	font-weight: normal;
}
.ui-state-disabled,
.ui-widget-content .ui-state-disabled,
.ui-widget-header .ui-state-disabled {
	opacity: .35;
	filter:Alpha(Opacity=35);
	background-image: none;
}
.ui-state-disabled .ui-icon {
	filter:Alpha(Opacity=35); /* For IE8 - See #6059 */
}

/* Icons
----------------------------------*/

/* states and images */
.ui-icon {
	width: 16px;
	height: 16px;
	background-position: 16px 16px;
}
.ui-icon,
.ui-widget-content .ui-icon {
	background-image: url(images/ui-icons_222222_256x240.png)/*{iconsContent}*/;
}
.ui-widget-header .ui-icon {
	background-image: url(images/ui-icons_222222_256x240.png)/*{iconsHeader}*/;
}
.ui-state-default .ui-icon {
	background-image: url(images/ui-icons_888888_256x240.png)/*{iconsDefault}*/;
}
.ui-state-hover .ui-icon,
.ui-state-focus .ui-icon {
	background-image: url(images/ui-icons_454545_256x240.png)/*{iconsHover}*/;
}
.ui-state-active .ui-icon {
	background-image: url(images/ui-icons_454545_256x240.png)/*{iconsActive}*/;
}
.ui-state-highlight .ui-icon {
	background-image: url(images/ui-icons_2e83ff_256x240.png)/*{iconsHighlight}*/;
}
.ui-state-error .ui-icon,
.ui-state-error-text .ui-icon {
	background-image: url(images/ui-icons_cd0a0a_256x240.png)/*{iconsError}*/;
}

/* positioning */
.ui-icon-carat-1-n { background-position: 0 0; }
.ui-icon-carat-1-ne { background-position: -16px 0; }
.ui-icon-carat-1-e { background-position: -32px 0; }
.ui-icon-carat-1-se { background-position: -48px 0; }
.ui-icon-carat-1-s { background-position: -64px 0; }
.ui-icon-carat-1-sw { background-position: -80px 0; }
.ui-icon-carat-1-w { background-position: -96px 0; }
.ui-icon-carat-1-nw { background-position: -112px 0; }
.ui-icon-carat-2-n-s { background-position: -128px 0; }
.ui-icon-carat-2-e-w { background-position: -144px 0; }
.ui-icon-triangle-1-n { background-position: 0 -16px; }
.ui-icon-triangle-1-ne { background-position: -16px -16px; }
.ui-icon-triangle-1-e { background-position: -32px -16px; }
.ui-icon-triangle-1-se { background-position: -48px -16px; }
.ui-icon-triangle-1-s { background-position: -64px -16px; }
```

```
.ui-icon-triangle-1-sw { background-position: -80px -16px; }
.ui-icon-triangle-1-w { background-position: -96px -16px; }
.ui-icon-triangle-1-nw { background-position: -112px -16px; }
.ui-icon-triangle-2-n-s { background-position: -128px -16px; }
.ui-icon-triangle-2-e-w { background-position: -144px -16px; }
.ui-icon-arrow-1-n { background-position: 0 -32px; }
.ui-icon-arrow-1-ne { background-position: -16px -32px; }
.ui-icon-arrow-1-e { background-position: -32px -32px; }
.ui-icon-arrow-1-se { background-position: -48px -32px; }
.ui-icon-arrow-1-s { background-position: -64px -32px; }
.ui-icon-arrow-1-sw { background-position: -80px -32px; }
.ui-icon-arrow-1-w { background-position: -96px -32px; }
.ui-icon-arrow-1-nw { background-position: -112px -32px; }
.ui-icon-arrow-2-n-s { background-position: -128px -32px; }
.ui-icon-arrow-2-ne-sw { background-position: -144px -32px; }
.ui-icon-arrow-2-e-w { background-position: -160px -32px; }
.ui-icon-arrow-2-se-nw { background-position: -176px -32px; }
.ui-icon-arrowstop-1-n { background-position: -192px -32px; }
.ui-icon-arrowstop-1-e { background-position: -208px -32px; }
.ui-icon-arrowstop-1-s { background-position: -224px -32px; }
.ui-icon-arrowstop-1-w { background-position: -240px -32px; }
.ui-icon-arrowthick-1-n { background-position: 0 -48px; }
.ui-icon-arrowthick-1-ne { background-position: -16px -48px; }
.ui-icon-arrowthick-1-e { background-position: -32px -48px; }
.ui-icon-arrowthick-1-se { background-position: -48px -48px; }
.ui-icon-arrowthick-1-s { background-position: -64px -48px; }
.ui-icon-arrowthick-1-sw { background-position: -80px -48px; }
.ui-icon-arrowthick-1-w { background-position: -96px -48px; }
.ui-icon-arrowthick-1-nw { background-position: -112px -48px; }
.ui-icon-arrowthick-2-n-s { background-position: -128px -48px; }
.ui-icon-arrowthick-2-ne-sw { background-position: -144px -48px; }
.ui-icon-arrowthick-2-e-w { background-position: -160px -48px; }
.ui-icon-arrowthick-2-se-nw { background-position: -176px -48px; }
.ui-icon-arrowthickstop-1-n { background-position: -192px -48px; }
.ui-icon-arrowthickstop-1-e { background-position: -208px -48px; }
.ui-icon-arrowthickstop-1-s { background-position: -224px -48px; }
.ui-icon-arrowthickstop-1-w { background-position: -240px -48px; }
.ui-icon-arrowreturnthick-1-w { background-position: 0 -64px; }
.ui-icon-arrowreturnthick-1-n { background-position: -16px -64px; }
.ui-icon-arrowreturnthick-1-e { background-position: -32px -64px; }
.ui-icon-arrowreturnthick-1-s { background-position: -48px -64px; }
.ui-icon-arrowreturn-1-w { background-position: -64px -64px; }
.ui-icon-arrowreturn-1-n { background-position: -80px -64px; }
.ui-icon-arrowreturn-1-e { background-position: -96px -64px; }
.ui-icon-arrowreturn-1-s { background-position: -112px -64px; }
.ui-icon-arrowrefresh-1-w { background-position: -128px -64px; }
.ui-icon-arrowrefresh-1-n { background-position: -144px -64px; }
.ui-icon-arrowrefresh-1-e { background-position: -160px -64px; }
.ui-icon-arrowrefresh-1-s { background-position: -176px -64px; }
.ui-icon-arrow-4 { background-position: 0 -80px; }
.ui-icon-arrow-4-diag { background-position: -16px -80px; }
.ui-icon-extlink { background-position: -32px -80px; }
.ui-icon-newwin { background-position: -48px -80px; }
.ui-icon-refresh { background-position: -64px -80px; }
.ui-icon-shuffle { background-position: -80px -80px; }
.ui-icon-transfer-e-w { background-position: -96px -80px; }
.ui-icon-transferthick-e-w { background-position: -112px -80px; }
.ui-icon-folder-collapsed { background-position: 0 -96px; }
.ui-icon-folder-open { background-position: -16px -96px; }
.ui-icon-document { background-position: -32px -96px; }
.ui-icon-document-b { background-position: -48px -96px; }
.ui-icon-note { background-position: -64px -96px; }
.ui-icon-mail-closed { background-position: -80px -96px; }
.ui-icon-mail-open { background-position: -96px -96px; }
.ui-icon-suitcase { background-position: -112px -96px; }
.ui-icon-comment { background-position: -128px -96px; }
.ui-icon-person { background-position: -144px -96px; }
.ui-icon-print { background-position: -160px -96px; }
.ui-icon-trash { background-position: -176px -96px; }
.ui-icon-locked { background-position: -192px -96px; }
.ui-icon-unlocked { background-position: -208px -96px; }
```

```
.ui-icon-bookmark { background-position: -224px -96px; }
.ui-icon-tag { background-position: -240px -96px; }
.ui-icon-home { background-position: 0 -112px; }
.ui-icon-flag { background-position: -16px -112px; }
.ui-icon-calendar { background-position: -32px -112px; }
.ui-icon-cart { background-position: -48px -112px; }
.ui-icon-pencil { background-position: -64px -112px; }
.ui-icon-clock { background-position: -80px -112px; }
.ui-icon-disk { background-position: -96px -112px; }
.ui-icon-calculator { background-position: -112px -112px; }
.ui-icon-zoomin { background-position: -128px -112px; }
.ui-icon-zoomout { background-position: -144px -112px; }
.ui-icon-search { background-position: -160px -112px; }
.ui-icon-wrench { background-position: -176px -112px; }
.ui-icon-gear { background-position: -192px -112px; }
.ui-icon-heart { background-position: -208px -112px; }
.ui-icon-star { background-position: -224px -112px; }
.ui-icon-link { background-position: -240px -112px; }
.ui-icon-cancel { background-position: 0 -128px; }
.ui-icon-plus { background-position: -16px -128px; }
.ui-icon-plusthick { background-position: -32px -128px; }
.ui-icon-minus { background-position: -48px -128px; }
.ui-icon-minusthick { background-position: -64px -128px; }
.ui-icon-close { background-position: -80px -128px; }
.ui-icon-closethick { background-position: -96px -128px; }
.ui-icon-key { background-position: -112px -128px; }
.ui-icon-lightbulb { background-position: -128px -128px; }
.ui-icon-scissors { background-position: -144px -128px; }
.ui-icon-clipboard { background-position: -160px -128px; }
.ui-icon-copy { background-position: -176px -128px; }
.ui-icon-contact { background-position: -192px -128px; }
.ui-icon-image { background-position: -208px -128px; }
.ui-icon-video { background-position: -224px -128px; }
.ui-icon-script { background-position: -240px -128px; }
.ui-icon-alert { background-position: 0 -144px; }
.ui-icon-info { background-position: -16px -144px; }
.ui-icon-notice { background-position: -32px -144px; }
.ui-icon-help { background-position: -48px -144px; }
.ui-icon-check { background-position: -64px -144px; }
.ui-icon-bullet { background-position: -80px -144px; }
.ui-icon-radio-on { background-position: -96px -144px; }
.ui-icon-radio-off { background-position: -112px -144px; }
.ui-icon-pin-w { background-position: -128px -144px; }
.ui-icon-pin-s { background-position: -144px -144px; }
.ui-icon-play { background-position: 0 -160px; }
.ui-icon-pause { background-position: -16px -160px; }
.ui-icon-seek-next { background-position: -32px -160px; }
.ui-icon-seek-prev { background-position: -48px -160px; }
.ui-icon-seek-end { background-position: -64px -160px; }
.ui-icon-seek-start { background-position: -80px -160px; }
/* ui-icon-seek-first is deprecated, use ui-icon-seek-start instead */
.ui-icon-seek-first { background-position: -80px -160px; }
.ui-icon-stop { background-position: -96px -160px; }
.ui-icon-eject { background-position: -112px -160px; }
.ui-icon-volume-off { background-position: -128px -160px; }
.ui-icon-volume-on { background-position: -144px -160px; }
.ui-icon-power { background-position: 0 -176px; }
.ui-icon-signal-diag { background-position: -16px -176px; }
.ui-icon-signal { background-position: -32px -176px; }
.ui-icon-battery-0 { background-position: -48px -176px; }
.ui-icon-battery-1 { background-position: -64px -176px; }
.ui-icon-battery-2 { background-position: -80px -176px; }
.ui-icon-battery-3 { background-position: -96px -176px; }
.ui-icon-circle-plus { background-position: 0 -192px; }
.ui-icon-circle-minus { background-position: -16px -192px; }
.ui-icon-circle-close { background-position: -32px -192px; }
.ui-icon-circle-triangle-e { background-position: -48px -192px; }
.ui-icon-circle-triangle-s { background-position: -64px -192px; }
.ui-icon-circle-triangle-w { background-position: -80px -192px; }
.ui-icon-circle-triangle-n { background-position: -96px -192px; }
.ui-icon-circle-arrow-e { background-position: -112px -192px; }
```

```
.ui-icon-circle-arrow-s { background-position: -128px -192px; }
.ui-icon-circle-arrow-w { background-position: -144px -192px; }
.ui-icon-circle-arrow-n { background-position: -160px -192px; }
.ui-icon-circle-zoomin { background-position: -176px -192px; }
.ui-icon-circle-zoomout { background-position: -192px -192px; }
.ui-icon-circle-check { background-position: -208px -192px; }
.ui-icon-circlesmall-plus { background-position: 0 -208px; }
.ui-icon-circlesmall-minus { background-position: -16px -208px; }
.ui-icon-circlesmall-close { background-position: -32px -208px; }
.ui-icon-squaresmall-plus { background-position: -48px -208px; }
.ui-icon-squaresmall-minus { background-position: -64px -208px; }
.ui-icon-squaresmall-close { background-position: -80px -208px; }
.ui-icon-grip-dotted-vertical { background-position: 0 -224px; }
.ui-icon-grip-dotted-horizontal { background-position: -16px -224px; }
.ui-icon-grip-solid-vertical { background-position: -32px -224px; }
.ui-icon-grip-solid-horizontal { background-position: -48px -224px; }
.ui-icon-gripsmall-diagonal-se { background-position: -64px -224px; }
.ui-icon-grip-diagonal-se { background-position: -80px -224px; }


/* Misc visuals
----------------------------------*/

/* Corner radius */
.ui-corner-all,
.ui-corner-top,
.ui-corner-left,
.ui-corner-tl {
        border-top-left-radius: 4px/*{cornerRadius}*/;
}
.ui-corner-all,
.ui-corner-top,
.ui-corner-right,
.ui-corner-tr {
        border-top-right-radius: 4px/*{cornerRadius}*/;
}
.ui-corner-all,
.ui-corner-bottom,
.ui-corner-left,
.ui-corner-bl {
        border-bottom-left-radius: 4px/*{cornerRadius}*/;
}
.ui-corner-all,
.ui-corner-bottom,
.ui-corner-right,
.ui-corner-br {
        border-bottom-right-radius: 4px/*{cornerRadius}*/;
}

/* Overlays */
.ui-widget-overlay {
        background: #aaaaaa/*{bgColorOverlay}*/ url(images/ui-
bg_flat_0_aaaaaa_40x100.png)/*{bgImgUrlOverlay}*/ 50%/*{bgOverlayXPos}*/ 50%/*{bgOverlayYPos}*/
repeat-x/*{bgOverlayRepeat}*/;
        opacity: .3/*{opacityOverlay}*/;
        filter: Alpha(Opacity=30)/*{opacityFilterOverlay}*/;
}
.ui-widget-shadow {
        margin: -8px/*{offsetTopShadow}*/ 0 0 -8px/*{offsetLeftShadow}*/;
        padding: 8px/*{thicknessShadow}*/;
        background: #aaaaaa/*{bgColorShadow}*/ url(images/ui-
bg_flat_0_aaaaaa_40x100.png)/*{bgImgUrlShadow}*/ 50%/*{bgShadowXPos}*/ 50%/*{bgShadowYPos}*/
repeat-x/*{bgShadowRepeat}*/;
        opacity: .3/*{opacityShadow}*/;
        filter: Alpha(Opacity=30)/*{opacityFilterShadow}*/;
        border-radius: 8px/*{cornerRadiusShadow}*/;
}
```

Site.css

```
body
{
    font-size: .85em;
    font-family: "Trebuchet MS", Verdana, Helvetica, Sans-Serif;
    color: #232323;
    background-color: #ffcc00;
}

header,
footer,
nav,
section {
    display: block;
}

#container {
   min-height:100%;
   position:relative;
}

#body  {overflow:auto;
         padding:10px;
         width:100%;
   padding-bottom:20px;}  /* must be same height as the footer */

#footer
{
    margin-left:auto;
    margin-right:auto;
    position:absolute;
    bottom:0;
    width: 99%;
    height:20px;
    color: DimGray !important;
    /*border-top:1px solid DimGray;
    border-bottom:1px solid DimGray;*/
}

.title
{
    /*display:block;*/
    position:absolute;
    text-align: center;
    left: 50%;
    margin-left:-100px;
    font-weight:bold;
    font-size:200%;
    }

 .header
 {
     position:relative;
     width:100%;
     }


/* Styles for basic forms
-----------------------------------------------------------*/



fieldset
{
    border:1px solid #ddd;
    padding:0 1.4em 1.4em 1.4em;
    margin:0 0 1.5em 0;
}

legend
```

```
{
    font-size:1.2em;
    font-weight: bold;
}

textarea
{
    min-height: 75px;
}

.editor-label
{
    margin: 1em 0 0 0;
}

.editor-field
{
    margin:0.5em 0 0 0;
}


/* Styles for validation helpers
-----------------------------------------------------------*/
.field-validation-error
{
    color: #ff0000;
}

.field-validation-valid
{
    display: none;
}

.input-validation-error
{
    border: 1px solid #ff0000;
    background-color: #ffeeee;
}

.validation-summary-errors
{
    font-weight: bold;
    color: #ff0000;
}

.validation-summary-valid
{
    display: none;
}
```

Controlery

LoginController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace TestMVCEmptyTemplate.Controllers
{
    public class LoginController : Controller
    {
        //
        // GET: /Login/

        public ActionResult Index()
```

```
        {
            return View();
        }
        [HttpPost]
        public ActionResult index(LoginModel model)
        {
            //  kod co neco dela
            if (ModelState.IsValid)
            {
                if (Membership.ValidateUser(model.Username,model.Password))
                    //(model.Username == "Jack" && model.Password == "getmein") //TO , pripojit na
AD
                { //here you have cookie
                    FormsAuthentication.SetAuthCookie(model.Username, false);
                    return RedirectToAction("index", "Share");
                }
                else
                {
                    ModelState.AddModelError("", "Invalid username or password");
                }
            }
            //konec kodu co neco dela
            return View();
        }
    }
}
```

Logout Controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace TestMVCEmptyTemplate.Controllers
{
    public class LogoutController : Controller
    {
        //
        // GET: /Logout/

        public ActionResult Logout()
        {
            FormsAuthentication.SignOut();
            return View();
        }

    }
}
```

Profile controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace TestMVCEmptyTemplate.Controllers
{
    public class ProfileController : Controller
    {
        //
        // GET: /Profile/

        public ActionResult Index()
```

```
        {
            return View();
        }

    }
}
```

ShareController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data;
using System.Data.SqlClient;
using System.DirectoryServices.ActiveDirectory;
using System.DirectoryServices;
using System.Text.RegularExpressions;
using System.Collections;



namespace TestMVCEmptyTemplate.Controllers
{
    public class ShareController : Controller
    {

        public ActionResult ShareTest1()
        {
            return View();
        }
        //
        // GET: /Share/
        //public ShareModel ShareModelInstance=new ShareModel();

        public List<string> RemovePriviledged(List<string> Groups) //removes priviledged groups,
this is first filtering
        {
            List<string> Patterns = new List<string> { "OU=DelegationGroups,OU=Servers", "Admin",
"OU=CitrixFarms" };
            foreach (string pattern in Patterns)
            {
                for (int i = 0; i < Groups.Count;i++ )
                {
                    if (Regex.IsMatch(Groups[i], pattern))
                    {
                        Groups.Remove(Groups[i]);
                        i--;
                    }
                }
            }
            return Groups;
        }

        public List<string> MnozinaRozdil(List<string> A, List<string> B) //A-B
        {
            //List<string> rozdil = new List<string>();
            foreach (string prvek in A)
            {
                if (!(B.Contains(prvek)))
                {
                    //rozdil.Add(prvek);
                    A.Remove(prvek);
                }
            }
            return A;
        }

        public List<string> MnozinaRozdilUpravaGrup(List<string> A, List<string> B, List<string>
RequestedShares,string potomek) //A-B //TODO you have to remove, not add!
```

```
        {
           //List<string> rozdil = new List<string>();
           string prvekA;
           for (int i=0; i< A.Count; i ++)
            {
               prvekA=A[i];
                foreach (string prvekB in B)
                {
                    if ((prvekB.Equals(prvekA) )) //TO DO
                    {
                        if ((Regex.IsMatch(potomek, "^CN=G.*")) && (Regex.IsMatch(prvekB,
"^CN=D.*"))) //jestlize je potomek G a predek DL tak predka vyradime, uprednostovany jsou globalni
skupiny
                        {
                            A.Remove(prvekB);
                            i--;
                        }
                        else
                        {

                            string query="Select g.Name, count (gts.id) as unrequested "+
                                         "from AccessManager.dbo.GroupToShare as gts "+
                                         "inner join AccessManager.dbo.[Group] as g on
g.id=gts.GroupID "+
                                         "where gts.ID not in "+
                                                 "(select gts2.id "+
                                              "from AccessManager.dbo.GroupToShare as gts2 "+
                                                 "inner join AccessManager.dbo.Share as s on
s.id=gts2.ShareID where ";
                            for (int j=0; j< RequestedShares.Count;j++)
                            {
                                if(j!=0)
                                {
                                    query+=" or ";
                                }
                                string cosi = Regex.Replace(RequestedShares[j], "\\\\", "\\\\");
                                query+=" s.path='"+cosi+"') ";
                            }
                            query += "and (g.DistinguishName='" + prvekA + "' or
g.DistinguishName='" + potomek + "') group by g.name";

                            System.Data.SqlClient.SqlConnection Conn = CreateConnection();
                            DataTable data = new DataTable();
                            System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
                            int rowsamount = adap.Fill(data);
                            //just 2 were returned (grop by)
                            if(rowsamount>1)
                            {
                              string pocetPredka;
                              string pocetPotomka;
                                if(data.Rows[0]["Name"]==prvekA){
                                    pocetPotomka=data.Rows[0]["unrequested"].ToString();
                                   pocetPredka=data.Rows[1]["unrequested"].ToString();
                                    }
                                else{
                                      pocetPredka=data.Rows[0]["unrequested"].ToString();
                                     pocetPotomka=data.Rows[1]["unrequested"].ToString();
                                     }
                              if (Convert.ToInt32(pocetPotomka) >
Convert.ToInt32(pocetPredka))//jestlize skupina predka poskytuje mene nevyzadanych sharu nez
potomek, vyradime potomka
                               {
                                   if (A.Remove(potomek))
                                   {
                                       i--;
                                   }
                              }
                            }
                        }
                    }
```

```
                }
            }
            return A;
        }

        public List<string> MnozinaPrunik(List<string> A, List<string> B)
        {
            List<string> prunik= new List<string>();
            foreach (string prvek in A)
            {
                if (B.Contains(prvek))
                {
                    prunik.Add(prvek);
                }
            }
            return prunik;
        }

        public List<string> MnozinaSjednoceni(List<string> A, List<string> B)
        {
            List<string> sjednoceni = A.ToList();
            foreach (string prvek in B)
            {
                if (!(sjednoceni.Contains(prvek)))
                {
                    sjednoceni.Add(prvek);
                }
            }
            return sjednoceni;
        }

        public List<string> GetUserMembership(string user)
        {
            List<string> TheList = new List<string>();
            //System.Collections.Queue TheQueue = new Queue();

            DirectoryEntry directory = new DirectoryEntry("LDAP://DC=prg-dc,DC=dhl,DC=com");
            string filter = "(&(cn=" + user +
"*)(!objectClass=computer)(!objectClass=nTFRSMember))";
            string[] strCats = { "memberOf" };
            Regex Regexp= new Regex(",OU=.*");

            DirectorySearcher dirUser = new DirectorySearcher(directory, filter, strCats,
SearchScope.Subtree);
            SearchResultCollection results = dirUser.FindAll();
            foreach (SearchResult result in results)
            {
                foreach (string propkey in result.Properties.PropertyNames)
                {
                    if (propkey == "memberOf")
                    {
                        ResultPropertyValueCollection valueCollection = result.Properties[propkey];
                        foreach (Object propvalue in valueCollection)
                        {
                            //TheQueue.Enqueue(propvalue.ToString());
                            TheList.Add(propvalue.ToString());
                        }
                    }
                }
            }

            return TheList;
        }

        public ActionResult GetUID(string term)
        {
            string uidstart = term;
            DirectoryEntry directory = new DirectoryEntry("LDAP://DC=prg-dc,DC=dhl,DC=com");
            string filter = "(&(cn=" +uidstart+
"*)(!objectClass=computer)(!objectClass=nTFRSMember))";
            string[] strCats = { "cn" };
```

```
            List<string> Found = new List<string>();
            DirectorySearcher dirUser = new DirectorySearcher(directory, filter, strCats,
SearchScope.Subtree);
            SearchResultCollection results = dirUser.FindAll();
            foreach (SearchResult result in results)
            {
                foreach (string propkey in result.Properties.PropertyNames)
                {
                    if (propkey == "cn")
                    {
                        ResultPropertyValueCollection valueCollection = result.Properties[propkey];
                        foreach (Object propvalue in valueCollection)
                        {
                            Found.Add(propvalue.ToString());
                        }
                    }
                }
            }
            return Json(Found.ToArray(), JsonRequestBehavior.AllowGet);
        }

        public bool ValidateUser(string UID)
        {
            List<string> TheList = new List<string>();
            DirectoryEntry directory = new DirectoryEntry("LDAP://DC=prg-dc,DC=dhl,DC=com");
            string filter = "(&(cn=" + UID +
"*)(!objectClass=computer)(!objectClass=nTFRSMember))";
            string[] strCats = {"distinguishedname"};

            DirectorySearcher dirUser = new DirectorySearcher(directory, filter, strCats,
SearchScope.Subtree);
            SearchResultCollection results = dirUser.FindAll();
            foreach (SearchResult result in results)
            {
                foreach (string propkey in result.Properties.PropertyNames)
                {
                    if (propkey == "distinguishedname")
                    {
                        ResultPropertyValueCollection valueCollection = result.Properties[propkey];
                        foreach (Object propvalue in valueCollection)
                        {
                            //TheQueue.Enqueue(propvalue.ToString());
                            TheList.Add(propvalue.ToString());
                        }
                    }
                }
            }

            if (TheList.Count > 0 && (!string.IsNullOrEmpty(TheList[0])))
            { return true; }
            else
            { return false; }

        }

        public bool ValidatePath(string path)
        {
            SqlConnection Conn = CreateConnection();
            List<string> returnstring = new List<string>();
            DataTable data = new DataTable();
            string query = "SELECT Path FROM dbo.Share WHERE Path ='" + path + "'";
            System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
            int rowsamount = adap.Fill(data);

            if (rowsamount > 0)
            { return true; }
            else
            { return false; }
        }
```

```
        public ActionResult Rozbocovac(ShareModel model)
        {
            /*Checking imput parrameters
            */
            bool error = false;
            bool ValidUser = false;
            bool ValidPath = false;

            if (model.errormsg.Count > 0)
            {
                model.errormsg[0] = "";
            }
            for (int i = 0; (i < model.usersstandard) && (model.usersstandard ==
model.amountofusers) &&(!error); i++)
            {
                if ((model.Rows[i].typeofrights == 3) &&
(string.IsNullOrEmpty(model.Rows[i].template)))
                {
                    error = true;
                    model.errormsg.Add("Here is and empty template yet you ask for rights according
template?");
                }

if((model.Rows[i].typeofrights==0)||(string.IsNullOrEmpty(model.Rows[i].forWhom))||(string.IsNullOr
Empty(model.Rows[i].forWhom)))
                {
                    error=true;
                    model.errormsg.Add("Name of user, path and type of rights are mandatory
attribues.");
                }
                if (model.Rows[i].typeofrights == 3)
                {
                    error = error && ValidateUser(model.Rows[i].template);
                }
                ValidUser=ValidateUser(model.Rows[i].forWhom);
                if (!ValidUser)
                {
                    model.errormsg.Add("Invalid user provided");
                }
                ValidPath=ValidatePath(model.Rows[i].path);
                if (!ValidPath)
                {
                    model.errormsg.Add("Invalid path provided\n");
                }
                error = error || !ValidPath || !ValidUser;
            }



            if (model.amountofusers > 20)
            {
                model.amountofusers = 20;
                model.errormsg.Add("Sorry, no more than 20 users");
            }



            if (model.usersstandard != model.amountofusers || error)
            {
                model.usersstandard = model.amountofusers;
                TempData["modelRozb"] = model;
                return RedirectToAction("Index");
            }
            else
            {
                TempData["model"] = model;
                return RedirectToAction("HandleRequest");
            }

        }
```

```
       /* public ActionResult Test()
        {

        }*/

        /*public DataTable GroupsOnShare(string path,int typeofrights)
        {
              if (typeofrights==1) //read
              {
                  string query = "SELECT g.Name, COUNT(g.Name) AS Count " +
                                 "FROM [dbo].[GroupToShare] AS gts" +
                                 "INNER JOIN [dbo].[Group] AS g ON gts.GroupID = g.ID" +
                                 "INNER JOIN [dbo].[Share] AS s ON gts.ShareID = s.ID" +
                                 "WHERE g.Name in ( SELECT g2.Name FROM [dbo].[GroupToShare] AS
gts2" +
                                                    "INNER JOIN [dbo].[Group] AS g2 ON
gts2.GroupID = g2.ID" +
                                                    "INNER JOIN [dbo].[Share] AS s2 ON
gts2.ShareID = s2.ID" +
                                                    "WHERE s2.Path='" + path + "'" +
                                                    "GROUP BY g2.Name) and
gts.RightsType='Modify'" +
                                                    "GROUP BY g.Name" +
                                                    "ORDER BY COUNT(ShareID) DESC, g.Name";
              }
              else if(typeofrights==2) //modify
              {
                string query = "SELECT g.Name, COUNT(g.Name) AS Count "+
                               "FROM [dbo].[GroupToShare] AS gts" +
                               "INNER JOIN [dbo].[Group] AS g ON gts.GroupID = g.ID" +
                               "INNER JOIN [dbo].[Share] AS s ON gts.ShareID = s.ID" +
                               "WHERE g.Name in ( SELECT g2.Name FROM [dbo].[GroupToShare] AS
gts2" +
                                                 "INNER JOIN [dbo].[Group] AS g2 ON gts2.GroupID =
g2.ID" +
                                                 "INNER JOIN [dbo].[Share] AS s2 ON gts2.ShareID =
s2.ID" +
                                                 "WHERE s2.Path='"+path+"'"+
                                                 "GROUP BY g2.Name) and gts.RightsType='Modify'" +
                                                 "GROUP BY g.Name" +
                                                 "ORDER BY COUNT(ShareID) DESC, g.Name";
                }
              else if (typeofrights == 3) //template
              {
                  string query = "SELECT g.Name, COUNT(g.Name) AS Count " +
                               "FROM [dbo].[GroupToShare] AS gts" +
                               "INNER JOIN [dbo].[Group] AS g ON gts.GroupID = g.ID" +
                               "INNER JOIN [dbo].[Share] AS s ON gts.ShareID = s.ID" +
                               "WHERE g.Name in ( SELECT g2.Name FROM [dbo].[GroupToShare] AS
gts2" +
                                                 "INNER JOIN [dbo].[Group] AS g2 ON gts2.GroupID =
g2.ID" +
                                                 "INNER JOIN [dbo].[Share] AS s2 ON gts2.ShareID =
s2.ID" +
                                                 "WHERE s2.Path='" + path + "'" +
                                                 "GROUP BY g2.Name)" +
                                                 "GROUP BY g.Name" +
                                                 "ORDER BY COUNT(ShareID) DESC, g.Name";
              }
              System.Data.SqlClient.SqlConnection Conn = CreateConnection();
              DataTable data = new DataTable();
              System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
              int rowsamount = adap.Fill(data);
              return data;

        }*/

        public List<String> GroupsOnShare(string path, int typeofrights)
        {
```

```
            List<string> returnstring = new List<string>();
            string query;
            string rights;
            if(typeofrights==1)
            {
                rights = "Read";
                query="select g.DistinguishName " +
                "from dbo.[Group] g, dbo.Share s, dbo.GroupToShare f "+
                "where s.ID = f.ShareID and g.ID = f.GroupID and f.RightsType='"+rights+"' and
s.Path='"+path+"'";
            }
            else if (typeofrights == 2)
            {
                rights = "Modify";
                query="select g.DistinguishName " +
                "from dbo.[Group] g, dbo.Share s, dbo.GroupToShare f "+
                "where s.ID = f.ShareID and g.ID = f.GroupID and f.RightsType='"+rights+"' and
s.Path='"+path+"'";
            }
            else
             {
                query="select g.DistinguishName " +
                "from dbo.[Group] g, dbo.Share s, dbo.GroupToShare f " +
                "where s.ID = f.ShareID and g.ID = f.GroupID and s.Path='"+path+"'";
            }

            System.Data.SqlClient.SqlConnection Conn = CreateConnection();
            DataTable data = new DataTable();
            System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
            int rowsamount = adap.Fill(data);

            //formating to string
            for (int i = 0; i < rowsamount; i++)
            {
                returnstring.Add(data.Rows[i]["DistinguishName"].ToString());
            }

            return returnstring;


        }

        public ActionResult Index(ShareModel model)
        {
            //ShareModel vm = new ShareModel();
           // SqlConnection Conn = CreateConnection();
            //model.amountofusers = 1;
          //  model.Shares = GetShares(Conn);
            ShareModel mod = (ShareModel)TempData["modelRozb"];

            if (mod !=null)
            {
                model=mod;
            }

            /*if (model.Rows == null)
            {
                ShareRow radek = new ShareRow(); //initilization for emty model
                model.Rows.Add(radek);
            }*/
            for (int i = model.Rows.Count; i < model.amountofusers; i++)
            {
                ShareRow radek = new ShareRow(); //just filing blank spaces so view is ok
                model.Rows.Add(radek);
            }
            return View(model);
        }

        public ActionResult HandleRequest()
        {
```

```
            ShareModel model=(ShareModel)TempData["model"]; //everything will be stored in model
and modelwill be passsed via tempdata
            Hashtable Userindex = new Hashtable(); //Key=names of users, value=int, index in
requestlist

            //List<string> GroupListTemplate =new List <string> ();
            //List<string> GroupListUser = new List<string>();
            List<string> GroupsToAdd = new List<string>();
            int index;
            DataTable GroupsTable=new DataTable();
            for (int i = 0; i < model.amountofusers; i++ ) //for each row in request (and no bigger
than showed rows)
            {
                //GroupListUser = GetUserMembership(model.Rows[i].forWhom);
                //is it new request or just old one?
                if (Userindex.ContainsKey(model.Rows[i].forWhom))
                {
                    index = (int)Userindex[model.Rows[i].forWhom]; //don't forget to pud as value
ints!
                }
                else
                {
                    Request req = new Request();
                    model.Requests.Add(req);
                    index = model.Requests.Count - 1;
                }

                //now lets decide the case, according the template or not?
                if (model.Rows[i].typeofrights == 3) //so template it is
                {
                    //GroupListTemplate = GetUserMembership(model.Rows[i].template);
                    GroupsToAdd =
MnozinaPrunik(MnozinaRozdil(GetUserMembership(model.Rows[i].template),
GetUserMembership(model.Rows[i].forWhom)), GroupsOnShare(model.Rows[i].path,
model.Rows[i].typeofrights)); //predpokladejme z to jsou vsechno DN
                }
                else if (model.Rows[i].typeofrights == 2 || model.Rows[i].typeofrights == 1)
//modify or read
                {
                    GroupsToAdd = GroupsOnShare(model.Rows[i].path, model.Rows[i].typeofrights);
                }
                model.Requests[index].User = model.Rows[i].forWhom;
               /* if ((model.Requests[index].Shares).is == null) //ostrit NULL //TODO
                {
                    model.Requests[index].Shares = new List<string> { model.Rows[i].path };
                }
                else
                {*/
                    model.Requests[index].Shares.Add(model.Rows[i].path);
               // }

model.Requests[index].RequestedGroups=MnozinaSjednoceni(model.Requests[index].RequestedGroups,
GroupsToAdd);
                /*foreach (string group in GroupsToAdd)
                {
                    model.Requests[index].RequestedGroups.Add(group);
                }*/
                //so that's it, now we proccessed all request rows, what is next?
            }

            //Filtering algorythms (template, descendant, shareamount)
            //Template filtering was incorporated in adding from template, so it is done
            //Filteing priviledged groups
            foreach (Request Req in model.Requests)
            {
                Req.RequestedGroups = RemovePriviledged(Req.RequestedGroups);
            }
            //Descendantfiltering, it is not needed to have more group and it's descendant as
descendant has at least same rights.
            for (int j=0; j < model.Requests.Count ; j++ )
            {
```

```
                for (int i = 0; i < model.Requests[j].RequestedGroups.Count; i++) //to find members
of all that groups in single reques
                {
                    //model.Requests[j];
                    List<string> ParsingList = new List<string>();
                    DirectoryEntry directory = new DirectoryEntry("LDAP://DC=prg-
dc,DC=dhl,DC=com"); //TO DO, it doesn't work
                    string[] strCats = { "memberOf" };
                    string GroupItselfDN = model.Requests[j].RequestedGroups[i];
                    string filtr = Regex.Replace(GroupItselfDN,
",OU=.*","");//"(&(memberOf=cn="+GroupItselfDN+",ou=ApplicationGroups,ou=DelegationGroups,ou=Server
s,dc=prg-dc,dc=dhl,dc=com)(objectCategory=group))"; //searching fo PARENTS
                    DirectorySearcher dirUser = new DirectorySearcher(directory, filtr, strCats,
SearchScope.Subtree);
                    SearchResultCollection results = dirUser.FindAll();

                    foreach (SearchResult result in results) //so here we fined all members of that
group
                    {
                        foreach (string propkey in
result.Properties.PropertyNames)//result.Properties.PropertyNames)
                        {
                            if (propkey == "memberof")
                            {
                                ResultPropertyValueCollection valueCollection =
result.Properties[propkey];
                                foreach (Object propvalue in valueCollection)
                                {
                                    ParsingList.Add(propvalue.ToString());
                                }
                            }
                        }
                    }
                    //we got the members, so now we need to remove them from the search list
                    List<string> NewList =
MnozinaRozdilUpravaGrup(model.Requests[j].RequestedGroups, ParsingList, model.Requests[j].Shares,
GroupItselfDN); //removed members, here be problem
                    int deletedgroups= model.Requests[j].RequestedGroups.Count- NewList.Count;
                    //parsed lis replaced the previous one in request recorded
                    //if ome groups are replaced, the counter i must decrease (but not bellow zero)
as t is quite difficult to figure out order of groups, it will decrease by difference
                    model.Requests[j].RequestedGroups = NewList;
                    j = j - deletedgroups;
                    if (j < 0)
                    {
                        j = 0; //not bellow 0, that would be index problem!
                    }

                }


            }//ok so descendat filtering is done
            //LAst filering, shares! after that, just priorizing groups and chosing right ones

            for (int i = 0; i < model.Requests.Count; i++) //for each request
            {
                if (model.Requests[i].RequestedGroups.Count > 0) //at least some groups were found
                {
                    DataTable Tabulka1 = SharesFiltering(model.Requests[i].Shares,
model.Requests[i].RequestedGroups); //we just have all the neccessary groups
                    // DataTable Tabulka2 = Tabulka1.Clone();
                    List<string> FinalLisToAdd = new List<string>();
                    do
                    {
                        int radky = Tabulka1.Rows.Count;
                        List<string> FinalListToRemove = new List<string>(); //list of shares to be
removed
                        string TheName = Tabulka1.Rows[0]["DistinguishName"].ToString(); //it is
first group becaus these are sorted from the best!
                        FinalLisToAdd.Add(TheName); //adding name of group
```

```
                        //gettng shares where this group has access
                        string selectexpression = "DistinguishName ='" + TheName + "'";
                        DataRow[] FoundRows = Tabulka1.Select(selectexpression); //gettng shares
where this group has access to
                        foreach (DataRow Row in FoundRows)
                        {
                            FinalListToRemove.Add(Row["path"].ToString()); //we got set of sares in
list
                        }
                        for (int j = 0; j < Tabulka1.Rows.Count; j++)
                        {
                            if (FinalListToRemove.Contains(Tabulka1.Rows[j]["path"].ToString()))
                            {
                                Tabulka1.Rows[j].Delete(); //removing that
                            }
                        }
                        Tabulka1.AcceptChanges();
                    } while (Tabulka1.Rows.Count > 0);
                    List <string> FinalListtoadd2=new List<string>();
                    foreach(string grp in FinalLisToAdd)
                    {
                       string hu = grp;
                       hu=Regex.Replace(hu,"^CN=","");
                       hu=Regex.Replace(hu,",OU.*=","");
                       FinalListtoadd2.Add(hu);
                    }
                    model.Requests[i].RequestedGroups = FinalListtoadd2;
                }
                else
                {
                    //int index=model.Missing.Count();
                    Unavilable cosi = new Unavilable();
                    cosi.User = model.Requests[i].User;
                    cosi.Shares = model.Requests[i].Shares;
                    model.Missing.Add(cosi);
                }


            }


                //passing data back
                TempData["model"] = model;
            return RedirectToAction("Evaluate");

        }

        public DataTable SharesFiltering(List<string> Shares, List<string> Groups)
        {
            //building that query
            string query = "select s2.path, g2.DistinguishName, x.pocet " +
                           "FROM [dbo].[GroupToShare] AS gts2 " +
                           "join [dbo].[Share] as s2 on gts2.ShareID = s2.ID " +
                           "join [dbo].[Group] AS g2 ON gts2.GroupID = g2.ID " +
                           "join ( " +
                               "select COUNT(gts.id) as pocet, g.DistinguishName " +
                               "from [dbo].[GroupToShare] AS gts " +
                               "join [dbo].[Group] AS g ON gts.GroupID = g.ID ";

                               if(Groups.Count>0)
                               {
                               query+="where g.DistinguishName in (";

                               for(int k=0; k<Groups.Count;k++)
                               {
                                   if(k!=0)
                                   {
                                       query+=",";
                                   }
                                   query+="'"+Groups[k]+"'";
```

```
                                }
                                query += ")";
                                }


                              query+="group by g.DistinguishName) as x on
g2.DistinguishName=x.DistinguishName " +
                            "where s2.Path in (";
                            for(int k=0; k<Shares.Count;k++)
                            {
                                if(k!=0)
                                {
                                  query+=",";
                                }
                                //string ShareKPridani=Regex.Replace(Shares[k],"\\\\","\\\\");
//hadning the \ in path
                                query += "'" + Shares[k] + "'";
                            }
                          query+=") "+
                            "order by  pocet DESC";
            SqlConnection Conn = CreateConnection();
            DataTable data = new DataTable();
            System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
            int rowsamount = adap.Fill(data);
            return data;
        }

        public ActionResult Evaluate(ShareModel model)
        {
            ShareModel mod = (ShareModel)TempData["model"];

            if (mod != null)
            {
                model = mod;
            }


            //model.zobraz = GetUserMembership(model.Rows[0].template);
            return View(model); //redirecting to report page
        }

        public ActionResult GetShares(string term)
        {
            //quering
            string pattern="\\\\\\\\\";
            string replace ="\\\\";
            Regex reg = new Regex(pattern);
            string result = reg.Replace(term, replace);
            Regex reg2 = new Regex("\\w+\\\\");
            result = reg2.Replace(result, "\\");

            SqlConnection Conn=CreateConnection();
            List <string> returnstring=new List<string>();
            DataTable data = new DataTable();
            string query="SELECT Path FROM dbo.Share WHERE Path LIKE '"+ result +"%'";
            System.Data.SqlClient.SqlDataAdapter adap = new
System.Data.SqlClient.SqlDataAdapter(query, Conn);
            int rowsamount = adap.Fill(data);

            //formating to string
            for (int i = 0; i < rowsamount; i++)
            {
                returnstring.Add(data.Rows[i]["Path"].ToString());
            }

            return Json(returnstring.ToArray(), JsonRequestBehavior.AllowGet);
            //return data;
        }

        System.Data.SqlClient.SqlConnection CreateConnection()
```

```
        {
            System.Data.SqlClient.SqlConnection connection = new
System.Data.SqlClient.SqlConnection("Server=czchowsint410,1525;Database=AccessManager;integrated
security=SSPI");
            return connection;
        }

    }
}
```

Modely

LoginModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.ComponentModel.DataAnnotations;

namespace TestMVCEmptyTemplate.Controllers
{
    public class LoginModel
    {
        [Required]
        public string Username { get; set; }
        [Required]
        public string Password { get; set; }
    }
}
```

ShareModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.ComponentModel.DataAnnotations;

namespace TestMVCEmptyTemplate.Controllers
{
     public class ShareModel
    {
       public ShareModel()
       {
         // this.conn = new
System.Data.SqlClient.SqlConnection("Server=czchowsint410,1525;Database=AccessManager;integrated
security=SSPI");
          this.amountofusers = 1;
          this.usersstandard = 1;
          this.Rows=new List<ShareRow>();
          this.Requests = new List<Request>();
          this.Missing = new List<Unavilable>();
          this.errormsg = new List<string>();
            //this.Requests.Shares=new List<string>();
            //this.Requests.RequestedGroups=new List<string>();
       }

       //public System.Data.SqlClient.SqlConnection conn { get; set; }
       public int usersstandard {get; set; }
       public int amountofusers { get; set; }


       public List<string> errormsg { get; set; }
       public List<ShareRow> Rows { get; set; } //list of requested shares
       public List<string> zobraz { get; set; }
         [Required]
       public List<Request> Requests { get; set; }
       public List<Unavilable> Missing { get; set; }
    }
```

```
    public class Request //contains request after being proccessed
    {
        public Request()
        {
            this.RequestedGroups = new List<string>();
            this.Shares = new List<string>();

        }
        [Required]
        public string User { get; set; }
        [Required]
        public List<string> RequestedGroups { get; set; }
        [Required]
        public List<string> Shares { get; set; }

    }

    public class Unavilable
    {
        public Unavilable()
        {
            this.Shares = new List<string>();
        }
        public string User { get; set; }
        public List <string> Shares { get; set; }
    }

    public class ShareRow
    {
        public string forWhom { get; set; }
        public string path { get; set; }
        public int typeofrights { get; set; }
        public string template { get; set; }
    }
}
```

Views

Share\Evaluate.cshtml

```
@model TestMVCEmptyTemplate.Controllers.ShareModel

@{
    ViewBag.Title = "Evaluation";
}

<h2>Evaluation</h2>

@foreach (var Request in Model.Requests)
{
    if (Request.RequestedGroups.Count > 0)
    {
    <text>User "</text>@Request.User<text>" should be added to following groups</text><br/>
    foreach (var group in Request.RequestedGroups)
    {
        @group <br />
    }
     <br />
    }
}

@if (Model.Missing.Count > 0)
{
   <text>Rights for follwing requests were not found and will require deeper investigation</text>
}
@foreach (var Mis in Model.Missing)
{

    <text>User "</text> @Mis.User <text>" to following shares:</text><br />
    foreach (var share in Mis.Shares)
```

```
    {
        @share <br />
    }
    <br />
}
```

Share\Index.cshtml

```
@model TestMVCEmptyTemplate.Controllers.ShareModel

@{
    ViewBag.Title = "Share list";
}



<!--<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.1/themes/base/jquery-ui.css" /> -->
<link rel="stylesheet" href="@Url.Content("~/Content/Query.css")" />
<!--<script src="http://code.jquery.com/jquery-1.9.1.js"></script>-->
<!--<script src="http://code.jquery.com/ui/1.10.1/jquery-ui.js"></script>-->
<script src="@Url.Content("~/Scripts/jquery-1.5.1.js")"type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery-ui-1.8.11.js")"type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/AutoUpdateScript.js")" type="text/javascript"></script>
<!--script src="Url.Content("~/Scripts/HideAndSeek.js")" type="text/javascript"></script-->


<h2>Welcome, please select shares for the user(s)</h2>
@{ foreach(var error in Model.errormsg)
   {
        <br />
        <b>@error </b>
   }
 }
@using (Html.BeginForm("Rozbocovac", "Share", FormMethod.Post))
{
    <text>Currently displaying </text>
    @Html.TextBoxFor(m => m.amountofusers, Model.amountofusers)
    @Html.HiddenFor(model => model.usersstandard)

    <input type="submit" name="Change" value="GO!" />
    <br />

   <table>
   <tr>
    <td>For User</td>
    <td>Path to share</td>
    <td>Type of rights</td>
    <td>Templateuser?</td>
   </tr>


    @for (var i = 0; i < Model.amountofusers; i++)
    {

        <tr>
         <td>@Html.TextBoxFor(m => m.Rows[i].forWhom, new { data_autocomplete_url =
Url.Action("GetUID"), placeholder = "defultname", min_length = "3" })</td>
         <td>@Html.TextBoxFor(m => m.Rows[i].path, new { data_autocomplete_url =
Url.Action("GetShares"), placeholder = "defaultpath" })</td>
         <td>
         @{var pole = "Rows[" + i + "].template";}
          @Html.RadioButtonFor(m => m.Rows[i].typeofrights, "1", new { HideHim = @pole,
data_HideHim = @pole, style="width:350"}) Read
          @Html.RadioButtonFor(m => m.Rows[i].typeofrights, "2", new { HideHim = @pole,
data_HideHim = @pole, style="width:350"}) Modify
          @Html.RadioButtonFor(m => m.Rows[i].typeofrights, "3", new { ShowHim = @pole,
data_ShowHim = @pole, style="width:350"}) According template
          </td>
          <td>
              <div id='template' class='kulove'>
```

```
                @Html.TextBoxFor(m => m.Rows[i].template, new { data_autocomplete_url =
Url.Action("GetUID"), placeholder = "defultTemplate", Hideme = @pole })
                </div>
            </td>
           </tr>
    }
   </table>


    <br/>
  <!--<input type="submit" name="Rozbocovac" />-->
    @Html.ValidationSummary()
}
```

Login\index.cshtml

```
@model TestMVCEmptyTemplate.Controllers.LoginModel

@{
    ViewBag.Title = "index";
}

<h2>Login</h2>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>

@using (Html.BeginForm())
{
    <table cellspacing="10">
    <tr> <td>@Html.LabelFor(m => m.Username)</td> <td>@Html.TextBoxFor(m => m.Username, new { style
= "width:150px" }) </td></tr>
    <tr><td>@Html.LabelFor(m => m.Password)</td><td>@Html.PasswordFor(m => m.Password, new { style
= "width:150px" })</td></tr>
    <tr><td colspan="2" align="center"><input type="submit" value='Log me in' /> </td></tr>
    </table>

    @Html.ValidationSummary()

}
```

Logout\Logout.cshtml

```
@{
    ViewBag.Title = "Logout";
}

<h2>Bye</h2>
```

Shared\Layout.cshtml

```
<!DOCTYPE html>
 <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
     @RenderSection("Content", false)
    <script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")" type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/modernizr-1.7.min.js")" type="text/javascript"></script>

<div id="header">
    <!--<meta charset="utf-8" /> -->

   <span style="float:left; margin-left:auto;"><img
src="@Url.Content("~/Content/dhl_gbs_logo_white.gif")" alt="DHL Logo" align="left" /> </span>
    <span class="title" title="Share access manager" > Share access manager</span>
    @if (User.Identity.IsAuthenticated)
    {
     <span style="float:right;margin-right:auto;">@Html.ActionLink("Logout", "Logout",
"Logout")</span>
```

```
    }
</div>

<div id='body'>
    @RenderBody()
    @RenderSection("body", false)

</div>



<div id='footer'>
by Zdeněk Špinka for DHL  &#169 2013
</div>
```