PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

DIPLOMOVÁ PRÁCE

Webové kontra klasické desktopové aplikace

E-mailový klient

2011 Martin Bělaška

Anotace

Diplomová práce představuje moderní webové aplikace a provádí jejich srovnání s desktopovými aplikacemi. Vyjmenovává požadavky, které jsou na webové aplikace kladeny, seznamuje s jejich vnitřní architekturou a specifiky zabezpečení. Jsou popsány charakteristické přístupy pro práci s dočasnou pamětí a uchování stavových dat a problémy, které přináší. Dále se práce zabývá vzhledem a ovládáním uživatelského rozhraní webových aplikací a diskutuje používané technologie. Pro účely práce byl implementován e-mailový klient v jazyce PHP, jenž slouží jako modelová webová aplikace pro srovnání s jeho desktopovou alternativou.

Děkuji všem, kteří se svými názory a připomínkami podíleli na výsledné podobě této diplomové práce.

Obsah

Seznam obrázků

Seznam tabulek

1. Zadání diplomové práce

Cílem práce (na vlastní téma) je studie srovnávající moderní webové a (klasické) desktopové aplikace dle různých kritérií jako např. funkčnost, uživatelský komfort, podporované vlastnosti (features), výkon, bezpečnost, náročnost vývoje apod. Srovnání bude konkrétně demonstrováno na vybrané modelové aplikaci z oblasti, kde webové aplikace (tradičně) pokulhávají, např. emailový klient. Aplikace bude implementována jako webová aplikace schopná konkurovat desktopovým ekvivalentům, v kritériích výše, s důrazem na kvalitu UI a zabezpečení aplikace. Součástí práce bude také (všeobecný) rozbor možností dnešních webových aplikací s ukázkami využití, zmapování vývoje a diskuze webových standardů.

Dílčí požadavky

- Studie srovnávající moderní webové a (klasické) desktopové aplikace dle různých kritérií.
- Konkrétní srovnání na vybrané modelové aplikaci, implementace modelové aplikace schopná konkurovat desktopovým ekvivalentům (kvalita UI, zabezpečení, pokročilé a nestandardní funkce).
- Rozbor možností dnešních webových aplikací s ukázkami využití, zmapování vývoje, diskuze webových standardů.
- Zpracování webové aplikace dle webových standardů.

2. Úvod

Webové aplikace přestavují stále sílící fenomén dnešní doby a každý běžný uživatel sítě Internet se s nimi denně setkává. Tato úvodní kapitola přesně definuje termín webová aplikace, tak jak jej bude práce dále používat. Také krátce představí historii a uvede důvody jejich srovnání s aplikacemi desktopovými.

2.1. Historie

Historie webových aplikací je logicky pevně spjata s historií klasických webových stránek. V době, kdy první webové stránky vznikaly, byl Internet chápán pouze jako médium pro uchování a šíření informací. I přesto bylo již poměrně brzy možné pozorovat u těchto webových stránek vykonávání kódu na straně webového serveru. Obecně se jednalo o kód implementující jednoduché nástroje, které sloužily například k vyhledávání textu.

Nabízená funkcionalita se začala velice rychle rozšiřovat. Jako příklad můžeme uvést, že v roce 1990 vznikl historicky první plnohodnotný Internetový prohlížeč, jehož autorem byl fyzik Tim Berners-Lee. Tento prohlížeč nakonec pojmenoval WorldWideWeb, čímž položil obecný název pro přístup k informacím pomocí Internetu – „web“. Pracoval v textovém režimu a byl ovládán výhradně klávesnicí. Web v této době představoval jednu počítačovou síť a první stránky zobrazované prohlížečem WorldWideWeb byly určené k představení tohoto systému kolegům. Pouze o pět let později, kdy má síť Internet již 16 miliónů uživatelů a běžně se používají grafické prohlížeče, vzniká programovací jazyk PHP. Ten je určen především k nasazení v oblasti webových stránek, je interpretovaný a vykonává se na straně serveru. Dnes bychom tedy řekli, že je to jazyk určený pro vývoj webových aplikací. V této době se ovšem ještě nehovoří o webových aplikacích nýbrž o dynamických nebo interaktivních webových stránkách.

S tím, jak rostl počet uživatelů sítě Internet, tak stoupala i funkcionalita webových stránek. Tomuto rychlému nárůstu pomohla patrně i internetová horečka, jejíž počátek je možné datovat přibližně do roku 1996. Termín internetová horečka nebo internetová bublina je označení pro období hromadného rozkvětu internetových firem, které neměly promyšlený obchodní model a brzy zkrachovaly, avšak dokázaly přilákat mohutné investice.[8] Vedlejším efektem této horečky byl vznik velké spousty internetových služeb. Ty se často sdružovaly do tzv. portálů.

S nárůstem funkcionality webových stránek začal být Internet chápán také jako médium pro provoz aplikací různého charakteru. Přímým důsledkem byl v roce 1999 vznik nové disciplíny – webového inženýrství, která se zabývá specifickými vlastnostmi vývoje aplikací určených pro síť Internet. V jejím rámci jsou tyto aplikace nazývány jako webové aplikace. Definicí tohoto termínu se zabývá následující kapitola.

2.2. Webové aplikace

Ačkoliv je webová aplikace termín, který je běžně používaný, jeho přesná definice chybí. Aby tento termín odpovídal tomu, jak je dnes přirozeně chápán a jak jej bude tato práce dále používat, tak zavedeme následující definici:

Webová aplikace je webová stránka obsahující vykonatelný kód, která neslouží primárně k prezentaci statických textových dat.

Co tedy tato definice říká? Webová aplikace je představena jako rozšíření webové stránky, které obsahuje navíc vykonatelný kód. Tento kód přestavuje přidanou funkcionalitu, přičemž nezáleží, zda je vykonáván na straně serveru nebo na straně klienta v prohlížeči. Pouze pod touto optikou bychom ovšem mohli webovou stránku obsahující jednoduché vyhledávání označit za webovou aplikaci, což je v rozporu s přirozeným chápáním tohoto termínu. Proto je nutné dodat, že webová aplikace neslouží primárně k prezentaci statických textových dat.

Zástupců webových aplikací je celá řada, jako jejich typický příklad můžeme uvést internetové obchody a bankovnictví, fulltextové vyhledávací služby nebo webové e-mailové klienty.

2.3. Desktopové aplikace

Termín desktop má ve výpočetní technice dva zcela odlišné významy. Jeden je používán v oblasti hardware, druhý v oblasti software a nelze je vzájemně zaměňovat. V rámci těchto dvou oblastí tedy desktop představuje:

1. Stolní počítač, někdy také označovaný desktopový počítač. Termín je možné v tomto kontextu volně přeložit jako na „na stole“. Přenosnou variantou stolního počítače je laptop („na klíně“) a kapesní variantou palmtop („na dlani“).

2. Anglické označení pracovního prostředí grafických operačních systémů, pro které se v českém jazyce používá termín plocha.

Desktopová aplikace samozřejmě vychází z druhého uvedeného významu termínu desktop a přirozeně je chápeme jako program, který pracuje v okně na ploše operačního systému. Aby bylo zřejmé, co bude tento termín přesně označovat v následujícím textu, tak jej zadefinujeme podobně jako webové aplikace.

Desktopová aplikace je aplikační software s grafickým uživatelským rozhraním, který je lokální instalován na osobní počítač.

2.4. Desktopové vs. webové aplikace

Použití webových aplikací je v současné době velmi široké. Je možné pozorovat jejich nasazení i v takových oblastech, které byly ještě donedávna doménou pouze klasického desktopového software, jenž ke své práci nevyužíval připojení k síti Internet. Společnost Google nabízí svým registrovaným uživatelům od roku 2007 službu Google Docs, která může sloužit jako ukázka této webové aplikace. Jedná se o sadu kancelářských webových aplikací, která obsahuje textový, tabulkový a grafický procesor a nástroj na tvorbu prezentací a formulářů. Z oblíbených českých webových aplikací je možné uvést například službu Mapy.cz, kterou provozuje společnost Seznam. Tato aplikace nejenže zobrazuje mapové podklady, ale dokáže také nalézt během několika málo vteřin trasu mezi dvěma libovolnými body na území Evropy. Tuto trasu je možné následně exportovat do GPS navigací. Pro běžné uživatele výše uvedené webové aplikace mohou plně nahradit jejich desktopové alternativy, tedy kancelářský nebo geografický software. Je tedy zřejmé, že si webové aplikace začínají vydobývat své pevné místo mezi softwarovými produkty obecně.

Členové výzkumné skupiny WEBING na katedře počítačů ČVUT FEL dokonce ve své publikaci [1] uvádí, že webové aplikace znamenají zásadní předěl v oblasti informačních technologií. Píší: „Během posledních padesáti let se výpočetní technika ve způsobu využití změnila zcela zásadně a web tuto změnu akceleroval dalším kvalitativním způsobem. Informační technologie zaznamenaly několik významných změn v minulosti (vnější paměťová média, mikroprocesory, PC, apod.) a odpověděly vznikem nových oblastí studia a výzkumu. Vývoj webových aplikací se nyní stává jedním z těchto předělů ve vývoji a představuje výzvu pro další vývoj a výzkum.“

Výše uvedené důvody vedou ke srovnání desktopových a webových aplikací, kterým se v dalších kapitolách bude práce věnovat. Srovnání bude provedeno jak z pohledu programátora tak i uživatele. V závěru se představí modelová webová aplikace, která byla implementována pro potřeby této práce a jejíž funkcionalita bude porovnána s její desktopovou alternativou.

3. Vývoj webových aplikací

Cílem této kapitoly je seznámit čtenáře s vývojem webových aplikací. Text se bude nejdříve věnovat požadavkům, které jsou na webové aplikace obecně kladeny a představí jejích vnitřní architekturu. Dále se bude zabývat specifiky webových aplikací oproti desktopovým při jejich vývoji a to především na úrovni aplikační a datové vrstvy. Jmenovitě budou diskutována témata zabezpečení, uchování stavových dat a také práce s dočasnou pamětí.

3.1. Požadavky

Stěžejní požadavek na webové aplikace spočívá v přiblížení jejího ovládání a funkcionality co nejblíže ke klasickým desktopovým aplikacím. Nejlépe tak, aby mohli uživatelé při práci s nimi používat návyky, které mají z práce s desktopovými aplikacemi. Dílčí požadavky vyplývající z tohoto stěžejního požadavku rozdělme do dvou základních skupin – technických a ergonomických.

3.1.1. Technické požadavky

Kompatibilita s webovými prohlížeči

V dnešní době je u moderních webových prohlížečů zřejmý trend s přibližováním interpretace vstupních dat ke standardům W3C. I přes to ale problém s kompatibilitou přetrvává a při vývoji webové aplikace, především jejího uživatelského rozhraní, musí být na tento fakt brán zřetel.

Typickým příkladem nekompatibility jsou chyby při zobrazování kaskádových stylů, ať už se jedná o jejich chybnou interpretaci nebo různé implicitní hodnoty CSS vlastností napříč spektrem prohlížečů. Tento problém je často řešen zavedením tzv. CSS hacků, kterými je možné určit, jaké bloky stylů budou které prohlížeče aplikovat. Toto řešení je sice funkční, ale rozchází se se standardy W3C a není tedy zaručená jeho funkčnost do budoucna, především při vydání nových verzí prohlížečů. Jediným validním řešením problému zůstává identifikace prohlížeče již na straně serveru a výběru kaskádových stylů, které jsou pro něj určeny. I tak ale není funkčnost do budoucna zaručena, protože nutnost separátní větve s kaskádovými styly pro nějaký konkrétní prohlížeč značí jistý odklon tohoto prohlížeče od standardu, který může být v jeho nové verzi revidován a tím vznikne nutnost zavést další větev stylů pro tuto novou verzi.

Problémem, který přímo souvisí s kompatibilitou prohlížečů, jsou i možnosti uživatelských nastavení, které mění výsledný vzhled stránky nebo i její funkčnost. Běžné prohlížeče nabízí funkce pro změnu velikosti písma, potlačení kaskádových stylů nebo vypnutí JavaScriptu. Především vypnutí JavaScriptu má nepříjemné dopady na funkčnost stránky. Pokud je dynamicky generovaná technologií AJAX, tak se stává prakticky celá nefunkční. Je nutné tuto situaci ošetřit a její chování

simulovat standardním pohybem po hypertextových odkazech s načítáním celé stránky. Tímto se přidává na složitosti celé situace.

Pokud bude webová aplikace používaná lokálně (např. jako firemní informační systém na síti intranet) nebo úzkým okruhem uživatelů, tak je možné aspekt kompatibility při vývoji vypustit, vyhlásit požadavky na klienta jako je typ prohlížeče nebo minimální rozlišení obrazovky a při vývoji aplikaci optimalizovat pouze pro definované prostředí.

Rychlost

Rychlost aplikace můžeme vyjádřit jako dobu, po kterou trvá zpracování požadavku uživatele a vykreslení (oznámení) jeho výsledku.

Pokud se oprostíme od graficky náročných aplikací, tak můžeme zjednodušeně tvrdit, že rychlost je u aplikačního software určena pouze rychlostí samotného výpočtu požadavku. Situace u webových aplikací je odlišná. Rychlost se snižuje o dobu přenosu výsledku od serveru ke klientovi, do něhož vstupují i další proměnné, které většinou nemůže programátor ovlivnit. Především to je rychlost připojení serveru a klienta k síti.

Uživatelé jsou s touto situací většinou seznámeni a při využívání webových aplikací s prodlevou několika málo vteřin, na rozdíl od desktopových aplikací, počítají. Problém ovšem nastává v případech, kdy doba prodlevy překročí únosnou mez. Informovat uživatele o postupu vyřízení požadavku formou skutečného progress baru není často z technických důvodů vůbec možné. Zpravidla bývá uživateli po odeslání požadavku zobrazen pouze animovaný obrázek vyjadřující zaneprázdnění systému nebo simulující progress bar. Tento obrázek samozřejmě nemá žádnou vypovídací hodnotu o tom, co se skutečně děje a tak uživatelé po delším čekání své požadavky často zbytečně ruší.

Delší doba odezvy je bohužel charakteristický rys webových aplikací, který se projevuje i při jejich používání na rychlých lokálních sítích. Reálné možnosti programátora, jak dobu odezvy snížit, jsou omezeny pouze na optimalizaci skriptu tak, aby byl požadavek na straně serveru obsloužen co nejrychleji. Je ovšem nutné podotknout, že uživatel tuto optimalizaci pozná pouze ve specifických případech náročných výpočtů na straně serveru.

Limity pro uživatele

Programátor, který je bez možnosti zasáhnout do konfigurace serveru, na kterém bude jeho webová aplikace nainstalována, mívá určené limity pro vykonávání svých skriptů administrátorem. Tyto limity se liší podle použitých technologií, jako příklad můžeme uvést:

- limit pro přidělenou operační paměť,
- maximální čas, po který může být skript vykonáván,

- povolené knihovny,
- maximální velikost uploadovaného souboru.

Tyto limity se musí promítnout do omezení uživatelů, které pro ně budou znamenat jistý diskomfort. Může se jednat o nutnost zmenšení rozměrů surových fotografií z digitálních fotoaparátů před uploadováním na server, kde se s fotografiemi dále pracuje. Mohou se vytvářet náhledy, přidávat vodoznak, atp. Velké obrázky totiž často není možné uložit do přidělené operační paměti a tím ani obsloužit uživatelův požadavek.

Limity bývají často velmi striktně určené u hostingových služeb, kde jeden server sdílí více aplikací. Zde jsou ovšem webové aplikace nejčastěji nasazovány, proto programátor musí s těmito možnými omezeními počítat již při vývoji a vhodně tyto situace ošetřit.

3.1.2. Ergonomické požadavky

Uživatelské rozhraní

Uživatelské rozhraní webové aplikace zobrazované v okně internetového prohlížeče se značně liší od uživatelského rozhraní aplikačního software, které je vytvořené pomocí grafických knihoven. To je důvod, proč by při jeho návrhu neměly být slepě aplikovány standardy pro uživatelské rozhraní aplikačního software.

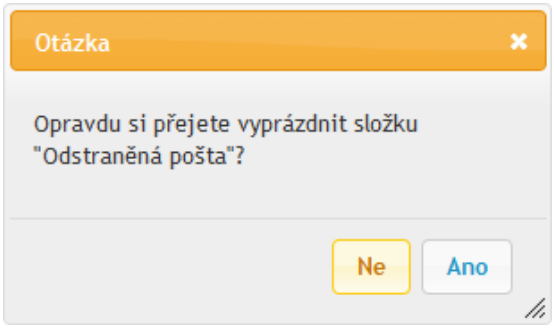
Uživatelé jsou nároční a často očekávají od webové aplikace jistou atraktivitu. Je to dáno především tím, že grafická kvalita stránek na síti Internet stále roste a uživatel se na webovou aplikaci dívá jako na jakoukoliv jinou stránku. Přesto by měla být zachována přehlednost jednotlivých bloků a jejich logické rozdělení, kontrastnost písma a celková grafická přívětivost. Vytvoření návrhu takového kvalitního a uživatelsky atraktivního rozhraní je obvykle mimo možnosti programátora a tak bývá často tento úkol delegován na grafika.

Ovládání

Standardní webové stránky jsou *ovládány* hypertextovými odkazy. Uživateli je po kliknutí odeslána do prohlížeče celá nová stránka jako statický dokument ve formátu HTML nebo XHTML.

Protože moderní webové aplikace nepoužívají tento přístup, předpokládá se i zavedení nových prvků v jejich ovládání. Právě kombinací technologie AJAX s přidáním funkčnosti skriptováním na straně klienta je možné uživatelům nabídnout interaktivní ovládání aplikace, které bylo dříve doménou pouze aplikačního software.

Práce programátora je navíc usnadněna existujícími knihovnami, které již nabízí hotová řešení pro ovládání uživatelského rozhraní. Mezi nejpoužívanější



Obrázek 1. Dialog s otázkou vytvořen knihovou jQuery UI.

knihovny v této oblasti patří jQuery User Interface, která je postavená na JavaScritpovém framework[1] jQuery. Tato knihovna nabízí sadu funkcí pro podporu interakcí, s jejíž pomocí je možné docílit ovládání metodou drag-and-drop (táhni-a-pusť), vybírat elementy klikáním, tažením je označovat nebo měnit jejich velikost.

Knihovna jQuery UI navíc doplňuje i některé chybějící prvky, které mají programátoři desktopových aplikací běžně k dispozici, nicméně ve standardech HTML chybí. V současné verzi[2] jsou doplněny následující prvky:

- tlačítka s ikonami,
- formulářový prvek pro výběr data,
- dialogová okna,
- progress bar (ukazatel postupu),
- track bar (posuvník),
- záložky.

Tato knihovna tedy představuje ucelenou sadu nástrojů, s jejíž pomocí je možné jednoduše přiblížit chování a ovládání webové aplikace aplikacím desktopovým a zvýšit tím její uživatelskou přívětivost.

Pohyb v aplikaci

Ačkoliv webové aplikace většinou nejsou podobně strukturované, jako klasické webové stránky, tak by měly dodržovat zásady, které u nich platí pro navigaci. I když je uživatelům nabídnuto rozhraní s pokročilým interaktivním ovládáním, tak musí být z pochopitelných důvodů možné k aplikaci a jejímu ovládání přistupovat jako ke klasické webové stránce. Proto by měla mít webová aplikace následující vlastnosti:

[1] aplikační rámec; slouží jako podpora při vývoji software
[2] 1.8

- Uživatel by měl být vždy informován o tom, kde se nachází.
- Funkční část je oddělená od části informativní.
- Pro pohyb mezi statickými stránkami slouží klasické navigační prvky.
- Pokud se při pohybu mění obsah, tak je možné se vrátit na předchozí stav zobrazeného dokumentu tlačítkem Zpět pro procházení historie internetového prohlížeče.

I taková samozřejmost jako je zachování funkčnosti tlačítka Zpět je nutné u webových aplikací ošetřit. Při dynamickém načítání obsahu se totiž nemění URL prohlíženého dokumentu a je tedy nutné vypomoci si jinak. Běžnou praxí je, že se při změně obsahu změní také část URL obsahující identifikátor pozice stránky[3]. Tyto změny si prohlížeč ukládá do své historie. Při kliknutí na tlačítko Zpět je vyvolána událost, která zkontroluje hodnotu tohoto identifikátoru a podle něj načte příslušný obsah.

3.2. Architektura

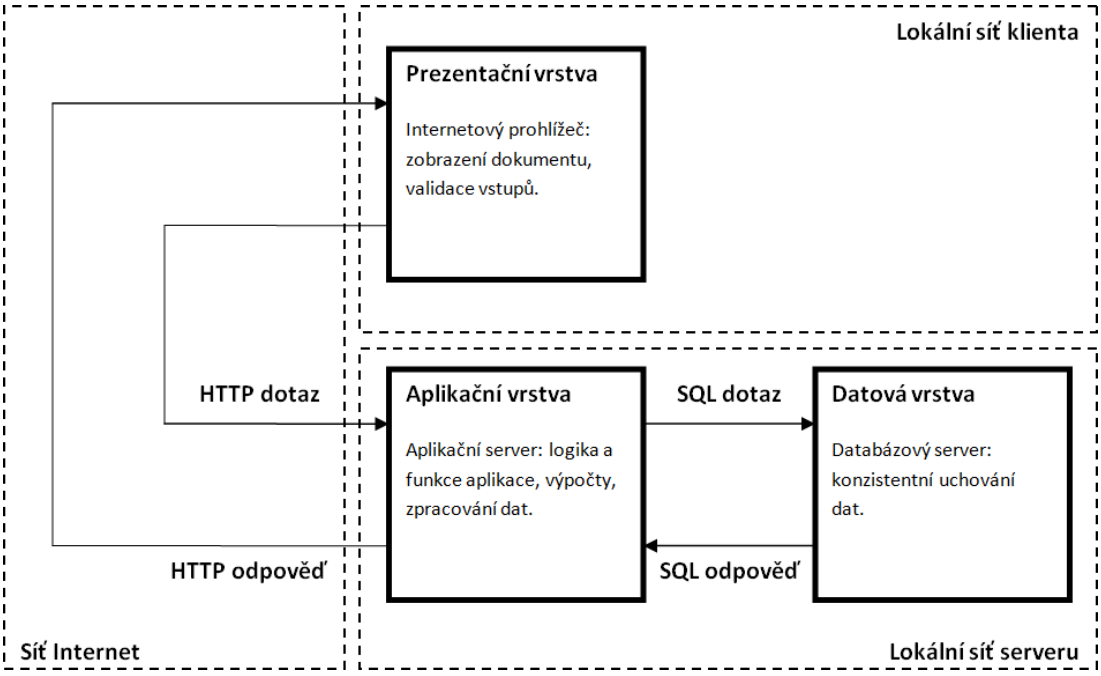
Moderní webové aplikace nejčastěji používají třívrstvou architekturu, jejíž struktura je stejná jako u desktopových aplikací. Obsahují tedy prezentační, aplikační a datovou vrstvu, které mezi sebou komunikují.

V různých zdrojích je možné najít drobné niance struktury webových aplikací, například zařazení validační vrstvy mezi aplikační a prezentační vrstvu, připojení vrstvy s obchodní logikou k aplikační vrstvě a podobně. Většina úprav struktury aplikace je možné interpretovat pouze jako rozdělení některé z původních vrstev nebo separování části jejích funkcí do nové oddělené vrstvy. Důvody pro zavádění nových vrstev jsou stejné, jako u desktopových aplikací, nejčastěji se jedná o:

- Charakteristický proces vývoje, kdy oddělené týmy pracují na svých vlastních vrstvách, za jejichž funcionalitu zodpovídají.
- Použití technologií, které to dovolují nebo vyžadují.
- Zvýšení přehlednosti kódu v rozsáhlých aplikacích.

Podíváme-li se ovšem na propojení jednotlivých vrstev typické webové aplikace, tak jak jej zobrazuje obrázek 2., zjistíme odklon od desktopových aplikací. Prezentační a aplikační vrstvy zde mezi sebou komunikují po síti Internet, zatímco u desktopových aplikací figuruje prezentační a aplikační logika jako jeden celek, který si uživatelé nainstalují do svého operačního systému. Pro komunikaci mezi těmito vrstvami nepoužívá žádného specifického přenosového média. Je ji

[3]angl. fragment identifier; označováno též jako hash



Obrázek 2. Architektura a propojení vrstev typické webové aplikace.

tak možné označit jako lokální a bez časových prodlev, které jsou charakteristickým nedostatkem webových aplikací diskutovaných v kapitole 3.1.1.

Specifickým vlastnostem jednotlivých vrstev se budou věnovat následující podkapitoly.

3.2.1. Prezentační vrstva

Prezentační vrstva obecně představuje data zobrazována uživateli a to většinou v grafické formě. U webových aplikací se prezentační vrstva skládá nejen ze zobrazovaných dat, ale i jejich interpretu, kterým rozumíme internetový prohlížeč. Důvod jeho zahrnutí do této vrstvy je především charakteristické postavení v rámci celé webové aplikace.

Samotné generování kódu pro prezentaci uživateli obstarávají nejčastěji šablonovací frameworky nebo knihovny, které zajišťují úplné odstínění aplikační a prezentační logiky. Jejich konkrétní použití je závislé na technologiích aplikační vrstvy.

Tímto přístupem jsou dodrženy principy třívrstvé architektury, které byly často u webových aplikací pomíjeny. Právě míchání aplikační a prezentační logiky bylo běžnou praxí. Přímo do jazyka HTML byly psány bloky kódu, které se vykonávaly na serveru. Důsledkem potom byla především problematická správa. Jakoukoliv změnu prezentačních dat musel provádět programátor namísto kodéra. Dále také prakticky nulová šance změny struktury generovaného dokumentu na-

příklad v případě změny designu.

Dříve byla tato problematika opomíjena, nebyla příliš diskutována a literatura se jí nevěnovala. Například navíc oblíbený skriptovací jazyk PHP pro tvorbu webových aplikací k tomuto přístupu přímo vybízí. V jeho dokumentaci jsou stále uváděny příklady, které nedodržují oddělení aplikační a prezentační vrstvy, přestože existuje oficiální šablonovací framework pro tento jazyk – Smarty. Programátoři webových aplikací tak často začali dodržovat principy třívrstvé architektury až pod tíhou důsledků.

3.2.2. Aplikační vrstva

Aplikační vrstva obsahuje logiku aplikace. Její interpret nebo platforma pro spouštění bývá zpravidla součást HTTP serveru, který přijímá požadavky od klienta.

Volba konkrétní technologie se odvíjí nejen od charakteru aplikace, ale také od způsobu nasazení. Bude-li vyvíjená aplikace nasazena na vlastní server, k němuž má programátor administrátorská práva, tak je volba technologie bez jakýchkoliv omezení a pouze na uvážení programátora.

Tato situace ovšem reálně nenastává. Běžně jsou webové aplikace nasazovány využitím různých hostingových služeb na servery, kde sdílí jejich výkon s ostatními aplikacemi. Každá aplikace má pak administrátorem nastavené striktní limity, jejichž překročení končí zpravidla chybou. Vzniká tak situace, kdy se konfigurace serveru přímo odráží do prostředí běhu aplikace. Tímto se stává nejisté a značně variabilní, čímž se velmi odlišuje od desktopových aplikací, kde zpravidla dostupnost určité verze platformy pro spuštění aplikace, která je zkontrolována jedou při startu, přímo implikuje její bezproblémový běh.

Ještě složitější situace nastává v momentě, kdy je vyvíjená aplikace určená pro široké spektrum uživatelů, tak aby ji bylo možné provozovat na většině serverů. Sítem technologií propadá pouze programovací jazyk PHP, který se stal jakýmsi standardem pro webové aplikace a jeho interpret je dostupný prakticky na všech serverech i základních hostingových služeb. Ačkoliv je PHP oblíbený a široce používaný jazyk, tak z jeho použití plynou některé charakteristické nedostatky.

Často mu je vytýkána nekoncepčnost, která je důsledkem snahy udržení kompatibility s předchozími verzemi jazyka, nicméně největší problém spočívá v jeho extrémní závislosti na konfiguraci interpretu. Nejedná se přitom pouze o běžné limity jako například přidělenou paměť nebo povolené knihovny, ale jazyk sám jde ještě dál a nabízí direktivu `safe_mode` k zakázání zcela běžně používaných funkcí pro práci se soubory. Důvod pro její zavedení je zvýšení zabezpečení sdíleného hostingového serveru. Toto je charakteristická ukázka problému s koncepcí. Jazyk by neměl docílit bezpečnosti v případě, kdy jeden interpret používá více aplikací tím, že bude programátorovi zakázáno pracovat s funkcemi z nativní knihovny, nota bene s takovými, které jsou velmi často používány, a můžeme je najít ve většině webových aplikací.

3.2.3. Datová vrstva

Datová vrstva se u webových aplikací neliší od desktopových aplikací, její úlohou je konzistentní uchování dat.

Nejčastěji je realizovaná databázovým systémem na samotném databázovém serveru, který se většinou nachází na lokální síti společně s webovým serverem. Nejpoužívanějšími databázemi v tomto prostředí jsou MySQL a PostgreSQL.

Je možné se také setkat s realizací této vrstvy klasickým souborovým systémem nebo také s kombinací souborového a databázového systému. Tento přístup je výhodný při ukládání většího množství velkých souborů, jako jsou například fotografie ve fotogalerii. Soubor je uložen do souborového systému a do databáze se zanese pouze odkaz na něj s příslušnými informacemi potřebnými pro jeho vyhledávání.

3.3. Zabezpečení

Na zabezpečení webových aplikací je třeba nahlížet zcela jinak nežli na zabezpečení desktopových aplikací.

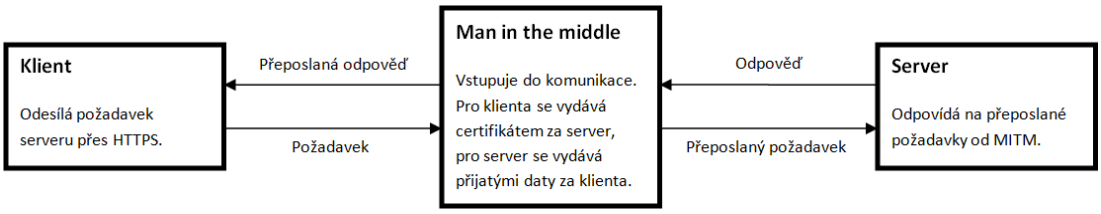
U offline desktopových aplikací pojmem „bezpečná aplikace“ rozumíme spíše fakt, že její používání nebude mít negativní dopad na ostatní software (především operační systém) nebo hardware. U osobního počítače také často odpadá problém s možným odcizením dat, které daná aplikace používá, protože jsou lokálně uložena a fyzický přístup k nim má pouze uživatel daného počítače. Pokud tomu tak není, je možné využít software pro šifrování souborů nebo celého disku. Tyto nástroje zpravidla nebývají součástí běžných desktopových aplikací. Můžeme proto tvrdit, že otázka zabezpečení dat se běžného aplikačního software vůbec netýká.

Zabezpečení webových aplikací má zcela jiný charakter, obecně jej lze rozdělit do dvou základních rovin – bezpečný přenos dat po cestě mezi klientem a serverem a bezpečné ukládání dat na vzdáleném serveru. Specifické postavení v této otázce mají uživatelské vstupy, kterým je věnována samostatná kapitola 3.3.3.

3.3.1. Přenos dat

Protokol HTTP je otevřený, a požadavky od klienta i odpovědi od serveru je možné odposlouchávat. Právě z požadavků je možné zachytit mimo jiné uživatelské vstupy z formulářů nebo aktuální cookies, což může pohodlně stačit například k odhalení hesla. K tomu, aby nebylo možné protokol odposlouchávat, slouží jeho nadstavba HTTPS. Po protokolu HTTPS proudí stejná data jako po protokolu HTTP, nicméně jsou kódována asymetrickou šifrou. Obecně je protokol HTTPS chápán jako řešení problému otevřenosti protokolu http a nejsou vedeny žádné diskuze zásadně zpochybňující jeho bezpečnost.

Achillova pata tohoto protokolu ovšem spočívá v certifikátech, který musí mít každá webová aplikace využívající HTTPS a především certifikačních autoritách,



Obrázek 3. Schéma HTTPS útoku man in the middle.

které jej přidělují. Není-li certifikát, který webová aplikace používá podepsaný důvěryhodnou certifikační autoritou nebo není-li přímo schválen uživatelem, tak je zobrazena informace o bezpečnostních rizicích, která nezkušeného uživatele zajisté mate, možná i vystraší. Uživatelé tak nabudou dojmu, že v případech, kdy internetový prohlížeč indikuje používání HTTPS protokolu, nejčastěji vyobrazením zamčeného zámku, jsou v naprostém bezpečí a žádné riziko odposlechu nehrozí. Reálná situace je ale ovšem taková, že pouze důvěřují autoritě, které jim bylo doporučeno důvěřovat.

Vzniká zde jakýsi morální problém, kdy autoři internetových prohlížečů, které jsou zdarma, a tím pádem nijak neručí za jejich funkčnost, dodávají i sadu přednastavených důvěryhodných certifikačních autorit, kterým mají uživatelé důvěřovat. Realita je ovšem taková, že některé z těchto autorit přidělují certifikáty prakticky s nulovou autentizací žadatele, čímž je možné získat bezpečností certifikát na jakékoliv jméno a jakoukoliv doménu, který je potvrzený důvěryhodnou autoritou, jinými slovy autoritou, kterou vydavatel prohlížeče používaného uživatelem označil za důvěryhodnou. Po získání tohoto certifikátu je možné provádět útoky typu man in the middle[4], kdy se útočník vydává za stranu, se kterou chce uživatel zabezpečeně komunikovat. Získaná data pak zasílá jejímu původnímu příjemci. Jeho odpověď je pak obdobně zaslána uživateli. Schéma tohoto útoku popisuje obrázek 3.

Tento problém není možné z pozice autora webové aplikace odstranit. Vhodné by proto bylo upozornit uživatele na možná rizika a nabídnout stručný návod na to, jak je minimalizovat (odstranění nejméně důvěryhodných autorit, kontrolování certifikátů, atp.). Toto jsou ovšem informace, kteří běžní uživatelé ignorují.

3.3.2. Uchování dat

Charakteristické je pro webové aplikace uchovávání dat na vzdálených serverech, o kterých nevíme, kdo k nim má reálný přístup a to ať už fyzický nebo systémový. Toto je klasický problém při využívání hostingových služeb nebo sdíle-

[4]Zkratka MITM; Patří mezi nejznámější problémy v informatice a kryptografii. Jeho podstatou je snaha útočníka odposlouchávat komunikaci mezi účastníky tak, že se stane aktivním prostředníkem. Důležitým faktem je, že v prostředí současných běžných počítačových sítí není nutné, aby komunikace přes útočníka přímo fyzicky procházela, protože lze síťový provoz snadno přesměrovat.[6]

ných serverů. Přetrvává ovšem i v případě, že má aplikace svůj vyhrazený server, který se ale nachází v housingovém centru, k němuž má přístup jeho administrátor. Ve většině případů tedy nastává situace, kdy k datům, které používá webová aplikace, má přístup nějaká třetí osoba.

Tento problém je možné řešit podobným přístupem jako u bezpečnostních certifikátů a to vybrat si důvěryhodného poskytovatele služeb, o kterém budeme přesvědčeni, že data nezneužije a postará se o jejich bezpečnost.

Mnohem elegantnějším a skutečně funkčním řešením je kódování ukládaného obsahu do databáze a souborů za pomoci šifry a tajného klíče. Takto zašifrovaná data pak může dekódovat pouze jejich majitel, tedy ten, kdo je na server uložil. Princip realizace tohoto řešení je popsán v kapitole 5.5.

3.3.3. Uživatelské vstupy

Bezpečnost webové aplikace je možné narušit také pouhými uživatelskými vstupy. V současné době již toto riziko není u moderních webových aplikací podceňováno a útočník u nich zpravidla nemá žádné možnosti k jejich úspěšnému napadení. I tak je možné se s ním stále setkat a to především u starších sociálních webů obsahující diskusní fóra, deníky atp.

Dva nejběžnější útoky založené na specifických uživatelských vstupech jsou představeny níže.

XSS a vkládání HTML

Při vkládání uživatelských vstupů do dokumentu je vždy nutné převést znaky, které mají v jazyce HTML speciální význam, na HTML entity, aby se zabránilo jeho možné modifikaci. Nebude-li takto uživatelský vstup ošetřen, může docházet k nebezpečnému vkládání škodlivého kódu. Je možné rozlišovat dvě varianty útoku: XSS a vkládání HTML kódu.

Cross site scripting známý pod zkratkou XSS představuje metodu podstrčení JavaScriptového kódu útočníka do dokumentu. Tento kód může například odesílat všechny uživatelské vstupy na stránce, kde se nachází, včetně hesel, na vzdálený server. Na tomto serveru jsou archivovány a analyzovány, následně mohou sloužit jako základ k prolomení uživatelských účtů.

Méně sofistikovanou, nicméně stále nebezpečnou, metodou je vkládání HTML kódu. Uživatelům tak může být například podstrčen odkaz se session klíčem, který bude sloužit k session fixation útoku diskutovaném v kapitole 3.4.5.

SQL injection

Klasický způsob napadení databáze se nazývá SQL injection a jak název napovídá, jedná se o vkládání SQL kódu do dotazu, který je dynamicky generován na základě uživatelského vstupu. Pokud není vstup proti tomuto útoku ošetřen, tak útočník může získat přístup k celé databázi. Limitující pro něj budou v

tomto případě pouze privilegia, která mu jsou přidělena pro práci s databázovým systémem. S tímto typem útoku se můžeme setkat i u desktopových aplikací využívající databáze, nicméně vzhledem k charakteru používání webových aplikací bývají tyto útoky časté právě zde.

Ukázku tohoto útoku budeme demonstrovat na jednoduchém příkladu. Uživatel v aplikaci napsané v jazyce PHP zadává do vstupu identifikační číslo (klíč) řádku, který chce zobrazit. Po odeslání vstupu metodou POST skript složí dotaz a odešle jej do databáze pro vyhodnocení. Situace při neošetřeném vstupu bude vypadat následovně:

Uživatelský vstup:

```
0;DROP TABLE tabulka
```

Složení dotazu:

```
$dotaz = 'SELECT * FROM tabulka WHERE id=' . $_POST['id'] . ';'
```

Provedený dotaz:

```
SELECT * FROM tabulka WHERE id=0; DROP TABLE tabulka;
```

Způsob ochrany proti těmto útokům záleží na konkrétním použitém databázovém systému a také technologii aplikační vrstvy. Vesměs bývají již v nativních knihovnách obsaženy funkce pro ošetření uživatelských vstupů. Obecně ale můžeme říci, že v případech, kdy se uživatelský vstup objevuje v části SQL dotazu, by mělo být vždy zkontrolováno, zda patří do domény, která mu na jeho místě v dotazu náleží.

3.4. Stav aplikace

U webových aplikací je často potřeba udržovat jejich stav. Je nutné si pamatovat například přihlášeného uživatele, otevřený adresář nebo obsah nákupního košíku. Protokol HTTP, po kterém komunikace se serverem probíhá je ovšem bezstavový. Mezi požadavky, které uživatel odesílá, nevidí server žádnou spojitost. Není tedy schopen bez dalších mechanismů jednoznačně zjistit, z jaké aplikace a od kterého uživatele přišel požadavek. Zde je možné pozorovat další zásadní odklon od desktopových aplikací při vývoji webových aplikací.

Uchovat stav webových aplikací je možné několika způsoby, které jsou obecně použitelné u všech nejčastěji používaných technologií při implementaci aplikační vrstvy. Konkrétním mechanismům se věnují následující podkapitoly.

3.4.1. Stavová data s IP adresou

Data pro udržení stavu aplikace jsou ukládána na straně serveru, například do databázového systému. K datům je přiložena informace o IP adrese uživatele, který je vytvořil pro účely jeho pozdější identifikace. Při každém požadavku na server se provede kontrola, zdali nejsou pro IP adresu, ze které byl požadavek přijat, uloženy data pro udržení stavu. Pokud ano, jsou vrácena a aplikace s nimi může dále pracovat.

Tato metoda je použitelná jen ve velice specifických případech, například u firemních informačních systémů, které jsou implementovány formou webové aplikace pro lokání síť intranet. Pro globální síť Internet je zcela nepoužitelná ze zřejmých důvodů – více uživatelů může sdílet jednu IP adresu nebo jeden uživatel může mít dynamicky přidělovanou IP adresu, jejíž hodnota se mění.

3.4.2. HTTP metoda GET

HTTP metoda GET slouží k předávání metaproměnných (nazývaných také parametr) na webový server. Tyto metaproměnné představují nejčastěji výstup formulářů, tedy uživatelských vstupů webových aplikací a jsou přímo součástí URI požadovaného dokumentu. Jednotlivé dvojice „název=hodnota“ se uvádí za znak otazníku, oddělují se znakem ampersand („&“).

Adresu požadovaného dokumentu je možné měnit tak, aby každý dotaz obsahoval parametr pro definování stavu aplikace. Největším nedostatkem tohoto přístupu je možnost uživatele tyto data měnit pouhým přepsáním URI a jejím znovu odesláním na server. Dále z maximální délky URI plyne omezení velikosti takto předaných dat serveru. Reálně se omezuje pouze na krátké textové řetězce.

Nasazení této metody je vhodné v případech, kdy se data z uživatelského vstupu přímo odráží do stavu aplikace. Jako příklad může posloužit procházení stránek s výsledky vyhledávání, kdy parametr obsahující dotaz vyhledávání stále zůstává neměnný (a udržuje tak stav). Výhoda tohoto přístupu spočívá také v zachování možnosti odkázání se na aktuální dokument a jeho stav pouhým zkopírováním adresy bez dalšího programového ošetření.

3.4.3. HTTP metoda POST

Alternativou k metodě GET je metoda POST. Liší se od sebe způsobem předání metaproměnných. Ty jsou v případě metody POST odesílány v těle dotazu a není tak ovlivněno URI. POST je tak odolnější vůči zásahu uživatele, stále je však možné data jednoduše měnit, protože není využito žádného zabezpečení.

Funkce metody POST není závislá na uživatelském nastavení internetového prohlížeče, což je patrně její jediná výhoda. Je možné ji využít u stavových dat, která nejsou citlivá, ovšem nechceme, aby se nacházela v URI dokumentu.

3.4.4. Cookie

Myšlenku cookies navrhl v 90. letech Lou Montulli, tehdy pracující u firmy Netscape Communications. Název cookie – sušenka asociuje zvyklost ze Spojených států nebo Velké Británie nabídnout účastníkům určitého zájmového spolku nebo skupiny jejich oblíbenou sušenku pro vytvoření příjemnější atmosféry.[7]

Cookies ukládají stavová data lokálně na disk uživatele ve formě textových souborů, bez jakéhokoliv zabezpečení. Tyto data jsou pak připojena ke každému požadavku na server, který si vyžádal jejich uložení. Speciální vlastností cookies je možnost nastavení doby jejich platnosti. Jsou buďto automaticky smazány po uzavření okna prohlížeče, anebo je jim nastavená doba vypršení. Zůstanou tak v počítači uživatele uloženy po přesně definovanou dobu.

Nevýhoda cookies spočívá především ve způsobu jejich ukládání. Uživatel může jejich hodnotu měnit přepisováním souborů nebo je mazat. Dále je možné jejich ukládání zakázat v internetovém prohlížeči čímž se stávají cookies zcela nefunkční.

Zcizením těchto souborů je možné získat stavová data webové aplikace, které jsou pak jednoduše použitelná na jiných počítačích. Z tohoto důvodu by neměly být použity pro uchování citlivých dat, jako je například identifikace přihlášeného uživatele. Zcizení cookies by v takovémto případě znamenalo krádež virtuální identity.

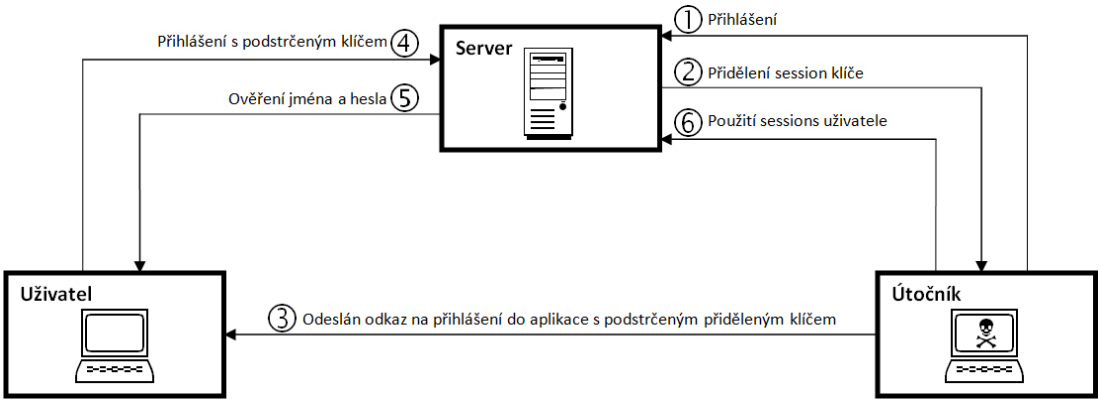
3.4.5. Session

Stavové informace jsou v případě sessions (relace) ukládány na serveru, uživatel tedy nemá šanci s uloženými daty manipulovat nebo je editovat jako u cookies. Při prvním požadavku na jejich uložení nebo čtení je vygenerován náhodný a dostatečně dlouhý klíč, kterým se následně uživatel identifikuje serveru. Tento klíč je serveru předáván při každém požadavku. K tomuto účelu se zpravidla využívá cookie. V případě, že jsou cookies na straně uživatele zakázané, předává se klíč alternativně metodou GET.

Session představují nejčastěji používaný mechanismus pro uložení stavových dat. Jsou považovány za relativně bezpečné, ovšem i tak není zaručeno, že data, které uživatel uložil, nepřečte někdo jiný. Jejich nejslabším místem je právě identifikační klíč, který uchovávají cookies. Nejznámější jsou dva druhy útoků na data uložená v sessions:

Session fixation

Útočník podstrčí uživateli session klíč tak, že mu nabídne odkaz, kde bude klíč předán jako metaproměnná metody GET nebo přepíše jeho hodnotu v souboru cookie. Po tom, co se uživatel přihlásí, použije útočník tento klíč pro sebe. Tím získá uživatelovu identitu v napadené webové aplikaci.



Obrázek 4. Schéma útoku session fixation.

Ochrana proti tomuto typu útoku spočívá ve změně klíče při provádění citlivých akcí, jako je právě přihlášení uživatele do aplikace. Původní klíč ztratí svou platnost a uživateli je vygenerován klíč nový, kterým se bude dále identifikovat. Klíč podstrčený útočníkem pak není možné dále použít.

Session hijacking

Při tomto útoku získá útočník session klíč uživatele, který pak následně použije a získá tak jeho stavová data, zpravidla včetně jeho identity. Klíč musí být použit do doby, než se uživatel odhlásí a zanikne tak jeho platnost.

Pro získání klíče se nejčastěji používá odposlech HTTP protokolu. Využívá se toho, že hodnota klíče je připojena jako cookie ke každému požadavku na server. Tomuto zcizení klíče je možné zabránit využitím protokolu HTTPS.

Jinou alternativou je získání klíče přímo z uložených souborů s hodnotami cookies na lokálním disku. Ochrana proti takovému útoku je mimo možnosti programátora webové aplikace, jediný možný způsob jak toto riziko minimalizovat (ne ovšem zcela vyvrátit) je dodatečná autentizace uživatele například IP adresou. Především je ale ochrana závislá na zabezpečení (softwarovém i fyzickém) osobního počítače uživatele.

3.5. Dočasná paměť

Dočasná paměť, též označovaná jako cache paměť, dočasné soubory Internetu nebo soubory historie, slouží ke snížení objemu přenášených dat po cestě mezi serverem poskytujícím webovou aplikaci a internetovým prohlížečem na straně klienta, tím pádem i k celkovému zrychlení odezvy aplikace.

Podle druhu paměti se přijaté soubory ukládají na lokálním disku uživatele aplikace (dočasná paměť internetového prohlížeče) anebo na disku proxy serveru (dočasná paměť proxy serveru). Při vyžádání nového objektu, jako je obrázek, soubor s kaskádovými styly, video a podobně, je vždy zkontrolováno, zdali již není

soubor uložen v dočasné paměti. Pokud je, vrátí se jeho uložená kopie namísto odeslání požadavku na server.

Je tedy zřejmé, že je značně urychleno procházení již navštívených dokumentů, ale i načítání grafických webů, kdy se grafické prvky s návštěvou první stránky uloží do dočasné paměti. Ty se na dalších stránkách pouze opakují a je možné používat jejich lokální kopie.

Tento přístup má nevýhodu v případech, kdy je obsah webu generován dynamicky a soubory se stejným názvem mění svůj obsah. Typicky to jsou obrázky, které vytváří webová aplikace sama, jako jsou grafy, náhledy, grafické statistiky atp. Webová aplikace totiž vygeneruje nový obrázek, nicméně uživateli je vrácena jeho stará kopie z dočasné paměti a vzniká tak problém s doručením neaktuálních dat. Aby k tomuto stavu nedocházelo, je potřeba ošetřit ukládání souborů do dočasné paměti. Existuje několik přístupů, kterými toho lze docílit:

HTML meta tag

Do hlavičky HTML dokumentu je možné umístit meta tag, který zapříčiní, že nebude dokument ukládán do dočasné paměti. Toto řešení ovšem přináší dvě zásadní nepříjemnosti, kvůli kterým jej můžeme označit za nefunkční.

Proxy server zpravidla nepročítá obsah dokumentu, proto se běžně ukládá do jeho dočasné paměti i dokument, který obsahuje tento meta tag.

Pro potlačení uložení jednoho souboru do dočasné paměti je potlačeno uložení celého dokumentu se všemi soubory, které se na něm nachází.

HTML meta tag pragma:

```
<META HTTP-EQUIV='pragma' CONTENT='no-cache'>
```

Změna názvu souboru

Řešením může být změna názvu souboru po každé změně jeho obsahu. Toto řešení je funkční, nicméně ne příliš elegantní. Je nutné si pamatovat aktuální název souboru a ten promítat do prezenční vrstvy webové aplikace. Navíc je znemožněno dlouhodobé odkazování na tento soubor z cizích zdrojů, jako jsou například fulltextové vyhledávače. Proto se toto řešení v praxi příliš nepoužívá.

HTTP hlavičky

HTTP hlavičkou Expire je možné docílit u konkrétního souboru nastavení jeho přesného vypršení v dočasné paměti. Po tom, co soubor vyprší, je z dočasné paměti smazán a při dalším požadavku je ze serveru stažena aktuální verze. Tento postup je vhodný v případech, kdy se soubory generují v určitou dobu. Uživateli jsou v dočasné paměti ponechány soubory vždy přesně do doby, než jsou vygenerovány nové.

U souborů, které mění svůj obsah často, v nespecifických intervalech, je nastavená doba vypršení volena jako kompromis – aby se uživateli nezobrazoval příliš dlouho neaktuální obsah a také aby nebyl server zbytečně často zatěžován požadavky na stažení souboru, který se ještě nezměnil.

U webového serveru Apache se pro změny HTTP hlaviček souborů nejčastěji používá konfigurační soubor .htaccess. Je možné nastavit dobu vypršení u jednoho konkrétního souboru, skupiny souborů podle jejich typu nebo obecné nastavení pro všechny ostatní soubory, které nebyly dříve v souboru specifikovány.

4. Uživatelské rozhraní, ovládání

Prezentační vrstva má u webových aplikací specifické postavení, proto je uživatelskému rozhraní a jeho ovládání věnována samostatná kapitola. První část rozebírá technologie, které jsou u uživatelského rozhraní používány a shrnuje jejich základní charakteristiky. Mimo to se také detailněji věnuje očekávané specifikaci HTML 5 a představuje její přínosy. Další text se zabývá vzhledem a ovládáním uživatelských rozhraní webových aplikací a poukazuje na typické odlišnosti od rozraní aplikací desktopových.

4.1. Technologie uživatelského rozhraní

Webové aplikace ve svých uživatelských rozhraních využívají zpravidla zcela jiné technologie než aplikace desktopové, které jsou tvořené pomocí grafických knihoven. Přehledu těchto technologií a jejich charakteristickým vlastnostem se věnují následující podkapitoly.

4.1.1. HTML 4.01

Zkratka HTML znamená hypertext markup language, což je možné přeložit jako odkazovací značkovací jazyk. Představuje základní nástroj pro vytváření dokumentů určených k prezentování na síti Internet. Jazyk se skládá ze značek nazývaných tagy, kterými se určuje forma a struktura dokumentu.

Aktuální verze jazyka je 4.01 vydaná 24. prosince 1999. Její kompletní specifikace je možné nalézt na [9]. S vydáním této verze byl vývoj ukončen a dále se pracovalo pouze na normě XHTML, o níž se předpokládalo, že bude nástupce HTML.

V roce 2007 ovšem vznikla pracovní skupina, která má za cíl vytvořit nový standard HTML. V této skupině figurují W3C HTML WG[5] a WHATWG[6], jejíž specifikace Web Applications 1.0 a Web Forms 2.0 jsou základem budovaného standardu. Hlasováním bylo rozhodnuto, že ponese označení HTML 5.

4.1.2. HTML 5

Aktuální verze HTML 4.01 jen opravuje chyby verze 4.0 z 18. prosince 1997. Všechny později vydané verze tohoto jazyka (XHTML) se omezují v zásadě pouze na jeho syntaktické úpravy a omezení některých tagů. Programátoři webových aplikací tedy nyní pracují se značkovacími jazyky, které jsou pouze syntaktickými variantami více než 13 let starého jazyka. To je při rychlosti vývoje Internetu neudržitelný stav.

[5] W3C HTML Working Group
[6] Web Hypertext Application Technology Working Group

Nový standard HTML nepřináší pouze důležitou revizi tagů, ale také novou funkcionalitu pro webové aplikace. HTML 5 tedy můžeme chápat jako sadu nové specifikace jazyka HTML a technologií pro JavaScript úzce spolupracujících s internetovým prohlížečem.

Představme si zásadní novinky HTML 5 a jeho technologií.

Offline webové aplikace

Jednou z velkých a často diskutovaných novinek HTML 5 je sada technologií pro offline práci s webovými aplikacemi. Těmi by měla být odstraněna jedna z jejich velkých nevýhod a to závislost na připojení k síti Internet.

S myšlenkou možnosti využívání offline webových aplikací již přišla společnost Google. Vyvinula technologii Gears [3], která umožňuje lokální uložení dat používaných webovými aplikacemi. Pokud nemá uživatel dostupné připojení k síti Internet, jsou využívána tato lokální data. Synchronizace offline dat na lokálním disku uživatele a online dat na serveru poskytovatele aplikace se provede po prvním využití aplikace v online režimu. Tato technologie je ovšem dostupná pouze pro omezenou sadu internetových prohlížečů. V internetovém prohlížeči Google Chrome je již nativně předinstalovanána a pro prohlížeče Firefox (od verze 1.5) a Microsoft Internet Explorer (od verze 6.0) je instalována jako jejich rozšíření.

Gears je distribuováno jako open source na základě BSD licence a programátoři jej tedy mohou volně používat ve svých webových aplikacích. Mimo aplikace provozované společností Google je například Gears nasazeno na službách YouTube nebo MySpace. Vývoj technologie Gears byl s vydáním první stabilní verze 22. února 2010 ukončen. Důvodem bylo právě jeho krytí s funkcionalitou pro offline aplikace standardu HTML 5.

HTML 5 využívá dva nástroje pro provoz offline webových aplikací – Application Caches (aplikační dočasná paměť) a Web Storage (webové úložiště). Dočasná paměť se používá pro uložení souborů aplikace, které mají být dostupné i v případě, že není k dispozici internetové připojení. Především to budou soubory obsahující logiku aplikace pro běh v režimu offline a soubory s grafikou. Konkrétní soubory, jež mají být takto uloženy, jsou specifikovány v souboru nazývaném manifest. Na druhé straně se webová úložiště používají pro uchování dat uživatele na lokálním disku. Jsou definovány Session Storage (úložiště relace) a Local Storage (lokální úložiště), které se liší pouze stálostí uložených dat. Úložiště relace uložená data uživatele maže po uzavření okna prohlížeče, zatímco lokální úložiště ukládá data trvale. Hodnoty se do úložiště ukládají s klíčem, pomocí něhož se k nim následně přistupuje. Jedná se tedy o podobný přístup jako při práci s asociativním polem.

HTML 5 tak zavádí (ne)zcela nový přístup uživatelů k webovým aplikacím. Může být chápán pouze jako záloha pro případ výpadku sítě Internet. Její největší přínos ale ocení mobilní uživatelé. Používání webových aplikací je pro ně zpravidla spjato s dostupností a kvalitou signálu sítí GSM nebo Wi-Fi, což například mimo

hranice měst bude často znamenat problém. Po odpadnutí této závislosti je možné k webovým aplikacím přistupovat jako ke klasickým desktopovým aplikacím.

Tagy pro zvuk a video

Je možné pozorovat zcela zřejmý trend zvyšování rychlosti připojení k Internetu. Uživatelé na domácích linkách často dosahují rychlostí, které byly dříve vyhrazeny pouze serverům. Tyto vysoké přenosové rychlosti otevřely bránu multimediálnímu obsahu internetových stránek. Je tak již zcela běžné například sledovat streamované televizní vysílání ve vysokém rozlišení v okně internetového prohlížeče.

Současné verze jazyka HTML je ovšem poplatná době svého vzniku a neumí s tímto obsahem pracovat. To samé platí i pro jeho XHTML variantu, která ve své podstatě žádné rozšíření jazyka nepřináší. Pro vložení plnohodnotného přehrávače do stránek tak musí programátor využít technologie třetích stran, které se instalují jako rozšíření internetového prohlížeče. K těmto účelům je nyní nejčastěji využíván Adobe Flash. Pokud uživatel tuto technologii do svého prohlížeče odmítne z jakýchkoliv důvodů nainstalovat, tak nemůže multimediální obsah přehrát. Je také nutné zmínit fakt, že pro některé platformy, především u mobilních zařízení, tyto rozšíření neexistují a tak je nemůže jistá skupina uživatelů vůbec využít.

HTML 5 tento problém řeší zavedením tagů <`audio`> a <`video`>, které do obsahu dokumentu vkládají přehrávač hudby a videa bez závislosti na využití jakýchkoliv jiných technologií třetích stran. Zcela tak odpadají problémy s instalací rozšíření prohlížečů a jejich dostupnosti pro specifické platformy.

Strukturování dokumentu

V aktuální verzi HTML není vyjma nadpisů udělen tagům žádný sémantický význam. Celé weby proto jsou zpravidla skládány pouze z různě vnořených bloků vytvářených tagy <`div`> nebo <`span`>, které samy o sobě neudělují informacím, jež obsahují, žádný význam a jsou následně pomocí CSS nastylovány pro uživatele.

HTML 5 z tohoto důvodu zavádí sadu párových tagů, přičemž každý má svůj specifický sémantický význam. Pomáhají lépe strukturovat dokument a zvyšují čitelnost kódu. Jejich použití se také zajisté podepíše na kvalitnějších výsledcích fulltextového vyhledávání v síti Internet.

<`section`> reprezentuje obecnou, tematicky celistvou, část stránky typicky s nadpisem.

<`nav`> reprezentuje část stránky, která odkazuje na jiné stránky nebo na různé pozice aktuální stránky.

<article> reprezentuje samostatnou část stránky, která není závislá na ostatním obsahu. Může se jednat například o diskusní příspěvek, novinový článek, příspěvek na blogu, interaktivní widget či gadget nebo jiné samostatné části stránky.

<aside> reprezentuje část stránky zpravidla obsahující dodatečné informace, které jsou nějakým způsobem spjaty s obsahem okolo, nicméně by od něj měly být odděleny.

<hgroup> reprezentuje nadpis části stránky. Používá se pro seskupení tagů H1 – H6 v případech, kdy má nadpis více úrovní. Nereprezentuje tedy nadpis samotný.

<header> reprezentuje záhlaví stránky. Může obsahovat základní navigaci, úvodní text, logo, atp.

<footer> reprezentuje zápatí části obsahu nebo celé stránky. Obvykle obsahuje informace o části, kterou terminuje jako například jméno autora, odkazy na související stránky, copyright a podobně.

<address> reprezentuje kontaktní informace. Ty mohou být spjaty s celou stránkou jako takovou nebo pouze s obsahem nejbližšího tagu <article>.

Nové formulářové prvky

Při vytváření formulářů má nyní programátor k dispozici velmi omezenou sadu formulářových prvků. U složitých formulářů je tuto sadu zpravidla nutné rozšířit některou z knihoven pro uživatelská rozhraní. Vzniká tím opět závislost na jiných technologiích, i když bez nutnosti jejich dodatečné instalace. HTML 5 tento problém řeší a přidává typy formulářových prvků uvedené v tabulce 1.

HTML 5 dále nabízí u formulářových prvků funkcionalitu, které je nyní možné docílit pouze využitím skriptování na straně klienta. Je to především validace vstupu včetně určení maximální přípustné hodnoty a automatické dokončování.

Závěr

HTML 5, tak jak je specifikováno, nepřináší žádné revoluční novinky. Pouze citlivě reaguje na směr vývoje webových aplikací a jako standard přináší nástroje, pro které musí být nyní využívány různé JavaScriptové knihovny nebo rozšíření internetových prohlížečů.

Toto je ovšem pro programátory webových aplikací velký posun. Omezuje se totiž závislost na externích knihovnách a technologiích třetích stran, které jsou často ve formě různých proprietárních řešení. Odpadají tak časté problémy s kompatibilitou a zvyšuje se přehlednost a efektivita kódu.

Původně se očekávalo uvolnění standardu HTML 5 do roku 2012, kdy by jej mohli začít programátoři využívat. Organizace W3C ovšem vydala 14. února 2011 zprávu [11], ve které odsunula vydání až na rok 2014.

Typ prvku	Význam
`datetime`	výběr nebo zadání data a času v globálním formátu
`datetime-local`	výběr nebo zadání data a času v lokálním formátu
`date`	výběr nebo zadání data
`month`	výběr nebo zadání roku a měsíce
`time`	výběr nebo zadání času
`week`	výběr nebo zadání roku a týdene
`number`	výběr nebo zadání čísla
`range`	výběr rozsahu hodnot
`email`	zadání e-mailové adresy
`url`	zadání URL adresy
`search`	pole pro vyhledávání
`tel`	zadání telefonního čísla
`color`	výběr barvy

Tabulka 1. Nové typy formulářových prvků.

4.1.3. XHTML

XHTML je zkratka anglického extensible hypertext markup language (rozšiřitelný odkazovací značkovací jazyk). Jedná se o značkovací jazyk, který vznikl syntaktickým převodem jazyka HTML na jeho XML variantu. Měl by tedy na rozdíl od HTML zajistit snadnější strojové čtení dokumentů, ale zároveň zachovat zpětnou kompatibilitu.

Přechod od HTML k validnímu XHTML 1.0 programátorům ulehčují tři vydané verze – Strict, Transitional a Frameset. Každá klade jinou úroveň požadavků na strukturu kódu a použité tagy tak, aby i u stránek formátovaných pro staré prohlížeče byla možnost plynulého přechodu směrem k novým standardům.

Současná verze XHTML 1.1 se podobá verzi 1.0 Strict. Jsou z něj ovšem odstraněny překonané tagy uvedené v tabulce 2., které byly oficiálně označeny jako deprecated. Dále je nutné přidat odesílaným souborům, které odpovídají její specifikaci [10], MIME typ application/xhtml+xml.

Mezi nejdůležitější odlišnosti XHTML a HTML patří:

- Dokument musí být vložený do kořenového tagu <`html`></`html`>.
- Párové tagy musí být ukončeny a nesmí se křížit.
- Nepárový tag musí být na konci ukončen lomítkem.
- Všechny atributy tagů musí mít hodnotu.

Tag	Význam
<basefont>	přednastavení fontu pro celou stránku
	vystředěný blok
<font>	změna fontu
<s>	přeškrtnutí textu
<strike>	přeškrtnutí textu
<u>	podtrhnutí textu
<applet>	Java applet
<dir>	seznam adresářů
<menu>	seznam navigačních položek
<isindex>	formulářový prvek

Tabulka 2. Překonané tagy XHTML 1.0.

- Povinně deklarace doctype na začátku dokumentu.
- Tagy a jejich atributy se píší malými písmeny.

Nástupcem současné verze se mělo stát XHTML 2.0. Vývoj byl ovšem zastaven s vývojem standardu HTML 5. Pracuje se totiž na XML variantě HTML 5, jako na součásti jeho specifikace. Ta ponese označení XHTML 5.

4.1.4. CSS

W3C definuje CSS (Cascading Style Sheets) jako jednoduchý nástroj pro přidávání stylů (fonty, barvy, řádkování apod.) do webových dokumentů.

Dříve určovalo způsob zobrazení pouze HTML, které mělo k těmto účelům definováno velké množství tagů a jejich atributů. Hlavním cílem CSS je právě oddělit vzhled (X)HTML dokumentu od informace, kterou nese. Kód dokumentu by tak měl být strukturován logicky podle jeho obsahu a ne podle způsobu zobrazení. Tento přístup umožňuje jednodušší správu a aktualizace dokumentu, jejich programové generování, ale také snadnější vyhledávání informací. Zásadní nedostatek CSS spočívá v kompatibilitě s internetovými prohlížeči diskutovaný v kapitole 3.1.1.

Pro CSS byly vydány dvě specifikace – CSS 1 a CSS 2. Ačkoliv je CSS ve své druhé specifikaci velmi silný nástroj na stylování dokumentů, tak stále má několik velkých nedostatků. Pomineme-li výše zmíněný problém s kompatibilitou, tak se jedná především o:

- jednoduché selektory neumožňující specifičtější výběr stylovaného tagu,
- pouze obdélníkové rámy,
- problémy s vertikálním centrováním.

Toto jsou mimo jiné důvody vývoje nové specifikace CSS 3, který probíhá paralelně s vývojem HTML 5.

CSS 3

S cílem jednoduššího vytváření uživatelsky atraktivních webů odstraňuje CSS 3 výše uvedené nedostatky. Jsou tedy zavedeny sofistikovanější selektory, které jsou schopné vybírat tagy například podle hodnoty některého jejich atributu nebo umístění v dokumentu. Je možné specifikovat rám tagu obrázkem nebo zakulatit jeho rohy. Nový způsob práce s rozvržením elementů také odstraňuje problém s vertikálním centrováním. Nejvíce kritizované nedostatky jsou tedy vyřešeny, CSS 3 jde ale mnohem dále.

Z přidaných grafických prvků bude zcela jistě nejzajímavější zavedení prostorových transformací, díky nimž můžeme elementy na stránce otáčet, zkosit nebo jim měnit měřítko. Tyto transformace se provádí ve 2D i 3D. Společně se zavedením animací, stínů, zrcadlení nebo průhlednosti již nebudou grafici omezeni limity, které jim specifikace CSS 2 velmi striktně určuje. Nové webové aplikace využívající vlastností CSS 3 se tak svým uživatelským rozhraním mohou zcela vyrovnat současnému desktopovému software.

Datum uvolnění standardu CSS 3 nebylo zveřejněno a většina specifikací nových vlastností je v současné době pouze ve formě pracovního návrhu. Reálně lze ovšem již nyní s CSS 3 v moderních prohlížečích pracovat. Většina z nich totiž zavádí své vlastní implementace nových vlastností, které se liší jen prefixem v názvu podle daného prohlížeče. Mozilla používá prefix `-moz-`, Opera `-o-` a Safari `-webkit-`. Tyto prefixy bude možné odstranit až po uvolnění končeného doporučení W3C pro CSS 3. Protože tyto vlastnosti nejsou ve standardech, tak jejich použití způsobí nevaliditu kaskádových stylů.

Tabulka 3. uvádí přehled podpory vlastností CSS 3 napříč spektrem internetových prohlížečů[7]. Je možné si všimnout úplné podpory CSS 3 internetovým prohlížečem Safari od společnosti Apple. Důvodem patrně je, že při vytváření uživatelských rozhraní pro platformy této společnosti je možné již nyní využívat CSS 3. Především aplikace pro iPhone velmi často CSS 3 používají. To můžeme interpretovat jako důkaz tvrzení o srovnání kvality uživatelských rozhraní webových aplikací stylovaných pomocí CSS 3 a klasických desktopových aplikací.

[7] *zdroj dat: http://findmebyip.com*

Vlastnost	Opera 10.63	Firefox 4.03	Safari 5	I. Explorer 9	Chrome 9
Barvy RGBA	+	+	+	+	+
Barvy HLSA	+	+	+	+	+
Velikost pozadí	+	+	+	+	+
Vícenásobné pozadí	+	+	+	+	+
Obrázek ohraničení	+	+	+	−	+
Zakulacené rohy	+	+	+	+	+
Stín bloku	+	+	+	+	+
Stín textu	+	+	+	−	+
Průhlednost	+	+	+	+	+
Animace	−	−	+	−	+
Sloupce	−	+	+	−	+
Gradient	−	+	+	−	+
Zrcadlení	−	−	+	−	+
2D transformace	+	+	+	−	+
3D transformace	−	−	+	−	−
Plynulý přechod	+	+	+	−	+
Vlastní font	+	+	+	+	+
Flexibilní blok	−	−	+	−	+

Tabulka 3. Přehled podpory vlastností CSS 3.

4.1.5. JavaScript

JavaScript je prototypový skriptovací jazyk nezávislý na platformě. Byl vydán v roce 1995 jako společný počin firem Netscape a Sun Microsystems a již od svého vzniku byl určen především jako doplněk k jazyku HTML. Název je mírně zavádějící protože s jazykem Java nemá vyjma spoluautora a podobné syntaxe nic společného. Původní název byl LiveScript, ale před vydáním byl patrně pouze z marketingových důvodů přejmenován na JavaScript.

Charakteristický je pro jazyk jeho interpret. Tím totiž není server, jako je tomu u většiny skriptů používaných u webových aplikací, ale internetový prohlížeč na straně klienta. Tento rys se odráží v jeho typickém použití – vytváření efektů a interakcí v uživatelském rozhraní webové aplikace. Toho je docíleno přístupem

JavaScriptu ke všem objektům, jejich hodnotám a atributům včetně stylů, které se nachází v zobrazeném dokumentu. K tomuto účelů je používán objektový model dokumentu, který W3C právě pro potřeby JavaScriptu standardizovalo. Jeho dřívější nasazení se omezovalo vesměs na jednoduché doprovodné akce typu animace po přejetí myší nebo validace vstupů formuláře. V současné době, kdy má silnou podporu v internetových prohlížečích, představuje základní nástroj k simulování chování uživatelského rozhraní desktopové aplikace.

JavaScript se u větších webových aplikací typicky nepoužívá ve své surové formě, ale spojen s nějakým frameworkem. Těch je v současné době celá řada a pro potřeby každé webové aplikace je možné vybrat takový, který ji bude nabízenou funkcionalitou vyhovovat. Patrně nejčastěji používanými JavaScriptovými frameworky dnes jsou jQuery, Prototype a Dojo.

4.1.6. AJAX

AJAX je možné představit jako technologii vývoje interaktivních webových aplikací, které mění svůj zobrazovaný obsah, aniž by bylo provedeno znovunačtení celého dokumentu. Garrett ve svém článku [4], jenž poprvé AJAX zmiňuje, uvádí, že AJAX není technologie jako taková, nýbrž název pro součinnost několika technologií, které vyjmenovává v těchto pěti bodech:

- prezentační vrstva založená na standardech XHTML a CSS,
- dynamické zobrazování údajů a interakce s uživatelem prostřednictvím DOM,
- asynchronní výměna dat se serverem pomocí rozhraní XMLHttpRequest,
- XML a XSLT pro samotnou výměnu dat,
- JavaScript pro zajištění součinnosti výše uvedených technologií.

Principy popsané v tomto článku z roku 2005 se staly velice rychle populární a AJAX je jedním ze základních stavebních kamenů moderních webových aplikací. Velmi široce je začala využívat především ve svém webovém prohlížeči společnost Google. Systém našeptávání při vyhledávání, kdy se podle rozepsaného dotazu snaží vyhledávač nabídnout jeho dokončení, bývá dáván jako typický příklad AJAX. Často také bývá, ne zcela přesně, uváděn tento příklad jako první použití AJAX vůbec.

Samotná myšlenka na možnost změny obsahu webové stránky bez jejího znovunačtení je starší. První kdo s uskutečněním tohoto přístupu přišel, byl patrně Microsoft, když do svého internetového prohlížeče Internet Explorer 3.0 z roku 1996 implementoval tag <`iframe`>. Tím je možné docílit základního principu technologie AJAX – pro změnu obsahu není nutné obnovovat celý dokument, ale jen jeho část. Tento tag byl následně standardizován ve specifikaci HTML 4.0

v roce 1998. Microsoft také vytvořil rozhraní XMLHttpRequest, které má nezastupitelné místo v sadě technologií AJAX. To bylo poprvé použito ve webové aplikaci Outlook Web Access pro mailový server Exchange v roce 2000. Zde byly poprvé aplikovány všechny principy z uvedeného článku. Můžeme tedy chápat Microsoft i jako prvního, kdo poprvé přišel s přístupem technologie AJAX tak, jak ji chápeme dnes.

Základní přínosy a potíže používání AJAX již vyplývají z uvedeného textu. Alespoň rámcově je tedy shrňme.

Výhody

Zásadní výhoda AJAX je zřejmá a vyplývá z jeho definice. V případě, že není nutné obnovovat celý dokument, je možné vytvářet uživatelská rozhraní webových aplikací, které se svým chováním přibližují k desktopovým aplikacím.

Například chce-li uživatel odpovědět v diskuzi na příspěvek, tak klikne na tlačítko Odpovědět. Místo toho, aby byl přesměrován na novou stránku obsahující formulář pro odpověď, se dynamicky připojí tento formulář přímo pod příspěvek, na který uživatel reaguje. Po vyplnění odpovědi ji odešle tlačítkem Odeslat. Validace vstupů se provede již na straně klienta, a pokud je vše v pořádku, tak zmizí odpovědní formulář a uživatelův příspěvek se rovnou zobrazí v diskusním stromu bez jakéhokoliv načítání celého dokumentu. V případě bez použití technologie AJAX by musel být opět uživatel ze stránky s formulářem, po kliknutí na tlačítko odeslat, přesměrován zpět na původní stránku s diskuzí. Jak je tedy vidět, tak AJAX nejenže zvyšuje uživatelský komfort, ale také snižuje objem přenášených dat mezi serverem a klientem, čímž zrychluje webovou aplikaci a snižuje zátěž serveru.

Nevýhody

V současné době u moderních webových prohlížečů již prakticky neexistují problémy s kompatibilitou AJAX. Všechny potřebné technologie jsou podporovány a bokem stojí jen velmi specifická minoritní skupina prohlížečů, jako jsou prohlížeče s textovým režimem atp. Na straně prohlížeče tak zůstává největší problém, jak je již zmíněno v kapitole 3.1.1., v možnosti nastavení uživatele vypnout JavaScript. Je tedy nutné nabídnout alternativní ovládání, jehož funkčnost není závislá na JavaScriptu. Po struktuře webu, která je takto ovládaná, se ale nepohybují pouze tito uživatelé, ale také fulltextové vyhledávače, které následně indexují obsah stránek. To je další důvod, proč k dynamické variantě ovládání webové aplikace vytvářet paralelně i jeho statickou variantu, která při každém požadavku uživatele načítá znovu celý obsah dokumentu.

Při využití technologie AJAX ve spojení s nějakým specifickým ovládáním webové aplikace, například posuvníky nebo tažením myši, může vzniknout navigační struktura, na kterou principiálně není možné aplikovat tlačítko Zpět pro

pohyb v historii internetového prohlížeče. Nejedná se ovšem o problém technologie jako takové, ale o nedostatek internetových prohlížečů, které prozatím nereagovaly na změnu ovládání webových aplikací. S tímto tématem také souvisí nutnost ošetření funkce tlačítka Zpět v případě dynamického generování obsahu dokumentu, jak uvádí kapitola 3.1.2.

Dále, ačkoliv je s AJAX snížen objem přenášených dat a rychlost webové aplikace se značně zvýší, tak stále dochází k charakteristickým prodlevám, kterým se věnuje kapitola 3.1.1. v oddíle o rychlosti. Se sníženým objemem přenášených dat také souvisí zvýšení počtu dotazů na serveru. Použití AJAX tedy paušálně neznamená snížení zátěže serveru.

4.1.7. Applety

V předchozím textu je možné nalézt termín technologie třetích stran, což je v našem kontextu synonymum pro webový applet. Appletem se obecně označuje komponenta, která je nasazena v rámci jiného programu. U webových aplikací je tímto programem internetový prohlížeč. Applet můžeme chápat jako technologii prezentační vrstvy, která pro interpretaci nebo spuštění svých zdrojových kódů vyžaduje software, který je dodáván většinou jako rozšíření internetového prohlížeče, alternativně se instaluje přímo do operačního systému.

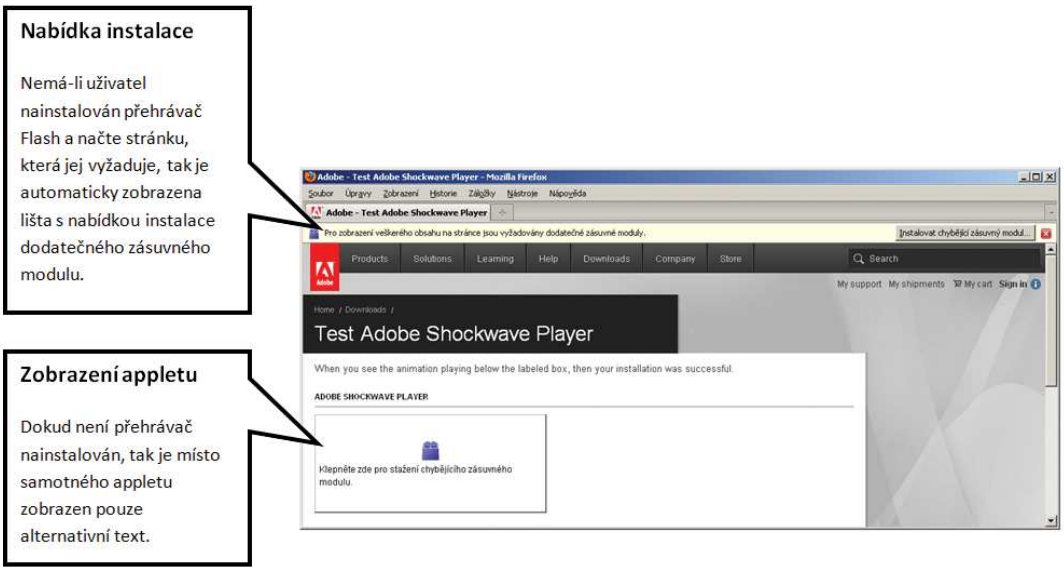
Pod touto optikou bychom mohli mezi tyto technologie zařadit i multimediální přehrávače, jako je například QuickTime, Windows Media Player nebo RealOne, což jsou také zástupci webových appletů, nicméně omezíme se pouze na ty technologie, které vykonávají kód a to ať už ve formě čistého programovacího jazyka nebo mezikódu.

Právě nutná dodatečná instalace představuje problém, kdy uživatelé nemusí mít z jakýchkoliv důvodů přítomný interpret nebo platformu pro spuštění appletu. Proto by applety měly být využívány pouze v opodstatněných případech, kde vynikají jejich specifické vlastnosti a pro docílení stejného efektu či funkcionality není možné použít jazyky, které jsou v prohlížečích nativně podporovány.

Patrně nejpoužívanější webové applety v tomto smyslu jsou Adobe Flash, Microsoft Silverlight a Java applet. V následujícím textu jsou krátce představeny.

Adobe Flash

Adobe Flash (dříve ve vlastnictví společnosti Macromedia) představuje animační platformu, která využívá skriptovací jazyk ActionScript. Je využíván především pro tvorbu vektorových animací, které mohou uživatelé ovládat. Jeho typické nasazení je tedy při vytváření webových stránek, u kterých se očekává vysoká uživatelská atraktivita (prezentace populárních hudebních kapel atp.) nebo u her určených pro internetové prohlížeče. Flash podporuje také práci s videem a zvukem jako takovým, takže je možné se s ním často setkat i ve formě různých multimediálních přehrávačů.



Obrázek 5. Chybějící zásuvný modul přehrávače Adobe Flash a výzva k jeho instalaci v internetovém prohlížeči Mozilla Firefox 3.5.

Tato technologie je nainstalována na většině počítačů. Podle společnosti Adobe dosahuje penetrace až 99 %. Přesto by se k ní nemělo paušálně přistupovat jako k obecně podporované a před jejím použitím by měly být zváženy i tyto její nedostatky:

- Neexistuje přehrávač pro 64 bitovou verze prohlížeče Internet Explorer.
- Špatná podpora u platforem pro mobilní zařízení.
- Nemožnost označení textu pro zkopírování.
- Náročné animace nejsou vhodné pro méně výkonné počítače.
- Problémy se zvukem u některých zvukových karet.

Microsoft Silverlight

Microsoft Silverlight kombinuje text, vektorovou i bitmapovou grafiku, animace a video. Jedná se o jakousi alternativu této společnosti k technologii Adobe Flash, která navíc přináší podporu čistě streamovaného obsahu. Vzhledem k podobnosti těchto technologií je jeho typické použití stejné jako u Flash — webové animace, atraktivní prezentace, hry a přehrávače.

Existuje také jeho svobodná varianta Moonlight, kterou vyvíjí společnost Novell. Jako programovací jazyky používá Silverlight C#, Visual Basic a IronPython

z platformy Microsoft .NET Framework, ale také třeba JavaScript. Pro vytvoření uživatelského rozhraní je využit deklarativní jazyk XAML založený na XML.

Jako výhody této technologie je možné uvést velký výběr programovacích jazyků, se kterými pracuje a patrně také lepší model animací, který je založen na Windows Presentation Foundation (WPF). I přes to, v porovnání s technologií Flash, je Silverlight méně nasazován pro jeho nižší podporu u uživatelů, především mimo platformu Windows.

Java applet

Existuje několik web appletů pro různé programovací platformy a jejich jazyky. Nejznámějším zástupcem je jazyk Java, který je pro použití ve formě appletu pro internetový prohlížeč velmi vhodný. Jazyk totiž není překládán přímo do vykonatelného strojového kódu, ale pouze do mezikódu. Tento kód následně interpretuje Java Virtual Machine (JVM), který existuje prakticky pro všechny současné platformy. Tím je zajištěna potřebná kompatibilita napříč sítí Internet. Protože byly Java applety představeny již v roce 1995, tak si stihly vybudovat velmi silnou podporu, díky které je často výraz applet chybně interpretován pouze jako Java applet.

V podstatě se jedná o desktopovou aplikaci, která pouze pracuje v okně internetového prohlížeče. Je tak dosaženo funkcionality, kterou by nebylo možné skriptováním na straně klienta nikdy docílit. Takovým pokročilým Java appletem je například JPC – plnohodnotný emulátor PC platformy x86. V okně prohlížeče je tak spuštěn vizualizovaný počítač, ve kterém může pracovat například celý operační systém. Autoři jako příklad uvádí DeLi Linux nebo některé klasické hry pro MS DOS. S tímto přístupem ovšem vznikají velká bezpečnostní rizika, a tak musí uživatel se standardním bezpečnostním nastavením každé spuštěním Java appletu odsouhlasit.

Obecně můžeme tvrdit, že jsou Java applety na ústupu. Pro jejich dřívější typické nasazení (animace, hry nebo pokročilejší uživatelská rozhraní) je možné použít technologie Flash nebo Silverlight, které jsou pro prostředí webu přímo určeny a nevznikají u nich problémy s bezpečností. Stále je ovšem možné najít velmi dobré příklady jejich použití (což patrně JPC není). Oblíbenou službou se například stal Screentoaster, který nabízí nahrávání videa z pracovní plochy operačního systému. Takto nahraná videa se na serveru uloží. Na ty je možné následně odkazovat z jiných stránek nebo je prezentovat na video komunitních webech (např. YouTube).

4.2. Vzhled

Tato kapitola seznámí čtenáře s problematikou standardů uživatelských rozhraní v prostředí webu. Další text je věnován používaným navigačním mechanismům, otázkám problematického vykreslování některých elementů napříč spek-

trem běžných internetových prohlížečů a nakonec také optimalizaci rozhraní pro různá rozlišení zobrazovacích zařízení.

4.2.1. Standardy

Při vytváření desktopové aplikace má programátor zpravidla k dispozici dokument určující konkrétní pravidla (doporučení) pro tvorbu uživatelského rozhraní. Tento dokument se obecně nazývá Human Interface Guidelines (zkráceně HIG) a řeší základní otázky uživatelského rozhraní, jako je vzhled oken, jejich velikost, ovládací prvky včetně menu, práci s klávesnicí a myší, atp. Pravidla nacházející se v tomto dokumentu mají za cíl nejen sjednotit obecné principy uživatelských rozhraní, ale také zvýšit komfort jejich ovládání a podpořit vytváření dobrých zvyků pro práci se softwarem. Nedodržení těchto pravidel je snadné identifikovat. U uživatelských rozhraní desktopových aplikací je tak možné říci co je správně (dle standardů) a co je naopak chyba. Protože rozhraní různých operačních systémů, pro které jsou tyto desktopové aplikace vytvářeny, mají svá vlastní specifika, existuje celá řádka těchto dokumentů. Jako jejich zástupce pro nejpoužívanější platformy můžeme uvést:

- Windows User Experience Interaction Guidelines,
- Apple Human Interface Guidelines,
- GNOME Human Interface Guidelines,
- KDE Human Interface Guidelines.

V polovině 90. let je možné pozorovat snahu o vytvoření podobných doporučení i pro webové stránky. Nezávisle na sobě bylo vydáno několik dokumentů, které se snažily specifikovat pravidla pro vytváření web designu, což můžeme chápat jako uživatelské rozhraní webových stránek. Nejčastěji se jednalo o doporučení velkých počítačových firem, jako například:

- Sun Microsystems: Guide to Web Style (1995),
- Apple: Web Design Guide (1996),
- IBM: Web Design Guidelines (1997).

Za vznikem tolika nezávislých doporučení stojí pravděpodobně masivní a nepříliš organizovaný rozvoj sítě Internet v této době. Možná také vědomí absence konkrétních pravidel uživatelských rozhraní pro toto prostředí a snaha o standardizaci právě jejich dokumentu doporučení. Organizace W3C zabývající se standardy v prostředí sítě Internet nikdy podobný dokument nevydala, ani nepřijala z již dříve vydaných dokumentů.

Z výše uvedeného vyplývá, že programátoři webových aplikací nemají k dispozici alternativu HIG, tak jako programátoři desktopových aplikací. Nejsou tedy řízeni striktními pravidly, jak by měl který prvek vypadat a jak by se měl ovládat. Uživatelské rozhraní webových aplikací má proto mnohem volnější formu.

Webové stránky, z nichž webové aplikace vycházejí, mají navíc zcela jinou filosofii v porovnání s desktopovými aplikacemi. Webové stránky slouží především k prezentaci dat. Všechny technologie (a standardy), které k tomuto účelu na úrovni prezentační vrstvy používají, zdědily i webové aplikace. Na druhé straně desktopové aplikace jsou typicky zaměřeny na vytváření a správu dat, k čemuž mají i nativně přizpůsobené technologie prezentační vrstvy. Je tedy zřejmé, že pro vytvoření funkčního a zároveň intuitivního uživatelského rozhraní webových aplikací je nutné aplikovat standardy z uživatelských rozhraní desktopových aplikací. Ty ovšem není možné slepě následovat právě z důvodů uvedených rozdílných filozofií. Měla by být použita pouze taková doporučení z HIG, která bude možné v rámci vytvářené webové aplikace jednoznačně označit za přínos pro uživatelské rozhraní.

4.2.2. Navigační mechanismy

I když nejsou k dispozici konkrétní oficiální doporučení, tak je vhodné následovat osvědčené principy, které v prostředí webu zdomácněly. Jako klasické zástupce těchto osvědčených principů můžeme označit specifické navigační mechanismy, které se v prostředí webu používají. Tyto navigační struktury slouží pro pohyb v hierarchii jednotlivých webových stránek nebo v případě webových aplikací po blocích nabízených funkcí. Je důležité si uvědomit, že tyto struktury mohou být velmi složité díky původní filozofii hypertextových dokumentů (tedy dokumentů, které jsou navzájem provázány odkazy). Uživatel by si měl být vždy vědom své pozice v této hierarchii. Jak uvádí publikace [2], tak navigační mechanismy na stránce by měly poskytovat tyto informace:

- kde jsem,
- kde jsem byl předtím,
- kam se mohu dostat,
- jak se tam dostanu přímo.

Jako dobrý příklad navigačního mechanismu splňující tento požadavek bývá často uvedena drobečková navigace (breadcrumbs trail), která velmi usnadňuje orientaci uživatele na webových stránkách nebo aplikacích. Jedná se přitom o typický prvek použitý v tomto prostředí. Instone [5] pojmenoval tři základní typy těchto navigací:

Drobečky hierarchie (location breadcrumbs) reprezentují pozici aktuální stránky v celkové hierarchii webu bez ohledu na to, jak se uživatel na tuto stránku dostal.

Drobečky cesty (path breadcrumbs) reprezentují skutečnou cestu, kterou se uživatel na aktuální stránku dostal bez ohledu na to, jaká je pozice této stránky v celkové hierarchii webu.

Drobečky atributů (attribute breadcrumbs) reprezentují postupnou specifikaci atributu zobrazeného výrobku, mezi těmito atributy může být například cena, výkon, záruka, velikost, atp. Tento typ drobečkové navigace je možné pozorovat nejčastěji na internetových obchodech.

Jako další takovýto typický navigační prvek je možné uvést mapy webů. Ty zobrazují ve formě stromu celkovou hierarchii dostupných stránek nebo funkcí, které jsou pro uživatele dostupné. Cesta od kořene k některému uzlu obsahuje drobečky hierarchie stránky nebo funkce, která je tímto uzlem reprezentována. Tyto navigační mapy slouží nejen uživatelům pro zorientování se v celkové struktuře webu, ale je výhodná také pro fulltextové vyhledávací služby (jedna stránka obsahuje odkazy na všechny ostatní podstránky webu).

4.2.3. Různá zobrazení stejného elementu

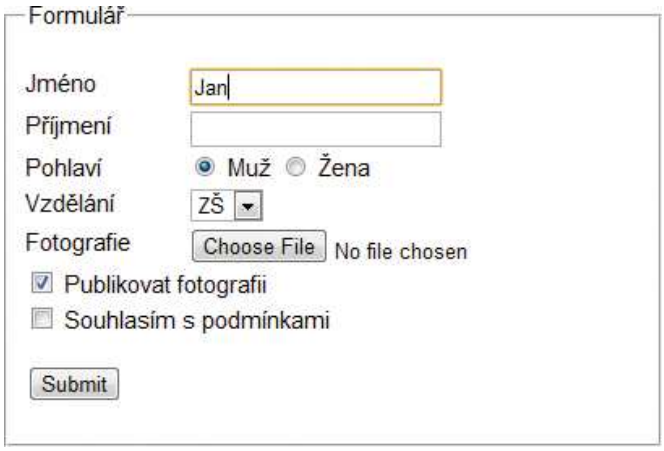
V kapitole 3.1.1. již byla zmíněna problematická interpretace některých vlastností kaskádových stylů CSS různými internetovými prohlížeči. Tento nedostatek nejčastěji v praxi znamená, že budou zobrazení jedné webová aplikace v různých prohlížečích vykazovat odchylky.

Různá zobrazení stejného dokumentu ovšem nemusí zapříčinit pouze nekompatibilita se standardy. Existuje totiž skupina tagů, které zobrazují všechny prohlížeče svým specifickým způsobem, protože W3C ve svých doporučeních přesně nespecifikuje jejich vzhled. Nejmarkantnější jsou tyto rozdíly v zobrazení formulářových prvků a tlačítek. Prohlížeče můžeme rozdělit do dvou skupin podle toho, jak s těmito elementy pracují:

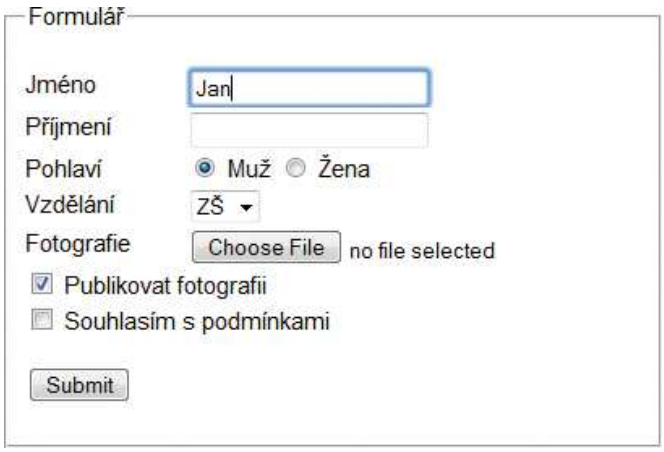
1. Prohlížeč používá své vlastní formulářové prvky, které se zobrazují stejně na všech platformách (např. Opera).

2. Prohlížeč používá jako formulářové prvky nativní ovládací prvky z operačního systému. Vzhled formulářových prvků se tedy mění s použitou platformou (např. Mozilla Firefox).

Je zřejmé, že se jedná o vlastnost, se kterou budou mít autoři uživatelských rozhraní webových aplikací problémy. Ponechat formu zobrazení těchto formulářových prvků na samotném prohlížeči je totiž možné pouze v případech, kdy je



Obrázek 6. Google Chrome 10.0, Windows Vista.

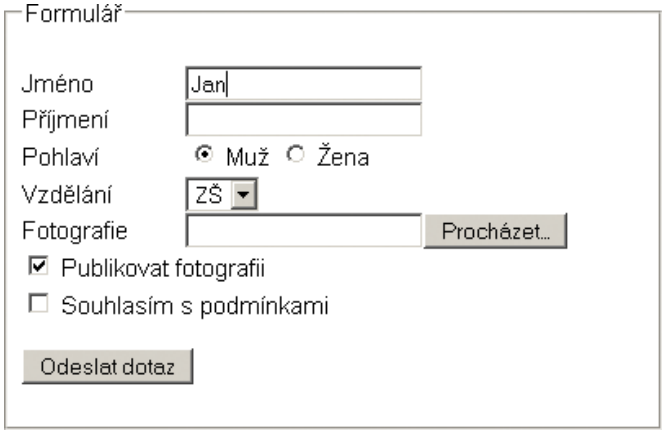


Obrázek 7. Safari 5.0, Windows Vista.

Obrázek 8. Opera 9.5, Windows Vista.

Obrázek 9. Mozilla Firefox 3.0, Ubuntu 8.04 (prostředí GNOME 2.22.2).

Obrázek 10. Mozilla Firefox 3.6, Mac OS X 10.6.



Obrázek 11. Mozilla Firefox 3.5, Windows XP.

vzhled webu velmi jednoduchý. Se stoupajícími nároky uživatelů jsou webové aplikace vytvářeny i s ohledem na jejich atraktivní formu a vzniká tak přirozená snaha sjednotit vzhled vstupů webové aplikace bez ohledu na prohlížeč nebo platformu, kterou návštěvník používá. Za tímto účelem jsou využívány různé techniky pro změnu vzhledu těchto formulářových prvků, aby nepůsobily jako rušivý element. Praktické řešení tohoto problému spočívá v zobrazení zcela nového elementu, který je uměle sestaven z různých tagů a určen k nastylování. Tím je nahrazen původní prvek, jehož zobrazení je potlačeno. Funkce tohoto nového prvku je obstarána JavaScriptem. Toto řešení ovšem není možné aplikovat na všechny formulářové prvky. Nejproblematičtější je v tomto ohledu patrně formulářový vstup typu file, který slouží k výběru souboru na disku uživatele. Bez použití webových appletů ho není možné nahradit jiným elementem, protože pouze za pomoci JavaScriptu nelze jeho funkci simulovat. Dále každý prohlížeč zobrazuje svou vlastní verzi tohoto vstupu, z čehož vyplývá, že aplikace kaskádových stylů je velmi problematická.

Obrázky 6. – 8. na straně 44 zobrazují interpretaci stejného webového dokumentu ve formátu XHMTL 1.0 Strict (nachází se na přiloženém DVD) v různých prohlížečích na platformě Windows Vista. Tento dokument obsahuje jednoduchý formulář se standardními vstupy a odesílacím tlačítkem. První formulářový vstup (jméno) je aktivní. Obrázky 9. – 11. na straně 45 zobrazují opět tento dokument v prohlížeči Mozilla Firefox verze 3 na třech různých platformách.

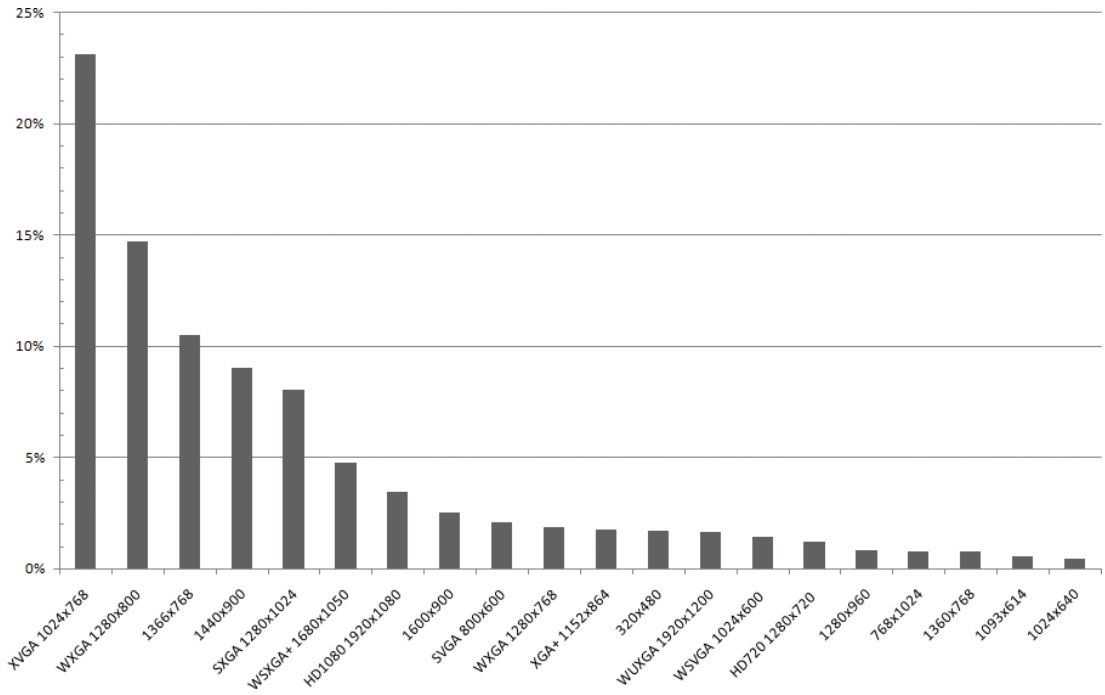
4.2.4. Velikost

Uživatelé přistupují k webovým aplikacím z různých zařízení a každé má jiné zobrazovací možnosti. Může se jednat o drobné dotykové mobilní přístroje, přes displeje klasických notebooků až po velké LCD panely s úhlopříčkou 30" u stolních počítačů. Je zřejmé, že každé z těchto zařízení bude používat jiné nativní rozlišení, nicméně všichni jejich uživatelé jej budou k zobrazení webových aplikací používat stejně.

Protože se v prostředí webových aplikací používá rastrová grafika (vektorová grafika je dostupná pouze u appletů), tak je nutné uživatelské rozhraní optimalizovat, aby bylo bezproblémově zobrazeno u naprosté většiny uživatelů. Tímto bezproblémovým zobrazením je především míněna absence horizontálního posuvníku a zobrazení nejdůležitějších navigačních prvků bez nutnosti vertikálního posunu stránky v okně internetového prohlížeče. Způsoby řešení tohoto problému u uživatelských rozhraní webových aplikací jsou shrnuty v následujícím textu.

Pevná šířka Rozhraní má pevnou šířku a není ovlivněno velikostí okna prohlížeče, přičemž je zarovnáno na střed nebo k levému okraji. Určuje se tak, aby většina uživatelů nebyla nucena používat horizontální posuvník. Jak ukazuje obrázek 12.[8], tak v současné době je nejpoužívanější rozlišení mo-

[8] *zdroj dat: http://screenresolution.org*



Obrázek 12. Zastoupení rozlišení monitoru u uživatelů.

nitoru 1024x768 px. Nižší rozlišení většinou patří přístupům z mobilních zařízení, kterým se zpravidla vytváří zvláštní uživatelské rozhraní. Proto se k určení minimální šířky vychází právě z šířky 1024 px. Od tohoto rozměru je nutné odečíst šířku rámu okna a vertikálního posuvníku. V praxi proto bývá pevná šířka nastavena přibližně na 990 px. Nevýhoda tohoto přístupu spočívá ve vzniku prázdných sloupců vedle uživatelského rozhraní u uživatelů s vysokým rozlišením. Například při rozlišení 2560x1600 px, které používají monitory s úhlopříčkou 30", bude uživatelské rozhraní zobrazeno pouze na 38% z celkové dostupné šířky.

Pohyblivá šířka Rozhraní má pohyblivou šířku, která se přizpůsobí velikosti okna. Protože při použití rastrové grafiky je prakticky nemožné uživatelské rozhraní optimalizovat pro všechna používaná rozlišení (tedy od 1024x768 px do 2560x1600 px), tak se již nepoužívá. Tato technika byla oblíbená především v dřívější době, kdy se obecně používaly menší monitory a rozptyl nejpoužívanějších rozlišení nebyl tak velký. Například v roce 2000 používalo 92% uživatelů sítě Internet rozlišení od 640x480 px do 1024x768 px.

Částečně pohyblivá šířka Jedná se o kombinaci dvou výše uvedených přístupů. Šířka rozhraní je pohyblivá, ale pouze v určeném intervalu hodnot. Minimální šířka je v tomto případě určena stejně jako pevná šířka. Maximální šířka naopak tak, aby pro ni bylo možné uživatelské rozhraní opti-

malizovat bez jakýchkoliv kompromisů. V praxi se tento interval nejčastěji volí od 1024 px do 1440 px.

Všechny uvedené přístupy optimalizace uživatelského rozhraní pro různá rozlišení uživatelů počítají s maximalizovaným oknem internetového prohlížeče. Pokud uživatel z jakýchkoliv důvodů nevyužívá při práci s webovou aplikací plného rozlišení, tak je zřejmé, že nevýhody, které se snaží tyto přístupy potlačit, pro něj nebudou v této chvíli důležité.

S problémem velikosti uživatelského rozhraní také přímo souvisí možnost uživatele změnit velikost písma v nastavení internetového prohlížeče. K tomuto účelu nabízí uživatelům dvě funkce:

Zvětšení textu Zvětšuje se pouze text. Velikost ostatních elementů zůstává nezměněna. Často tak dochází i při menším zvětšení k nevhodnému přetékání a zalamování textu.

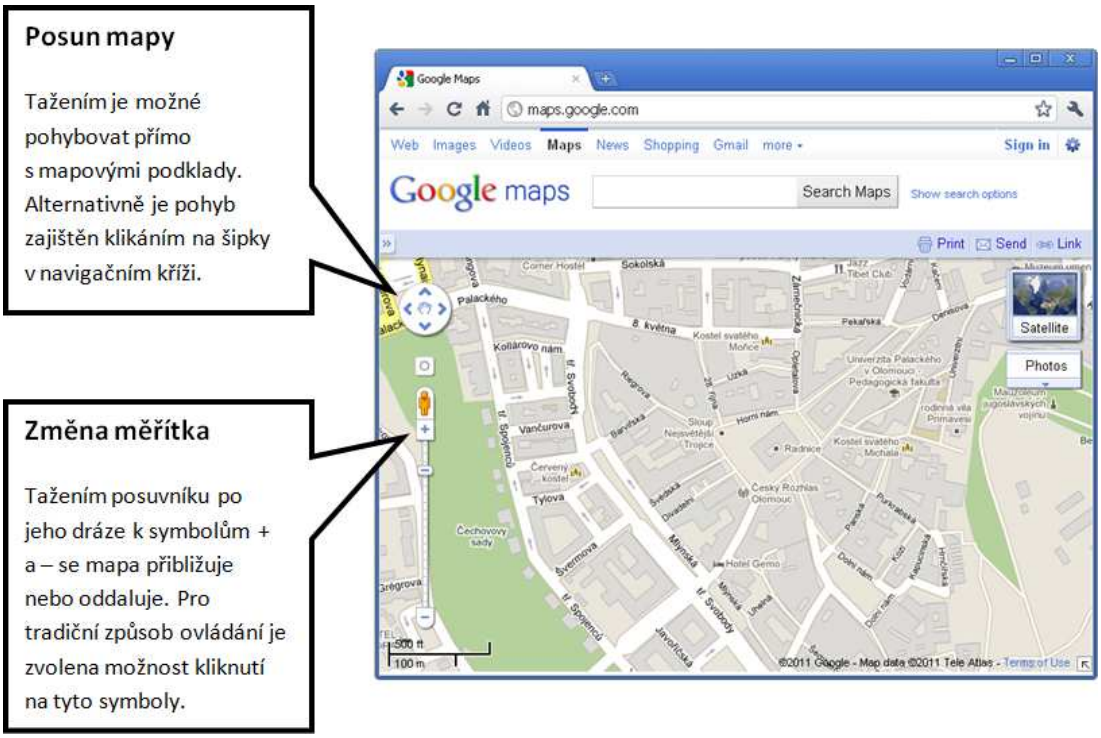
Zvětšení stránky S textem je zvětšována ve stejném poměru i celá stránka, tato funkce se často nazývá lupa. Jedná se o jednoznačně lepší přístup, než pouhé zvětšení textu, nicméně rastrová grafika se při větším přiblížení stává rozmazanou.

Je zřejmé, že ani jeden z těchto přístupů není zcela vhodný, ale požadavek na zvětšení textu je naprosto legitimní a autoři uživatelských rozhraní by jej neměli ignorovat. Aby si mohli uživatelé upravit velikost písma a nedocházelo přitom k popsaným deformacím, tak jsou funkce pro ovládání velikosti textu často nabízeny již v rámci uživatelského rozhraní. Při implementaci na této úrovni je také možné uživatelům nabídnout například změnu kontrastu písma nebo potlačení nepotřebných obrázků pro maximální zvýšení čitelnosti.

Obrázek 13. Ukázka panelu pro změnu velikosti a barvy písma.

4.3. Ovládání

S rostoucí funkcionalitou webových aplikací rostou také nároky na jejich ovládání. Snaží se chováním svého uživatelského rozhraní často simulovat chování desktopových aplikací. Z toho vyplývá, že by měly uživatelům nabídnout i stejný komfort ovládání. Zde ovšem naráží na historický vývoj. Jejich předchůdcem je totiž statický webový dokument, který neobsahoval žádné ovládací prvky. Dovolil uživatelům pouze zadávání vstupů do formulářových prvků nebo pohyb po jeho struktuře pomocí hypertextových odkazů. V tomto bodě vznikají dva fundamentální problémy ovládání webových aplikací.



Obrázek 14. Příklad duálního ovládání uživatelského rozhraní.

1. Současné standardy W3C neberou v potaz principy ovládání desktopových aplikací. Většinu akcí je tak nutné pracně implementovat pomocí JavaScriptu.

2. Uživatelé jsou zvyklí na jednoduché principy ovládání ze statických webů a budou se snažit je promítnout i do ovládání složitých webových aplikací. To je v rozporu se základním požadavkem, který je na ně kladen a to přiblížení jejich ovládání desktopovým aplikacím.

S nedostatkem standardů zmíněných v prvním bodě nezbývá programátorovi nic jiného než se smířit a čekat na vydání jejich nové specifikace případně použít některou z JavaScriptových knihoven pro tvorbu uživatelského rozhraní, která jeho práci usnadní.

Problém uvedený v bodě druhém znamená rozpor v uživateli kladených požadavcích na ovládání. Chce-li autor, aby byla jeho webová aplikace přístupná širokému spektru uživatelů, tak jim musí nabídnout duální způsob ovládání. Jedno je tradiční, které funguje na základě klikání na tlačítka či odkazy a na práci s formulářovými prvky, čili takové, které uznává principy ovládání ze statických webů. Druhé můžeme nazvat moderní nebo pokrokové používající principy, jež byly dříve u webových stránek zapovězeny, nicméně zvyšují komfort jejich užívání.

Vzniká zde podobná situace jako v případě, kdy si uživatel vypne JavaScript a programátor musí alternativně řešit všechno skriptování na straně serveru. Ukázku principu duálního ovládání ukazuje obrázek 14. na příkladu webové aplikace Google Maps.

Následující kapitoly se budou zabývat některými odlišnostmi ovládání rozhraní webových aplikací a jejich specifickými vlastnostmi.

4.3.1. Tažení objektů

Posun nebo změnu velikosti objektu uživatelského rozhraní tažením myši a akci drag-and-drop můžeme jednoznačně označit za nové prvky ovládání webových aplikací přesto, že všechny události potřebné pro jejich implementaci byly součástí již JavaScriptu 1.2 z roku 1997. Opožděný nástup tak můžeme patrně přičíst na vrub problematické podpory internetových prohlížečů (v době vzniku pouze Netscape Navigator).

V současné době všechny majoritní prohlížeče potřebné události podporují. Programátorům jsou k dispozici také knihovny, které nasazení těchto ovládacích prvků usnadňují. Situace se ještě zlepší s příchodem HTML 5, které pro tyto akce myši zavádí standardizované API. Jediným problémem zůstává způsob informování uživatele o možnosti ovládání tak, aby jej mohl přirozeně použít. To se děje zpravidla změnou kurzoru, s čímž jsou spjaty některé potíže uvedené dále.

Posun nebo změna velikosti objektu

Změna velikosti a přesun jsou akce prováděné myší, na které si uživatelé navykli především pro jejich časté používání u oken grafických operačních systémů. Jejich dostupnost se indikuje změnou kurzoru, a to obousměrnými šipkami vyznačující směr pro změnu velikosti nebo navigačním křížem při posunu.

Vzhledem k tomu, že CSS podporuje potřebné ukazatele a současné prohlížeče jsou kompatibilní s vyžadovanými událostmi, tak je možné toto chování zcela identicky simulovat u webových aplikací i při splnění aktuálních W3C standardů.

Drag-and-drop

U desktopových aplikací probíhá akce drag-and-drop zpravidla bez změny kurzoru. Objekt, který je přetahován, se ke kurzoru připojí jako částečně průhledný. Přesně simulovat toto chování není u webových aplikací příliš vhodné ze dvou důvodů:

1. Akce drag-and-drop je u desktopových aplikací uživateli již dlouho přirozeně používána a není ji tak nutné explicitně upozorňovat na možnost provedení této akce. U webových aplikací je situace odlišná, obzvláště s ohledem na fakt, že standardní kurzor při pohybu nad webovým dokumentem slouží k označování textu.

2. Připojení průhledného objektu se rozchází s nynějšími standardy CSS 2.1, protože není podporována vlastnost průhlednosti. Situace se změní až s příchodem CSS 3, nicméně již nyní je možné tuto vlastnost reálně v některých prohlížečích používat (viz tabulka 3.).

Nejvhodnějším kurzorem, pro informování uživatele o dostupné akci drag-and-drop, by patrně byla ruka s rozevřenými prsty (při přejetí nad uchopitelným objektem) a ruka se sevřenými prsty (při tažení). Definování těchto kurzorů ovšem není pomocí samotného CSS možné (a to i ve specifikace CSS 3). Specifikace CSS 2 umožňuje načtení vlastního kurzoru z externího souboru, což je ale vlastnost, která vykazuje problematickou kompatibilitu u internetových prohlížečů. Například prohlížeč Opera, který je v České republice stabilně 5. nejpoužívanější prohlížeč (7% uživatelů)[9], tuto vlastnost nepodporuje vůbec. U tohoto prohlížeče je nutné využít některý ze sady kurzorů, který je možné pomocí CSS alternativně specifikovat. Existuje tedy skupina uživatelů, která nebude o dostupnosti akce drag-and-drop adekvátně informována. Jediným možným řešením je v tomto případě změna internetového prohlížeče.

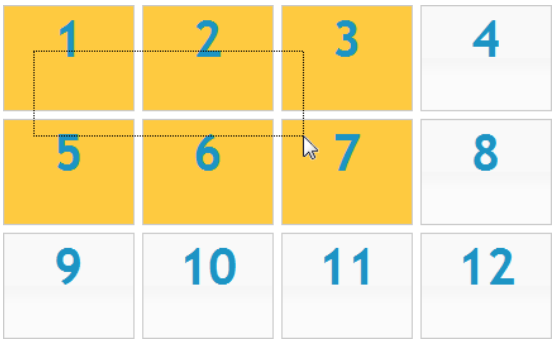
Druhý problém s průhledným objektem u kurzoru se zpravidla řeší jeho připojením bez průhlednosti (jako při přesunu). Tento přístup zavádí i knihovny, které tuto akci implementují. Jedná se o odklon od uživatelského rozhraní desktopových aplikací, nicméně není nijak závažný a většina uživatelů jej ani nepostřehne. Alternativně je možné využít nevalidních vlastností CSS 3 v prohlížečích.

4.3.2. Výběr tažením

Grafická uživatelská rozhraní operačních systémů nabízí uživatelům možnost tažením označit skupinu objektů, se kterými je následně provedena vybraná akce, například mohou být hromadně přesunuty nebo smazány.

Události potřebné k implementaci této akce v prostředí internetového prohlížeče jsou shodné s událostmi potřebnými při tažení objektů uvedených v před-

[9] *zdroj dat: globální statistiky služby TOPlist*



Obrázek 15. Výběr objektů tažením vytvořen knihovnou jQuery UI.

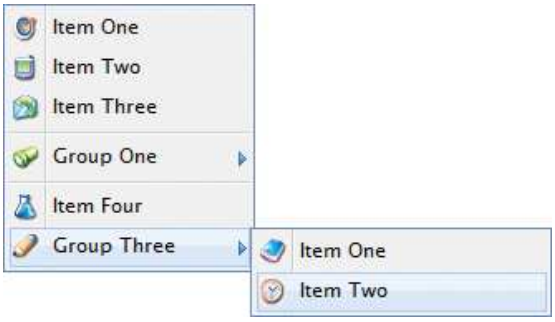
chozí kapitole. Je tedy možné tuto akci nabídnout i uživatelům webových aplikací. Také zde jsou k dispozici knihovny, které již nabízejí hotová řešení, reálné nasazení je programátorům velmi usnadněno.

Problematickou částí tak zůstává opět způsob informování o dostupnosti této možnosti ovládání, protože ji uživatelé nebudou u webových aplikací používat stejně přirozeně, jako v případě rozhraní operačních systémů. Změna kurzoru zde nebude vhodná. Zvláště, když se bude kurzor měnit i nad označovanými objekty. Nezbývá tedy než seznámit uživatele s dostupností formou dokumentace a zachovat výše popsaný princip duálního ovládání.

4.3.3. Kontextové menu

Kontextové menu se běžně používá v grafických uživatelských rozhraních. Zobrazení si uživatel vyžádá zpravidla kliknutím pravého tlačítka myši. Toto menu nabízí možnosti akcí, které jsou spjaté s aktuálním kontextem nebo se stavem aplikace. Tyto akce jsou nejčastěji aplikovány na vybraný objekt nebo skupinu objektů. Protože se jedná o uživatelsky příjemný a často používaný způsob ovládání, tak můžeme uvažovat i o jeho zavedení do prostředí webových aplikací.

Běžné internetové prohlížeče zobrazují svá vlastní kontextová menu nad webovou stránkou. Obsahují například akce pro pohyb v historii, přidání stránky do záložek nebo zobrazení zdrojového kódu. Nasazení vlastního kontextového menu webové aplikace tedy můžeme interpretovat jako jeho výměnu s původním kontextovým menu internetového prohlížeče, k čemuž jsou využity technologie prezentační vrstvy. Tento přístup ovšem může znamenat snížení komfortu ovládání webu pro uživatele, kteří jsou na původní kontextové menu zvyklí a často jej používají. Z tohoto důvodu by měla být tato technika ovládání používána pouze v situacích, kdy je uživateli přirozeně očekávána. Tedy v případě, kdy simulujeme v uživatelském rozhraní takové chování desktopové aplikace, u které je práce s kontextovým menu zcela běžná. Může se například jednat o nabídku akcí nad tahem označenými objekty, správu souboru zobrazeného v adresářovém stromu nebo možnost označení nějakého bodu.



Obrázek 16. Kontextové menu vytvořené knihovnou wdContextMenu pro jQuery.

Moderní internetové prohlížeče kontextová menu vesměs podporují. Problematická je pouze Opera, kde není podporována událost oncontextmenu, jenž se volá v momentě, kdy si uživatel kontextové menu vyžádá a je tedy klíčová pro potlačení jeho zobrazení. Jsou tak přes sebe zobrazována dvě kontextová menu – webové aplikace a internetového prohlížeče. V tomto prohlížeči je proto diskutovaná technika ovládání zcela nefunkční.

4.3.4. Scrollování vs. stránkování obsahu

Charakteristický prvek uživatelského rozhraní webových aplikací je stránkování záznamů v případě, že jejich počet přesáhne přijatelnou mez pro zobrazení v celku. Uživatel má k dispozici ovládací prvky pro pohyb mezi těmito stránkami a často i možnost změny způsobu řazení a úpravu počtu záznamů na jedné stránce. Tento koncept zobrazení je odlišný od přístupu desktopových aplikací, kde je pro pohyb mezi záznamy využíváno vertikální scrollování. Stránkování obsahu se u těchto aplikací používá pouze v případech, kdy ho není technicky možné bez rozdělení zobrazit. Může se jednat například o knihovní informační systém, u kterého je počet zobrazovaných knih v řádech statisíců atp.

Hlavním důvodem pro zavedení tohoto přístupu bylo zrychlení načítání dokumentu, protože není nutné stahovat všechny odpovídající záznamy, ale pouze jejich část neboli stránku, což platilo především v dobách, kdy bylo pro přístup k Internetu nejčastěji využíváno klasické vytáčené analogové připojení. Kvůli jeho nízké rychlosti se každé zvětšení velikosti přenášeného dokumentu přímo promítlo do prodloužení odezvy na uživatelův požadavek. V současné době, kdy je široce používáno vysokorychlostního připojení k síti Internet, ztrácí tento argument na síle.

Druhým důvodem byl přístup grafiků, kteří vytvářeli design webů. Dřívější literatura, která se zabývala webdesignem, totiž často nutnost posouvání kritizovala a bylo doporučováno vytvářet *nízké* stránky, u kterých není potřeba používat vertikální posuvník, jehož ovládání se považovalo za nekomfortní. Tento stav byl dlouhodobě neudržitelný a tak se právě kvůli posunu internetových stránek velmi rozšířily myši, které mají kolečko nebo tlačítka pro ovládání vertikálního posuvníku. V současné době jsou již takto vybaveny prakticky všechny myši na trhu. Alternativní ovládání vertikálních posuvníků ale nabízí také většina touchpadů u přenosných počítačů. Posouvání stránek je tedy v dnešní době bezproblémové a proto můžeme argumenty proti jejich používání považovat za neplatné.

I když výše uvedené argumenty není možné v dnešní době aplikovat, tak princip stránkování záznamů v prostředí Internetu zcela zdomácněl a je nadále využíván. Jediným důvodem, který je stále platný, je snížení zátěže na straně serveru, typicky potom databázového systému, pokud je pro vygenerování zobrazovaných záznamů potřeba složitých databázových dotazů. Ten se překrývá s důvodem nasazení stránkování u desktopových aplikací.

Stránkování často doplňuje dynamická filtrace podle parametrů záznamů zalo-

Obrázek 17. Stránkování nalezených záznamů vyhledávačem Seznam.cz.

žená na technologii AJAX. Jako příklad může sloužit procházení některé kategorie zboží v internetovém obchodě, kde si uživatel vybere pouze produkty, které spadají do definovaného cenového intervalu a mají požadované specifikace. Listování se tak stává efektivnější – sníží se počet stránek a uživateli jsou zobrazeny pouze produkty, o které má zájem.

I přesto, že je stránkování považováno za jakýsi obecný standard v prostředí webových aplikací, při zobrazení většího počtu záznamů, je možné se poslední dobou setkat s přístupy, které využívají pro posun mezi záznamy pouze posuvník nebo jeho kombinaci se stránkováním. Obecně lze rozlišit tři základní přístupy založené na práci s posuvníkem, kterým se věnuje následující text.

Scrollovací blok

V případě menšího počtu záznamů je možné využít pro jejich zobrazení scrollovací blok. Všechny záznamy jsou načteny do tohoto bloku a pro pohyb mezi nimi slouží přidaný vertikální posuvník.

Toto řešení přináší problém s dvojím vertikálním posuvníkem, který se pak na stránce nachází. Jeden patří internetovému prohlížeči a posouvá celou stránku, druhý pak vloženému bloku a ovládá pouze posun mezi záznamy. Jinými slovy tedy vzniká scrollovatelný obsah ve scrollovatelném obsahu. Tato situace může být pro uživatele matoucí a navíc vykazuje v některých případech (v závislosti na konkrétním hardware a jeho ovladačích) problematické ovládání kolečkem myši. I přes to je scrollovací blok často využíván zejména pro jednoduché nasazení a jeho konstantní výšku.

Dynamické prodloužení obsahu

Dynamické prodloužení obsahu je možné použít pouze v případě, že jsou zobrazované záznamy seřazeny a záznamy na prvních pozicích mají nejvyšší váhu, zatímco záznamy na posledních pozicích váhu nejnižší. Může se jednat například o chronologické seřazení zpráv, kdy zprávy na prvních místech jsou nejnovější a tedy aktuální zprávy s nejvyšší váhou, zatímco na jejím konci zprávy nejstarší, neaktuální s nejnižší váhou.

Uživateli je na začátku zobrazeno několik záznamů, mezi kterými se pohybuje posunem celé stránky. V případě, že se blíží k nejnižší pozici na stránce, tak je ke konci zobrazovaných záznamů dynamicky připojena další skupina záznamů. Tato připojená skupina začíná záznamem, který je v pořadí, jenž navazuje na původní poslední zobrazený záznam před připojením.

Dynamické načtení obsahu podle pozice

Je-li u záznamů předpoklad, že mezi nimi bude uživatel intenzivněji listovat, například při hledání nějakého konkrétního záznamu pouze podle rychlé vizuální kontroly, tak je výhodné klasické stránkování doplnit i o dynamické načítání záznamů na základě pozice stránky, jenž je také založené na technologii AJAX.

Princip tohoto způsobu procházení záznamů spočívá v jejich rozdělení na velké stránky tak, aby bylo možné souvislé zobrazení vysokého počtu záznamů. Počet záznamů na jedné takové stránce může odpovídat deseti až stonásobku záznamů na stránce při využití klasického procházení stránkováním. Aby se nemuselo čekat, než se celá tato stránka načte a také se snížila režie serveru, je uživateli na začátku načteno pouze několik prvních záznamů (jejich počet může odpovídat například jedné původní stránce). Všechny ostatní záznamy obsažené na stránce se prozatím nenačítají a čeká se na pohyb uživatele posuvníkem stránky. Jakmile uživatel započne vertikální posun, tak jsou chybějící záznamy, které se při posunu zobrazí, dodatečně načteny a uživateli zobrazeny.

Jedná se o uživatelsky příjemný způsob procházení záznamů v případě, že je dostupné rychlé připojení k síti Internet. Tento přístup používá například služba Google při procházení výsledků hledání obrázků.

4.3.5. Ovládání klávesnicí

Za software, který je ovládán výhradně klávesnicí, se v dnešní době dají považovat pouze konzolové aplikace. Grafická uživatelská rozhraní klasických desktopových aplikací bývají primárně určena pro ovládání myší. Klávesnice zde slouží jen jako alternativa k myši, často i neplnohodnotná, pouze pro zadávání zkratek.

Podobná situace panuje i u webových aplikací. Uživatelská rozhraní webových aplikací jsou nativně určena pro ovládání myší a klávesnice se nejčastěji používá pouze pro zadání krátkých vstupů do formulářových prvků. Je ovšem možné zvýšit uživatelský komfort přidáním základní podpory klávesnice a klávesnicových zkratek, na které jsou uživatelé zvyklí z desktopových aplikací. Může se jednat například o používání šipek pro pohyb v rozbalovací stromové struktuře, skupinový výběr objektů za pomocí držení klávesy Ctrl a klikání myši nebo přejmenování označené položky stiskem F2. Jako primární ovládací prvek bude ovšem stále sloužit myš.

Vytváření webových aplikací výhradně nebo především pro klávesnici v případech, které tento přístup nevyhnutelně nevyžadují, by znamenalo ignorování současného trendu, kdy je klávesnice jako hardwarový prvek na ústupu. Důsledkem by patrně byla pro uživatele nepoužitelná aplikace z důvodu jejího exotického ovládání. Jako důkaz uvedeného trendu může posloužit zvýšený prodej tabletů a dotykových mobilních zařízení, které jsou určeny především pro práci s Internetem a multimediálním obsahem. Tato zařízení zcela postrádají hardwarovou klávesnici a ovládají se pouze dotykovou vrstvou na svém displeji. Uživatel zde poklepem a pohybem prstu simuluje kliknutí a tažení myši.

Důvodem pro postavení klávesnice pouze jako alternativy k ovládání myší by také měla být špatná podpora událostí onKeyDown, onKeyPress a onKeyUp. Ačkoliv jsou součástí již JavaScriptu 1.2 z roku 1997, tak se nesprávně interpretují i v současných internetových prohlížečích. Nejproblematičtější je zde stisk speciálních kláves jako jsou Enter, Tab, mezerník, atp.

5. Modelová webová aplikace – e-mailový klient

Jako modelová aplikace pro potřeby této práce byl implementován e-mailový klient v jazyce PHP. Důraz byl kladen především na kvalitu uživatelského rozhraní a celkové zabezpečení aplikace. E-mailový klient byl vybrán právě proto, že ekvivalentní webové aplikace (například SquirrelMail, V-webmail, IMP, NOCC, atd.) v těchto kritériích pokulhávají.

Tato webová aplikace je významným, ale ne hlavním, výstupem této diplomové práce. Bude sloužit pouze pro podporu srovnání webových a desktopových aplikací a jako doklad jejich konkurenceschopnosti. Následující text se proto zaměřuje především na krátké uvedení aplikace, provedení srovnání a představení nestandardních funkcí, jako je především vysoké zabezpečení. Z těchto důvodů není uveden detailní programátorský rozbor a uživatelská příručka.

5.1. Použité technologie, knihovny a jejich výběr

Následující kapitoly představují použité technologie a jejich knihovny rozdělené do vrstev při vývoji aplikace. Uvádí důvody, proč byly vybrány a také důsledky, které jejich použití přináší.

5.1.1. Prezentační vrstva

Aplikace je určena pro široký okruh uživatelů, proto byly zvoleny pro implementaci prezentační vrstvy standardní technologie, u kterých je možné očekávat širokou podporu u uživatelů resp. u internetových prohlížečů, které používají.

XHTML, CSS, JavaScript

Aplikace neobsahuje žádné webové applety. Jejich nasazení nemá žádné opodstatnění, proto je pro tvorbu uživatelského rozhraní použita klasická dvojice značkovacího jazyka XHTML s kaskádovými styly CSS. Dokument je konkrétně ve formátu XHTML 1.0 Strict, k jeho nastylování je použito CSS ve specifikaci 2.1. Interaktivitu takto vytvořeného dokumentu zajišťuje JavaScript, který je pro tyto účely přímo určen.

Můžeme tedy tvrdit, že uživatelské rozhraní této webové aplikace je vytvořeno za použití běžných technologií, které odpovídají standardům W3C.

jQuery

Pro usnadnění implementace uživatelského rozhraní je použit JavaScriptový framework jQuery. Základní myšlenka tohoto frameworku je oddělení skriptů od samotného XHTML, jedná se o tzv. princip nevtíravého JavaScriptu. Napřed je nalezen specifikovaný prvek v DOM a s ním je následně provedena příslušná akce

(změněn manipulátor událostí, upraveno CSS, atp.). Zvyšuje tak přehlednost a čistotu kódu.

jQuery přináší nativní podporu technologie AJAX včetně JSON formátu pro výměnu dat, zjednodušuje práci s DOM a CSS díky propracovaným selektorům a zavádí také podporu jednoduchých animací, založených na manipulaci s CSS vlastnostmi. Jeho dalším nezanedbatelným přínosem je vyřešení kompatibility interpretů JavaScriptu napříč spektrem internetových prohlížečů.

K jQuery je připojeno několik dalších knihoven, především jQuery User Interface (UI), což je oficiální knihovna uživatelského rozhraní pro jQuery. Nabízí nové možnosti interakce uživatelů a ovládací prvky pro vytváření uživatelsky atraktivních webů.

5.1.2. Aplikační vrstva

Technologie aplikační vrstvy je závislá na serveru, na kterém je aplikace nainstalována. Budeme-li předpokládat, že uživatel nemá k těmto serverům administrátorská práva, tedy nemůže zajistit fungování libovolných technologií, tak je nutné volit u aplikační vrstvy takovou technologii, která je běžně a široce podporována.

PHP

V současné době je naprostá většina webových aplikací instalována na servery různých hostingových služeb, na nichž sdílí jejich výkon spolu s ostatními aplikacemi. Situace, kdy má webová aplikace vlastní server je ekonomicky náročná a vyskytuje se proto velmi zřídka ve specifických situacích. Zejména v případech, kdy přidělený výkon sdíleného serveru není dostačující. Jako standard je u hostingových serverů nejčastěji nainstalován softwarový HTTP server Apache s interpretem jazyka PHP. A právě tyto důvody vedly při výběru technologie aplikační vrstvy k jazyku PHP.

Ačkoliv PHP samo o sobě představuje velice silný nástroj, tak je konfigurací interpretu jeho funkcionalita často administrátorem omezena. Různé konfigurace proto znamenají různá prostředí běhu aplikace, s čímž je nutné při vývoji aplikace počítat. Vznikají tak komplikace, které jsou cenou za bezproblémové zprovoznění aplikace v praxi. Mezi dvě největší omezení aplikace plynoucí z použití této technologie můžeme uvést:

Omezená velikost přílohy Kódování a dekódování souborů s přílohou je paměťově nejnáročnější část aplikace a to i přes to, že je použit nejefektivnější dostupný šifrovací algoritmus Twofish. Velikost přidělené operační paměti je součástí konfigurace PHP interpretu a tedy plně v rukou administrátora serveru. Důsledkem je, že se toto nastavení přímo promítne do limitu pro maximální velikost souboru přílohy. Přílohy, které tento limit překročí, potom není možné přijmout.

Problematické stahování větších dat PHP obsahuje nástroje pro spouštění externích programů nebo zavádění nových procesů. Použití takovéto komponenty na pozadí interpretu by bylo vhodné především pro stažení zpráv z poštovních účtů uživatelů. To není ovšem možné, protože jsou tyto potřebné nástroje administrátorem v interpretu zakázány, zejména z bezpečnostních důvodů a také kvůli omezenému výkonu serveru. Veškeré stahování pošty je tedy nutné provádět pouze při běhu PHP skriptu. Maximální čas běhu skriptu ovšem bývá striktně určen administrátorem. Pokud není veškerá pošta během tohoto časového limitu stažena, je nutné skript ukončit a následně znovu zavolat, spojit se opět s poštovním serverem a dále pokračovat ve stahování od zprávy, u které byl skript naposledy přerušen. Tato smyčka je v aplikaci obsluhována technologií AJAX. Stahování velkého počtu zpráv je tedy zdlouhavé.

Smarty

Smarty je šablonovací systém pro PHP, který slouží k oddělení aplikační logiky od prezentace. Princip jeho funkce je podobný jako u jiných takovýchto systémů. Aplikační logika provede potřebné výpočty, přičemž data, která budou složit k zobrazení uživateli, se uloží do speciálních proměnných a určí se šablona, do níž se mají tyto proměnné vložit. Šablona je v samostatném souboru a obsahuje pouze kód, který má být v prezentační vrstvě zobrazen. Typicky se tedy jedná o značky jazyka XHMTL. Do něj jsou vložena výstupní data předaná z aplikační logiky, přičemž mohou být na této úrovni ještě upravena. Tyto úpravy mají pouze charakter optimalizace výstupu pro jeho grafické zobrazení, tedy například omezení délky řetězců, převod speciálních HTML znaků na entity a podobně. K těmto účelům používá Smarty jazyk s vlastní syntaxí a sémantikou, čímž je také velmi podpořeno oddělení aplikační logiky od prezentace, protože nedochází k míchání XHTML a PHP kódů. Šablony jsou při prvním spuštění kompilovány tak, aby při každém dalším spuštění skriptu, jenž volá tuto šablonu, nemusely být znovu překládány značky Smarty na proveditelný kód. Po této kompilaci je vytvořen soubor, který spojí kód šablony s aplikační logikou a to tak, že provede kritizované smíchání XHTML a PHP kódů. To vše se ovšem děje na úrovni, do které programátor nevstupuje. Takto je vygenerován efektivní a rychlý kód a programátor dodržuje klasické principy oddělení jednotlivých vrstev aplikace.

5.1.3. Datová vrstva

Pro konzistentní uchování uživatelských dat používá aplikace databázový i souborový systém. Ve formě souborů jsou na souborovém systému uchovávány přílohy zpráv. Všechna ostatní data se ukládají do relační databáze databázového systému.

MySQL

Protože je aplikační logika implementována jazykem PHP, tak byl jako databázový systém přirozeně vybrán MySQL, který tento jazyk tradičně doplňuje. MySQL je databázový systém zaměřený především na rychlost za cenu drobných zjednodušení. I přesto nepostrádá žádnou důležitou vlastnost relačních databází a tak jeho výběr neznamená v aplikaci žádná reálná omezení. Naopak výhodou MySQL je nativní podpora šifrovacího algoritmu AES, což usnadňuje implementaci zabezpečení dat popsanou v kapitole 5.5. Pro podobnou funkcionalitu by například alternativní databázový systém PostgreSQL, jehož podpora je u hostingových služeb také velká, musel mít nainstalovaný přídavný balíček kryptografických funkcí, což by pro uživatele bez administrátorských práv mohlo znamenat neřešitelný problém.

5.2. Funkcionalita

Modelová aplikace poštovního klienta se snaží nabídnout běžnou funkcionalitu potřebnou pro pohodlnou práci se zprávami. Vyniká nadstandardním zabezpečením a přístupem ke správě poštovních účtů. Podle dostupných zdrojů se totiž jedná o jediného webového e-mailového klienta, který se instaluje na server a nabízí možnost stahování zpráv z libovolných poštovních účtů, které si definují sami uživatelé ve svém rozhraní. Webové e-mailové klienty, jako například známý SquirrelMail, běžně umí spolupracovat pouze s jedním účtem, který nemůže uživatel bez administrátorských práv nijak měnit. Modelová aplikace takto zavádí stejný přístup k práci s poštovními účty jako u klasických desktopových aplikací tohoto charakteru (například Microsoft Outlook). Obecně v porovnání s desktopovými aplikacemi jsou možnosti nabízených funkcí modelové aplikace limitovány pouze jejím přizpůsobením pro většinu PHP serverů a samozřejmě také omezenou dobou vývoje. V případě garance neomezující konfigurace PHP interpretu již při vývoji by bylo technicky možné implementovat jakoukoliv funkční vlastnost desktopových aplikací. Následující text kapitoly se věnuje krátkému popisu základních funkcí aplikace.

Poštovní účty

Každý uživatel si spravuje sám poštovní účty, ze kterých jsou zprávy přijímány a může jich v aplikaci definovat libovolný počet. Zprávy z poštovních účtů jsou stahovány vždy do lokálních složek aplikace. Stahování je možné provádět protokoly POP3 i IMAP, přičemž je podporováno i zabezpečené SSL připojení. Dále má také uživatel možnost ponechat kopii přijaté zprávy na poštovním serveru tak, aby mohla být stažena také jiným poštovním klientem.

Pro odesílání zpráv je využíván SMTP protokol a také zde je možné využít SSL. Nemá-li uživatel v rámci svého poštovního účtu přístup k tomuto protokolu, tak může předat odeslání zprávy lokálnímu serveru, na kterém je aplikace nainstalována.

Pravidla pro příjem pošty

Pravidla pro příjem pošty se typicky používají pro efektivní automatické třídění přijatých zpráv do složek s poštou. Skládají se z podmínek, které se vyhodnocují u každé přijaté zprávy. Tyto podmínky určuje pro každé pravidlo sám uživatel. Na jejich základě je možné kontrolovat například, jestli je zpráva přijatá z vybraného účtu, zda má přílohou nebo její předmět obsahuje určitá slova. Pokud přijatá zpráva odpovídá všem určeným podmínkám v pravidlu, tak se s ní provede jedna z následujících akcí, kterou opět vybere uživatel:

- pošta se přesune do libovolné složky,
- pošta se nebude ze serveru stahovat a bude na něm ponechána,
- pošta se nebude ze serveru stahovat a smaže se z něj.

Kontakty

Aplikace obsahuje, tak jako většina e-mailových klientů, správu kontaktů. Kontakt je dvojice jméno a adresa, kde adresa musí být unikátní a jméno je nepovinné. Kontakty jsou přidávány automaticky z hlaviček přijatých zpráv. Je také k dispozici jejich správa, kde si může uživatel kontakty ručně přidávat, mazat nebo editovat. Při vyplňování adres v nové zprávě jsou uživateli uložené kontakty našeptávány na základě zadaných prvních znaků jména nebo adresy.

Koncepty

Rozepsané zprávy se ukládají do složky konceptů. Ty je následně možné zobrazit, dopsat a odeslat. Uložení se provádí automaticky, pokud uživatel rozepsanou zprávu opustí. Uložení zprávy je také možné provést ručně, kliknutím na tlačítko Uložit.

Odstraněná pošta

Aplikace využívá princip koše pro smazaná data, který používají i jiné e-mailové klienty. Pokud uživatel smaže zprávu, tak je ve skutečnosti pouze přesunuta do složky s názvem Odstraněná pošta, jenž v tomto kontextu představuje koš. Tento přesun je proveden bez potvrzení uživatele. Trvale může uživatel odstranit zprávy tak, že je smaže z této složky nebo tuto složku vyprázdní. Pro provedení této akce je již vyžadováno potvrzení uživatele, zobrazením dialogového modálního okna s otázkou. Tento přístup, který je známý i z operačních systémů, minimalizuje možnost nechtěného smazání zprávy. Díky možnosti hromadného smazání všech zpráv ve složce Odstraněná pošta je navíc udržen vysoký uživatelský komfort.

Správa složek pošty

Uživatel má po instalaci k dispozici čtyři systémové složky, které jsou pevné, a není možné s nimi manipulovat. Jsou to složky Doručená pošta, Odeslaná pošta, Odstraněná pošta a Koncepty. Pro přehledné roztřídění všech zpráv (doručených, přijatých i rozepsaných) může uživatel vytvořit vlastní strukturu složek, které mohou být jakkoliv vnořené a nést libovolné jméno do 25 znaků. Počet takto uživatelsky vytvořených složek není nijak omezen a je možné je vložit i do systémových složek. Jednotlivé zprávy se pak do těchto složek přesouvají. To je proveditelné buďto ručně nebo za pomocí pravidel pro příjem pošty.

Uživatelská nastavení

V rámci nastavení účtu může uživatel změnit formát data a času, upravit zobrazování informativních hlášek (v aplikaci nazvaných bubliny) nebo nastavit délku intervalu automatického příjmu pošty.

Uživatelské nastavení umožňuje také změnu přístupového hesla. S ohledem na vysoké bezpečnostní standardy aplikace je po uživateli vyžadováno, aby bylo nové heslo dostatečně silné. O síle svého nového hesla je informován ihned, při psaní formou progress baru, který se se sílou hesla nejen prodlužuje, ale také postupně mění barvy od červené (slabé heslo) až po zelenou (velmi silné heslo).

Administrace

Administrace se zpřístupní přihlášeným uživatelům, kteří mají administrátorská privilegia. Základním právem administrátora je vytvářet, upravovat a rušit přihlašovací účty. Každému uživateli také určuje maximální diskový prostor, který může využít. Administrátor by měl zajistit, že takto garantovaný prostor bude na serveru uživatelům skutečně vyhrazen.

Dále provádí administrátor nastavení, která jsou určená konfigurací PHP interpretu. Konkrétně se jedná o maximální velikost jednoho souboru a maximální celkovou velikost souborů v příloze a také určení, zde se má vytvářet nová relace připojení k serveru příchozí pošty po stažení zprávy s přílohou.

5.3. Uživatelské rozhraní

Uživatelské rozhraní modelové aplikace bylo vytvářeno s důrazem na celkový komfort ovládání, který by měl být srovnatelný s rozhraním ekvivalentní desktopové aplikace. Toho se snaží docílit třemi různými způsoby – snížením doby odezvy, přehledným rozložení zobrazených informací a zavedením nových ovládacích prvků s intuitivním ovládáním.

5.3.1. Rychlost

Po tom, co se uživatel úspěšně přihlásí, je načten dokument, který obsahuje rozhraní aplikace. Adresa tohoto dokumentu se po celou dobu práce nemění. Pokud uživatel v rozhraní vyžádá zobrazení jiného obsahu, tak se modifikuje pouze část URL obsahující identifikátor pozice stránky a nový obsah je do aplikace načten dynamicky s využitím technologie AJAX. Oproti klasickému přístupu, kdy je s každou změnou zobrazovaného obsahu sestaven nový dokument, který je odeslán do prohlížeče, přináší toto dynamické načítání dvě velké výhody:

1. Snižuje se objem přenášených dat mezi serverem a klientem, čímž se zkrátí doba odezvy.
2. Zobrazení celého nového dokumentu trvá internetovému prohlížeči podstatně déle než pouze provedení změny části obsahu původního dokumentu.

Důsledkem je celkové zrychlení aplikace. Jistá prodleva mezi zadáním požadavku a vykreslením výsledku stále existuje, nicméně je minimalizovaná na nezbytné minimum. Pokud uživatel, který k aplikaci přistupuje, využívá připojení k síti Internet s rychlostí stahování dat nad 8 Mbit/s (například ADSL), tak bude běžná doba odezvy řádově v desetinách vteřiny. Dobu jedné vteřiny odezva překročí jen výjimečně a to zejména v případech, kdy server odesílá delší odpověď. Může se ale také jednat o zahlcení aplikace AJAX požadavky, tedy pokud je zpracováváno několik takovýchto požadavků současně. Tuto situaci je možné v aplikaci pozorovat, například pokud si při stahování zpráv z poštovního účtu uživatel vyžádá zobrazení jiného obsahu. Takto vzniknou paralelně dva asynchronní AJAX požadavky a zobrazení vyžádaného obsahu potom trvá citelně déle.

Rychlost aplikace je také podpořena zavedením ovládacích prvků, které pro vykonání své funkce využívají pouze skriptování na straně klienta a zcela vynechávají komunikaci se serverem. Jejich doba odezvy je tedy srovnatelná s odezvou desktopových aplikací. Tyto prvky jsou použity pro ovládání často používaných funkcí, jako je:

- pohyb mezi jednotlivými stránkami tabulky zpráv,
- vyhledávání ve složce zpráv,
- změna způsobu řazení v tabulce zpráv,
- pohyb mezi záložkami stránky.

5.3.2. Rozložení

Vzhledem k tomu, že elektronická komunikace patří k běžným standardům dnešní doby, tak většina uživatelů osobních počítačů pravidelně používá různé

e-mailové klienty. Vynecháme-li specifické aplikace, například určené pro mobilní zařízení, tak můžeme říci, že jejich grafická uživatelská rozhraní jsou podobná a postavená na stejných principech. Může se jednat například o zobrazení zpráv ve formě tabulek s možností řazení dle jednotlivých sloupců nebo systém složek, do kterých jsou zprávy vkládány. Modelová aplikace proto nepřichází s žádnými překvapivými prvky uživatelského rozhraní, ale právě naopak se v něm snaží implementovat tyto obecně zažité principy. Díky tomu budou moci běžní uživatelé aplikaci ovládat intuitivně, i bez nahlédnutí do nápovědy.

Samotné rozhraní je rozděleno do dvou základních bloků – navigačního sloupce a rámce hlavního obsahu. Navigační sloupec se skládá z několika samostatných částí. Je v něm zobrazena struktura složek pro zprávy s nástroji pro jejich editaci, informaci o využitém diskovém prostoru a menu s položkami pro různá nastavení a odhlášení. Uživatelem vyžádaný obsah se zobrazuje v hlavním rámci, který se nachází vedle navigačního sloupce. Nad tím je umístěno ovládání dvou nejdůležitějších funkcí e-mailového klienta a to vytvoření nové zprávy k odeslání a příjmu zprávy z poštovního účtu, přičemž je možné přijmout zprávy ze všech účtů nebo pouze z jednoho vybraného. Každá samostatná část uživatelského rozhraní se zobrazuje ve vlastním grafickém rámci, který je jasně odděluje. Je tak zachována celkově vysoká přehlednost, která je navíc podpořena použitím ikon a jiných vizuálních prvků, namísto textu ve vhodných případech, jako je například:

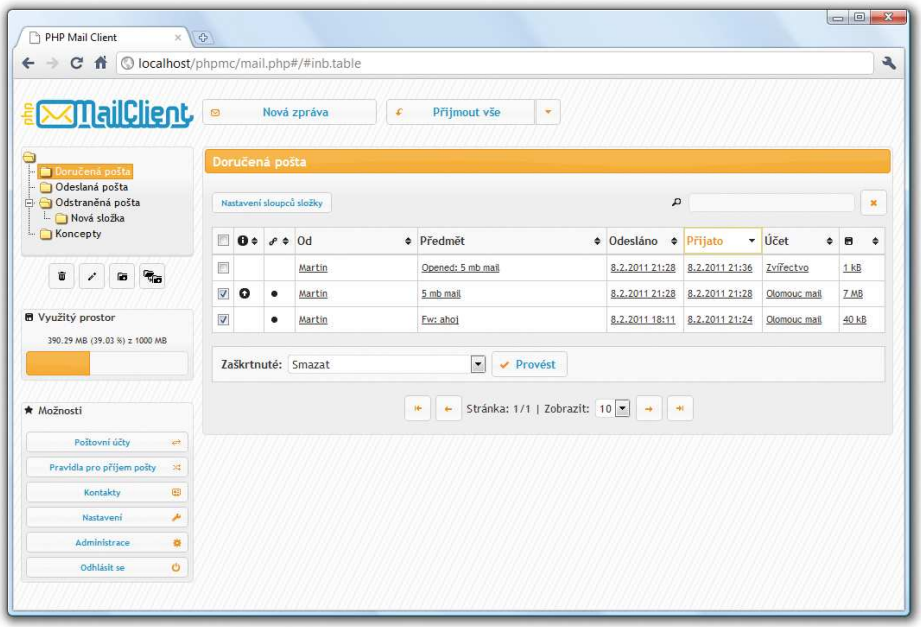
- zobrazení priority zprávy,
- ikony pro editaci struktury složek se zprávami,
- progress bar pro využitý prostor, atp.

Pokud je namísto textu zobrazena pouze ikona, tak je vždy doplněna o nápovědu, která vysvětluje její význam. Tyto nápovědy mají formu tzv. tooltipů, které se zobrazují po přejetí kurzoru nad ikonou. K implementaci těchto nápověd byla použita knihovna TipTip pro jQuery.

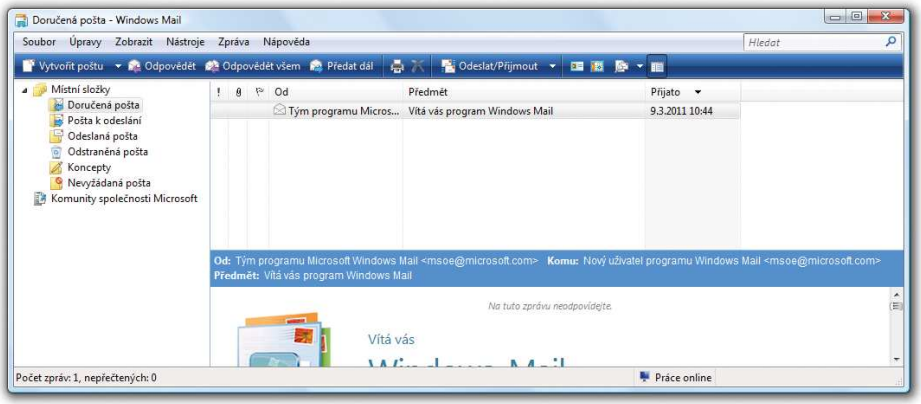
Na straně 65 je možné porovnat rozhraní modelové aplikace s rozhraním desktopového e-mailového klienta Windows Mail, který je standardní součástí operačního systému Windows Vista a tudíž i velmi často používaný. Na tomto příkladu je zřejmé podobné rozložení i použití stejných obecně zažitých principů ovládání e-mailových klientů popsaných výše.

5.3.3. Ovládací prvky

V aplikaci je obsaženo několik zajímavých ovládacích prvků, které nejsou u webových aplikací běžné a jejich použití je typické především u desktopových aplikací. U těchto prvků je dodržen princip duálního ovládání diskutovaný v kapitole 4.3. Existuje tedy vždy alternativní ovládání založené na způsobech,



Obrázek 18. Uživatelské rozhraní webového e-mailového klienta.



Obrázek 19. Uživatelské rozhraní desktopového e-mailového klienta.

kterými se ovládají statické weby, aby bylo zachováno snadné ovládání i pro konzervativní uživatele.

Následující text uvádí konkrétní příklady použití moderních ovládacích prvků v modelové aplikaci.

Drag-and-drop

Seznam zpráv obsažených v otevřené složce se zobrazuje ve formě uživatelsky definovatelné tabulky. Přesun zprávy do jiné složky patří k velmi často používaným funkcím, především pokud uživatel třídí zprávy ručně. Pro zvýšení komfortu ovládání, je pro tuto akci možné použít metodu drag-and-drop, kdy je jednoduše řádek tabulky s vybranou zprávou přetažen do nové složky. Pokud je možné přetažení dokončit, zobrazí se u kurzoru s přetahovanou zprávou zelená potvrzující značka, namísto předchozí červené zamítající značky. Po tom, co se zpráva úspěšně zařadí do nové složky, je také dynamicky odebrána z tabulky právě otevřené složky, z níž byla původně přetažena, bez provedení jejího znovunačtení.

Metoda drag-and-drop také v aplikaci slouží k velmi jednoduché změně struktury stromu složek. Po uchopení je možné složku přesunout na jakékoliv jiné místo ve stromu, včetně jejího vnoření do jiné složky. Opět platí, že pokud je požadovaná změna struktury proveditelná, zobrazí se u kurzoru zelená potvrzující značka, v opačném případě červená zamítající značka. Tento přístup je velmi intuitivní a pohodlný na ovládání v porovnání s použitím klasických formulářových vstupů, kterými by bylo docíleno stejné funkcionality.

Změna velikosti tažením

Strom složek pošty se zobrazuje v grafickém rámci. Pokud je strom příliš vysoký a přesahuje přes spodní ohraničení tohoto rámce, například po rozbalení větví stromu, tak se zobrazí posuvník, kterým je možné strom posunout a zobrazit jeho nezobrazenou část. Uživatel ale může také upravit výšku tohoto rámce tažením jeho spodní hrany, dle svých vlastních preferencí. Například tak, aby se jeho velikost stala dostatečnou pro celý rozbalený strom a nebylo potřeba používat posuvník pro přístup ke spodním větvím složek.

Ovládání klávesnicí

Jak je již uvedeno v kapitole 4.3.5., ovládání webových aplikací klávesnicí je problematické a mělo by být stavěno pouze na úroveň alternativy k ovládání myší. Výjimku tvoří pouze rozhraní založená na webových appletech, což ale není případ modelové aplikace. Proto je implementováno ovládání klávesnicí pouze pro práci se stromem složek, čímž může být usnadněn pohyb po složitějších strukturách stromu. Pro práci se stromem se používají následující klávesy:

- Šipka doprava a doleva – rozbalení a zabalení větve.
- Šipka nahoru a dolů – pohyb ve stromu, přesun na předchozí nebo následující řádek.
- Mezerník – otevření označené složky.
- Klávesa Del – smazání označené složky.
- Klávesa F2 – přejmenování označené složky.

5.4. Porovnání s desktopovou aplikací

Jako desktopová alternativa modelové aplikace bude sloužit již dříve zmíněný program Microsoft Windows Mail. Tato kapitola provede jejich krátké srovnání.

Z výše uvedeného textu vyplývá, že aplikační logika webových aplikací může mít implementovánu jakoukoliv funkcionalitu, která je dostupná u desktopových aplikací. Z pohledu programátora se totiž tyto aplikace od sebe příliš neliší. Většinu běžných programovacích jazyků (jako jsou Java, C#, Ruby, Python, atd.) je možné použít, jak pro vývoj desktopových, tak i webových aplikací. Jediným požadavkem je dostatečné technické zázemí na serveru, kde bude aplikace nainstalována. Tento text proto nebude srovnávat konkrétní nabízené funkce aplikací.

Zaměříme-li se na uživatelská rozhraní a jejich ovládání, tak je zřejmé, že se modelová aplikace snaží nabídnout alternativně i takové způsoby ovládání, které jsou standardem pro desktopové aplikace. Pokročilá je především podpora práce s myší. Aplikace podporuje u některých objektů přesunutí tažením, změnu velikosti nebo metodu drag-and-drop. Windows Mail navíc nabízí pouze možnost přesunutí souboru z externího programu, metodou drag-and-drop, do přílohy nové zprávy. Tato funkcionalita není s technologiemi použitými v uživatelském rozhraní modelové aplikace realizovatelná. K podpoře ovládání myší je také možné zařadit kontextové menu, které Windows Mail nabízí. Je ho zde možné vyvolat například nad složkami nebo zprávami. Modelová aplikace kontextová menu nepoužívá, ačkoliv jejich implementace je možná. Důvodem je především méně nabízené funkcionality, než je tomu u Windows Mail, proto nejsou tato menu prakticky potřeba. Ovládací prvky vztahující se k aktuálnímu kontextu jsou zobrazeny již v rámci tohoto kontextu, a není je tak třeba přidávat do konktextového menu.

Ovládání klávesnicí samozřejmě implementuje desktopový e-mailový klient lépe, ale i zde je patrné, že klávesnice představuje pouze slabší alternativu k myši. Problematika ovládání webové aplikace myší je již diskutována v kapitole 4.3.5. Kvůli uvedeným důvodům modelová aplikace podporuje ovládání klávesnicí jen pro práci se stromem složek, stejně jako Windows Mail. Zde je ale uvedena spíše pro demonstraci této možnosti ovládání, než pro skutečný přínos uživateli.

Způsob zobrazení zpráv včetně jejich řazení je podobný u obou aplikací. Windows Mail ale obsahuje navíc rámec zobrazující náhled zprávy pro přečtení, který se načítá po jejím označení v tabulce. Modelová aplikace tyto náhledy neimplementuje, protože používá jiný systém označování. Windows Mail označí celý řádek tabulky reprezentující konkrétní zprávu po tom, co uživatel klikne kamkoliv do plochy tohoto řádku. U modelové aplikace je po kliknutí na řádek zprávy tato zpráva otevřena, což můžeme označit za akci, jež bude uživatel v tomto případě očekávat. Označení se provádí kliknutím na zaškrtávací políčko, které se nachází v levém rohu řádku se zprávou. Svázání tohoto označení s načtením náhledu by bylo pro uživatele matoucí. Implementace této vlastnosti uživatelského rozhraní by byla možná pouze se změnou uvedeného přístupu označování. Se způsobem zobrazení také souvisí posun tabulky zpráv, v případě, že obsahuje větší počet záznamů. Modelová aplikace používá stránkování, zatímco alternativní desktopová aplikace klasický posuvník. I když je možné v prostředí webových aplikací implementovat alternativy tohoto posuvníku, které jsou podrobně popsány v kapitole 4.3.4., tak se aplikace drží tradičního přístupu s běžným stránkováním, jež je uživateli obecně přijato.

Windows Mail i modelová aplikace nabízejí dynamické vyhledávání, kdy jsou při psaní vyhledávaného textu ihned zobrazovány zprávy, které jej obsahují. Rozdílný je ale způsob tohoto vyhledávání, jenž je důsledkem šifrování dat používaného modelovou webovou aplikací. Zatímco Windows Mail prohledává zprávy ve složce fulltextově, tak webová aplikace hledá zadaný text pouze v zobrazené tabulce zpráv (v případě, kdy je tabulka rozdělena na stránky, se prohledávají všechny tyto stránky). Implementace fulltextového vyhledávání by byla možná pouze v případě garance dostatečného výkonu na straně databázového serveru. Všechny zprávy v prohledávané složce by totiž musely být při této akci dešifrovány, což představuje velkou zátěž. Z tohoto důvodu je použito zjednodušené vyhledávání, které ale pro běžné sortování, například podle předmětu nebo odesílatele, stačí. Výše uvedené vlastnosti jsou shrnuty v tabulce 4.

Nebudou-li brány v potaz pouze otázky uživatelského rozhraní a jeho přístupnosti, tak má modelová aplikace několik charakteristických výhod oproti Windows Mail, které vyplývají z faktu, že se jedná o webovou aplikaci a obecně tedy platí i pro jiné webové aplikace. Tyto výhody jsou popsány v následujících bodech.

- Aplikace je platformově nezávislá. Jediný softwarový požadavek, který pro spuštění má, je dostupnost moderního internetového prohlížeče s podporou JavaScript.
- Protože se všechny složité výpočetní operace vykonávají na serveru, tak je hardwarově nenáročná. Na straně klienta se provádí pouze vykreslení stránky a interpretace JavaScriptu.
- Uživatel má své zprávy přístupné odkudkoliv, kde je schopen se připojit k síti Internet se zařízením, které má nainstalovaný internetový prohlížeč.

Vlastnost nebo funkce rozhraní	Web	Desktop
Tažení objektů		
Úprava stromu složek	+	+
Přesun zprávy do jiné složky	+	+
Přesun souboru z externího programu do přílohy zprávy	−	+
Změna velikostí bloků uživatelského rozhraní	+	+*
Ovládání klávesnicí		
Pohyb po stromu složek	+	+
Pohyb po tabulce zpráv	−	+
Pohyb po menu aplikace	−	+
Zobrazení zpráv		
Náhled zprávy	−	+
Definice sloupců tabulky zpráv	+	+
Výběr skupiny zpráv označením	+	+
Stránkování tabulky zpráv	+	−
Posun tabulky zpráv	−	+
Vyhledávání a řazení zpráv		
Dynamické zobrazení výsledků vyhledávání	+	+
Libovolné řazení zpráv	+	+

Tabulka 4. Porovnání dostupnosti některých charakteristických vlastností uživatelského rozhraní a funkcí modelové webové aplikace s alternativní desktopovou aplikací Microsoft Windows Mail.

* pouze částečná dostupnost

- Webové a databázové servery běžně používají RAID disková pole pro zabezpečení dat proti jejich ztrátě, případně se pravidelně zálohují. Klasický uživatel desktopové aplikace používá pro ukládání pouze jeden hardwarový pevný disk, po jehož selhání často přijde o veškerá data. Obecně tedy můžeme tvrdit, že data uživatele jsou v případě webové aplikace mnohem bezpečněji uchována.

- Pokud více uživatelů přistupuje k jedné instalaci aplikace na jednom serveru, tedy mají více uživatelských účtů, tak přechod na novější verzi u všech těchto uživatelů znamená pouze překopírování nových souborů se zdrojovým kódem na server. U Windows Mail by bylo nutné přeinstalovat program u všech uživatelů zvlášť.

5.5. Zabezpečení

Aplikace využívá specifický model šifrování veškerých uložených uživatelských dat a to v databázovém i souborovém systému. Snaží se tak zabránit problému diskutovanému v kapitole 3.3.2., tedy možnému zneužití těchto dat osobami, jenž mají reálný přístup k serverům využívaných aplikací a tím i k datům, které se na nich nachází.

Celý proces začíná vytvořením přihlašovacího účtu administrátorem, a to bez ohledu na to, zdali je vytvářen s administrátorskými nebo uživatelskými právy. Se založením nového účtuje je do databáze uložena dvojice přihlašovacího jména a otisk hesla, pro jehož výpočet je použit hashovací algoritmus SHA 256. Dále je tomuto účtu přidělen příznak, díky němuž je možné určit, že se jedná o nový účet, který ještě nebyl použit pro přihlášení.

Uživatel vstoupí na přihlašovací stránku a zadá přihlašovací jméno s heslem a odešle tuto dvojici na ověření serveru. Ze zadaného hesla se spočítá algoritmem SHA 256 otisk a případě, že je v databázi nalezena stejná dvojce uživatelského jména a vypočítaného otisku je autentizace považována za úspěšnou. V opačném případě je uživatel vrácen zpět na přihlašovací stránku a vyzván znovu k přihlášení. Z bezpečnostních důvodů je možné zadat pouze 5 neplatných přihlášení během 5 minut. Je tak zabráněno využití hrubé síly k prolomení hesla.

Po úspěšné autentizaci je ze zadaného hesla vypočítán další otisk algoritmem RIPEMD 160. Po výpočtu je otisk jednoduchými řetězcovými operacemi upraven na přesnou délku 256 bitů. Takto upravený otisk nazveme vstupní klíč. Pro zabránění útoku typu Session fixation je také uživateli vygenerován nový session klíč, kterým se bude nadále identifikovat serveru.

Dále je zjištěno, jestli má účet nastavený příznak nového účtu, jinými slovy, zdali se jedná o první přihlášení uživatele. Mohou nastat dva případy:

1. Uživatel nebyl přihlášen – je vygenerován náhodný řetězec o délce 128 bitů, který se uloží do session proměnné. Tento řetězec nazveme databázový klíč.

Databázový klíč je posléze zakódován symetrickou blokovou šifrou AES. Jako klíč je při zakódování použit dříve vypočítaný vstupní klíč. Takto zakódovaný databázový klíč je uložen do databáze a přihlašovacímu účtu zrušen příznak nového účtu.

2. Uživatel byl přihlášen – z databáze se načte zakódovaný databázový klíč. Ten je pomocí šifry AES za použití vstupního klíče dekódován a následně uložen do session proměnné.

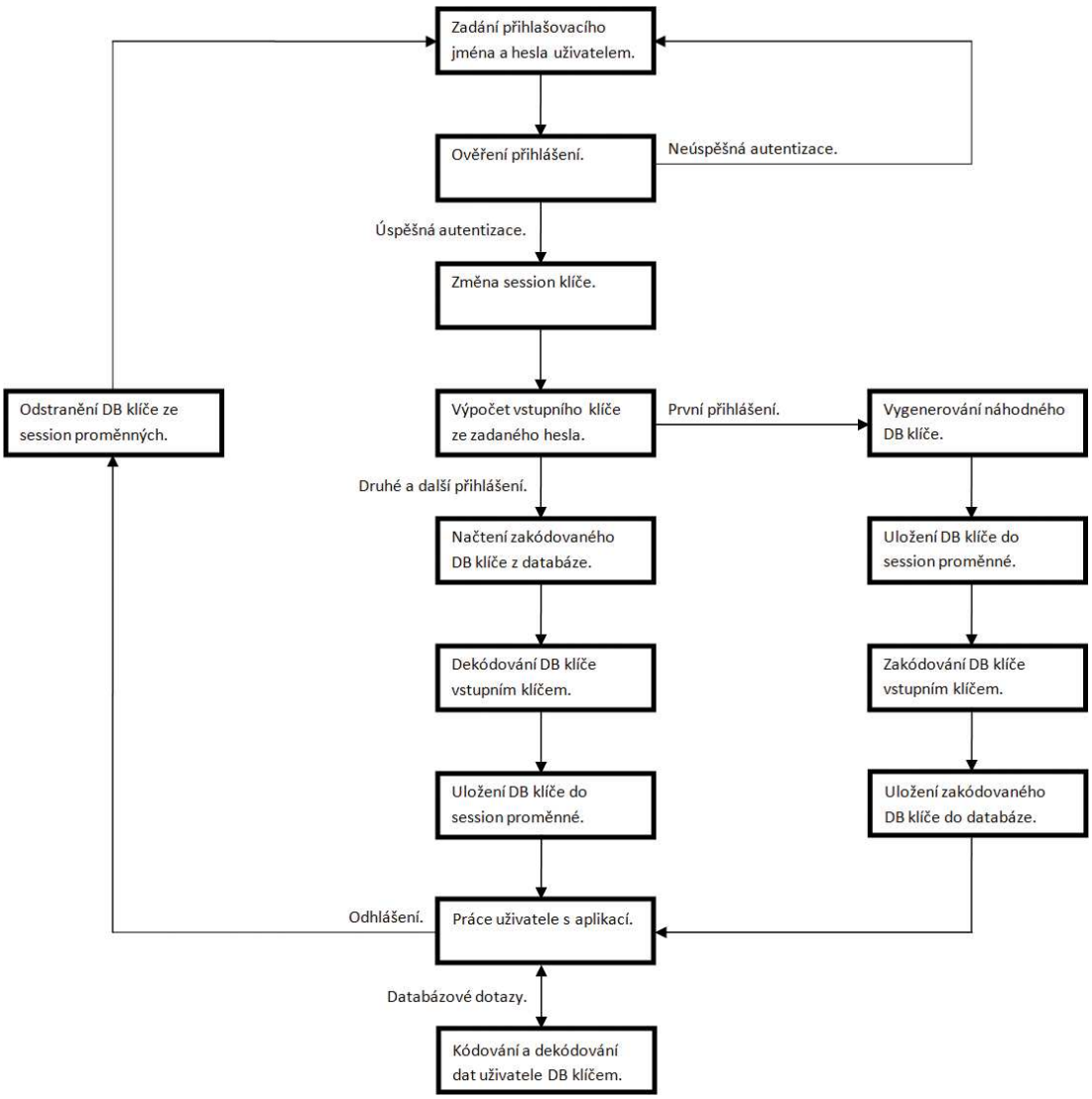
Všechna data, která přihlášený uživatel při práci s aplikací do databáze uloží, jsou zakódována šifrou AES za použití databázového klíče. Při potřebě jejich přečtení jsou tímto klíčem opět dekódována. Po ukončení práce s aplikací se uživatel odhlásí a databázový klíč se odstraní ze session proměnných.

Podobný přístup je využit i u souborů, které uživatel v aplikaci uloží (uložené soubory uživatelem v aplikaci představují výhradně přílohy zpráv). Před jeho uložením souboru je vygenerován náhodný řetězec o délce 128 bitů. Tento řetězec nazveme souborový klíč. Souborový klíč je uložen do databáze a jako všechna ostatní uživatelova data je zakódován šifrou AES za použití databázového klíče. Samotný soubor je zakódován symetrickou blokovou šifrou Twofish. Jako klíč je při zakódování použit souborový klíč. Pokud uživatel vyžaduje zobrazit uložený soubor, tak musí být přihlášený, což znamená, že má přidělený databázový klíč. Šifrou AES s databázovým klíčem je dekódován souborový klíč uložený v databázi. Souborovým klíčem je následně šifrou Twofish dekódován samotný vyžádaný soubor.

Síla využití vstupního klíče vyniká při změně hesla uživatele. Kdyby byl namísto vstupního klíče z hesla generován přímo databázový klíč, tak by změna hesla znamenala nutnost dekódovat starým klíčem veškerá data, která uživatel uložil a znovu je zakódovat novým klíčem. To by představovalo náročnou operaci, v prostředí webových aplikací s přiděleným omezeným výkonem ze serveru přímo nereálnou. Takto stačí pouze z nového hesla vypočítat i nový vstupní klíč. Tím je zakódován databázový klíč a následně uložen do databáze pro další přihlášení, již s použitím nového hesla.

Data uložená v databázi jsou tedy chráněna AES šifrou se 128 bitovým klíčem, který není nikde uložen a může jej vygenerovat pouze uživatel. Soubory jsou chráněny šifrou Twofish, přičemž každý má svůj 128 bitový klíč uložený v chráněné databázi. Je tak zcela odstraněn problém s přístupem třetích osob k serverům, ať už databázovým, souborovým nebo aplikačním, na kterých aplikace běží. Pro absolutní bezpečnost, výše popsaného způsobu šifrování, je nutné používat protokol HTTPS, který zaručuje, že na cestě mezi klientem a serverem nebudou žádná data odposlechnuta. Aplikaci je možné provozovat i po protokolu HTTP, v tomto případě je ale uživatel upozorněn varovnou hláškou, že používá nezabezpečený protokol.

Nevýhoda popsaného modelu šifrování je vyšší zátěž serveru, jistým způsobem také nutnost použití šifrovaného protokolu. Z uživatelského pohledu může



Obrázek 20. Schéma přidělení databázového klíče.

znamenat problém nemožnost obnovy uživatelského hesla a to ani jeho reset administrátorem. Nebylo by totiž možné znovu vypočítat platný vstupní klíč a všechna uživatelova data by byla nenávratně ztracena. Při zadávání hesla je proto uživatel na tuto skutečnost vždy upozorněn, aby byla zdůrazněna jeho důležitost a minimalizovala se možnost zapomenutí.

6. Ukázky webových aplikací

Jak již bylo v úvodu řečeno, tak webové aplikace představují stále sílící fenomén sítě Internet a uživatelé se s nimi pravidelně setkávají, běžně je používají a jsou tak přirozenou součástí webu. Nejnavštěvovanější webová stránka sítě Internet, služba Google, je vlastně také webová aplikace, která je zaměřená na vyhledávání dokumentů relativních se zadaným dotazem.

Z práce vyplývá, že možnosti webových aplikací jsou opravdu široké, proto mají také velkou oblast působnosti. Oblastí, ve kterých jsou dnes webové aplikace nasazovány, je nepřeberné množství. Můžeme se s nimi setkat mimo jiné při řízení firemních procesů, sestavování tréninkového plánu, zálohování souborů, editaci hudby nebo dokonce videa. Dvě oblasti jsou ovšem zcela tradiční. Je to komunikace, kam můžeme zařadit diskusní fóra, chaty nebo e-mailové klienty. Jejich vznik byl pouze logickou reakcí na nové přirozené prostředí, kde je možné tyto již existující služby provozovat. Dále je to komerce, především internetové obchody a aukce, které byly přítomné již v počátcích masivního používání webu. Podíváme-li se například na největší internetový obchod Amazon, tak zjistíme, že byl spuštěn již v roce 1995. V tomto roce byl také založen aukční web AuctionWeb, který byl později přejmenován na eBay.

I přes širokou oblast působnosti webových aplikací je můžeme obecně rozdělit do dvou obecných kategorií, které reflektují jejich základní filosofii:

Simulace desktopových aplikací Webové aplikace se snaží simulovat chování aplikací desktopových nabízenou funkcionalitou nebo uživatelským rozhraním. Přidaná hodnota zde spočívá v jejich dostupnosti z kteréhokoliv počítače s připojením k síti Internet. V těchto webových aplikacích má uživatel zpravidla dostupná i data, která v nich vytvořil. Data tedy není nutné přenášet mezi jednotlivými počítači. Jako ukázka tohoto typu webové aplikace může posloužit například modelová aplikace z 5. kapitoly.

Aplikace pro Web 2.0 Webová aplikace je založená na principech Webu 2.0, především na vytváření sdíleného obsahu a práci se sociálními atributy u uživatelů. Je zřejmé, že jsou tyto aplikace v prostředí webu původní a jejich desktopové varianty se prakticky nevyskytují. Příkladem může být webová aplikace sociální sítě Facebook.

Do těchto kategorií není možné aplikace striktně rozdělovat, v praxi se často vyskytují jejich kombinace. Zpravidla ale bývá u těchto aplikací jedna kategorie významnější. Jako příklad můžeme uvést simulaci libovolné desktopové aplikace, ke které je pouze přidáno sdílení dat. Je tedy zřejmé, že kritéria pro Web 2.0 zde nebudou důležitým faktorem.

Na následujících stránkách je čtenáři představeno několik zajímavých webových aplikací, přičemž každá je ze zcela jiné oblasti.

6.1. Adobe Photoshop Express Editor

Tato webová aplikace slouží jako jednoduchý WYSIWYG editor digitálních fotografií. Uživatel po spuštění nahraje fotografii určenou k úpravě v rozlišení do 16 Mpx (podporován je pouze formát JPG). Tato fotografie se zobrazí v uživatelském rozhraní a je na ní následně možné aplikovat dostupné grafické funkce. Ty jsou rozděleny do dvou kategorií – editační a dekorativní. Uživatel má k dispozici například vyvážení bílé, softwarové doostření, korekci červených očí, vytažení jedné barvy na černobílé fotografii, zkreslení tvarů, umístění obrázkových rámů, atd. Po tom, co uživatel dokončí úpravy, si může výsledný soubor z aplikace stáhnout zpět do lokálního počítače. Je mu při tom nabídnuto několik různých rozlišení.

Uživatelské rozhraní je zpracováno na platformě Adobe Flash. Je přehledné a celkově uživatelsky přívětivé. Všechny funkce mají jednoduché možnosti nastavení, často jsou přitom využívány výsledné náhledy místo textových popisků, což práci ještě zjednodušší. Je zřejmé, že bez použití appletu by takovýto druh aplikace nemohl mít srovnatelné rozhraní. Jediné limity zde spočívají v rychlosti překreslení změněné fotografie u plynulých změn. Jako obecnou nevýhodu této aplikace můžeme označit prodlevy při nahrávání a stahování velkých fotografií.

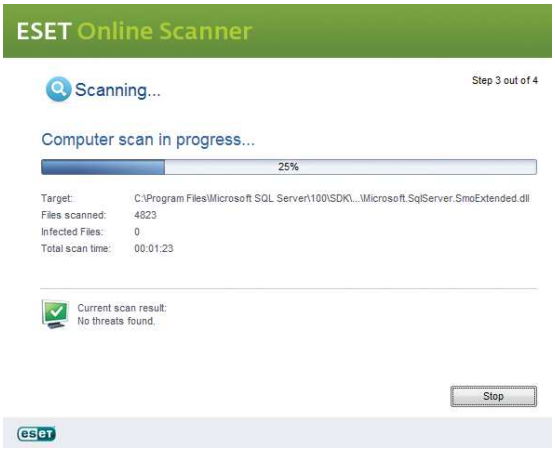
Obrázek 21. Adobe Photoshop Express Editor.

6.2. ESET Online Scanner

Webovou aplikací s nezvyklou funkcionalitou je ESET Online Scanner. Jedná se o nástroj, který vyhledává a odstraňuje škodlivý software lokálně v počítači uživatele. K tomu používá ovládací prvky ActiveX prohlížeče Internet Explorer, aplikaci tedy není možné spustit s jiným prohlížečem. Tyto ovládací prvky jsou digitálně podepsané společností ESET a instalují se ihned po spuštění aplikace. Pokud uživatel nemá v operačním systému právo instalovat software, tak nemůže tuto aplikaci používat. Po tom, co jsou potřebné ovládací prvky zavedeny, je spuštěn samotný test, který probíhá ve čtyřech krocích:

1. Nastavení parametrů testu (kontrola archivů, potenciálně nebezpečných nebo nechtěných aplikací, odstranění nalezeného škodlivého software, atp.).
2. Stažení aktuální virové databáze.
3. Skenování počítače.
4. Zobrazení výsledku testu.

Aplikace tedy nenahrazuje klasické antivirové programy, které se snaží zabránit napadení počítače škodlivým softwarem, nicméně nabízí kvalitní nástroj pro nalezení takového software, který se již v počítači nachází. Její výhoda je především neustále aktuální virová databáze a rychlé provedení testu bez jakékoliv registrace nebo instalování celého antivirového programu do operačního systému. Tuto webovou aplikaci je tedy možné například pohodlně použít i jako alternativu desktopové aplikace, která selhala při nalezení škodlivého software nebo pro provedení rychlého testu počítače, na kterém není žádný antivirový program nainstalovaný.



Obrázek 22. ESET Online Scanner.

6.3. Online OCR

OCR[10] je název metody pro rozpoznávání textu v obrázcích. Tyto obrázky zpravidla představují oskenované tištěné texty, které se po rozpoznání ukládají do klasických textových formátů. S těmi je pak možné pracovat jako s jakýmkoliv jiným textem.

Přesně tuto službu nabízí webová aplikace Online OCR. Uživatel nahraje ve formuláři soubor s obrázkem, který obsahuje text. Je-li těchto souborů více, tak je všechny postupně nahrává nebo je možné nahrát jejich ZIP archiv. Maximální velikost nahraného souboru je 30 MB. Dále vybere jazyk rozpoznávaného textu a požadovaný formát výstupu. Po odeslání tohoto formuláře jsou uživateli vráceny soubory ve vybraném formátu, které obsahují extrahovaný text z nahraných obrázků.

Uživatelské rozhraní je velmi jednoduché, v podstatě se omezuje pouze na jeden formulář. Důležité je pro tuto aplikaci kvalita výstupních souborů, která je určena počtem chyb v rozpoznaném textu. V tomto ohledu si stojí Online OCR velmi dobře. Bez problémů rozpoznává i český text, včetně formátování, a pokud je předloha v přiměřené kvalitě, tak se chyby v rozpoznaném textu prakticky nevyskytují. To není standard ani u některých desktopových ORC aplikací.

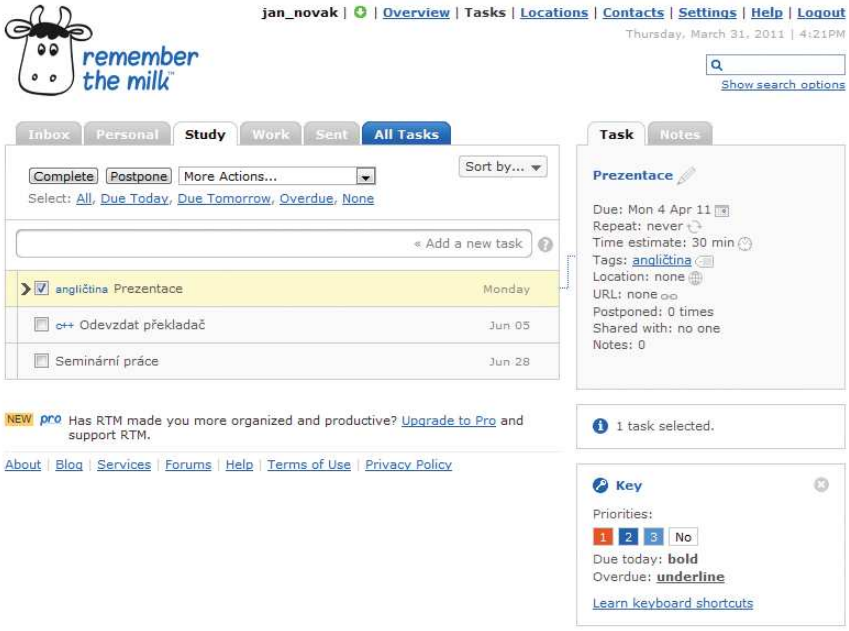
Obrázek 23. Online OCR.

[10]Optical Character Recognition; optické rozpoznávání znaků

6.4. Remember The Milk

Tato webová aplikace, jak již název napovídá, je zaměřena na správu úkolů. Nabízí přehledné a funkční rozhraní vytvořené bez jakýchkoliv appletů, ve kterém uživatel jednoduše organizuje jednotlivé úkoly. Ty je možné třídit dle různých kategorií, přiřazovat jim prioritu, sdílet je mezi ostatními uživateli atp. Na takto zanesené nastávající úkoly je pak uživatel upozorněn e-mailem, zasláním SMS na mobilní telefon nebo pomocí některé z IM služeb[11].

Zajímavostí je, že aplikace používá jako jedna z mála Google Gears, představené v kapitole 4.1.2. Je s ní tedy možné pracovat i v případě, kdy uživatel aktuálně nemá na svém počítači k dispozici připojení k síti Internet. Funguje tak lokálně v offline režimu a synchronizace dat se serverem proběhne po připojení uživatele. Použití této technologie implikuje závislost na internetovém prohlížeči, který ji podporuje, resp. je pro něj dostupná ve formě rozšíření. Naopak pro mobilní online uživatele je k dispozici zjednodušená varianta pro internetové prohlížeče palmtopů a mobilních telefonů. Tito uživatelé také patrně ocení možnost uložení úkolu do aplikace pouhým odesláním e-mailu.



Obrázek 24. Remember The Milk.

[11] Instant Messaging; např. IRC, ICQ, MSN Messenger, Excite, Ubique, atd.

Závěr

Technologie používané webovými a desktopovými aplikacemi na úrovni aplikační a datové vrstvy jsou velmi podobné a často i shodné. Výpočty logiky webových aplikací bývají zpravidla omezeny pouze konfigurací webového serveru, jenž danou aplikaci hostuje a limity sítě Internet v případě práce s velkými daty. Nebudeme-li brát tyto proměnné v potaz, tak můžeme tvrdit, že je funkcionalita nabízená oběma druhy aplikací shodná. Vývoj webových aplikací bývá ale náročnější, protože jsou u nich kladeny vyšší nároky na zabezpečení, je nutné komunikovat bezstavovým protokolem, výpočetní výkon sdílí více uživatelů atp. Zásadní odlišnost nalezneme až v prezentační vrstvě, kde je uživatelské rozhraní desktopových aplikací implementováno, nejčastěji grafickými knihovnami, zatímco u webových aplikací jsou použity klasické značkovací a stylovací jazyky původně určené k vytváření statických dokumentů. Pro zajištění jejich interaktivity se používá skriptování na straně klienta – internetového prohlížeče. Takto se snaží simulovat chování a ovládací prvky, které nabízí uživatelům desktopové aplikace. Tím je možné v uživatelských rozhraních nabídnout i pokročilejší prvky, nicméně limitující jsou v tomto směru aktuální standardy. Velkým přínosem by proto měla být nová specifikace HTML 5 a CSS 3. Moderní webové aplikace tedy mohou bez větších obtíží konkurovat desktopovým aplikacím, je proto ovšem nutné vynaložit větší úsilí ve fázi vývoje. Drobné nedostatky webových aplikací, které jsou důsledkem především zcela jiné formy uživatelského rozhraní, snadno vyrovnají pozitiva plynoucí z obecných výhod webových aplikací. Především je to dostupnost aplikace a dat jejich uživatelů z kteréhokoliv počítače s připojením k síti Internet.

Conclusions

Technologies used in application layer and data layer of web and desktop applications are very similar to one another or even identical. Web application computations are usually limited to web server configuration and Internet limits in case of large data processing. If we do not take into consideration these technicalities, the functions offered by both types would be alike. The web applications development is more complex because of higher requirements for security, handling state over stateless protocols, shared computation resources, and etc. The fundamental differences can be found only in the presentation layer. The user interface of desktop applications are usually implemented via graphic libraries. Web applications use classic markup and style languages which were originally designed for creating static documents. Scripting on the client's side (in the browser) is usually used for ensuring interactivity. This is the method to simulate the appearance and behavior of desktop applications. It offers advanced features in the user interface. However, current standards have limits on these features. Therefore, new specifications of HTML 5 and CSS 3 would be a major benefit. So modern web applications are easily comparable with desktop applications, but programmers have to make greater effort during the development phase. Minor shortcomings of web applications, mostly resulted from other forms of user interfaces, can easily compensate the positives resulting from general benefits. Above all, web applications and user's data can be available from any computer connected to the Internet.

Reference

[1] BUREŠ, Mirosla; MORÁVEK, Adam; JELÍNEK, Ivan. *Nová generace webových technologií: Informace v 21. století.* Vyd. 1. Praha: 1. VOX a.s. – Nakladatelství, 2005. 264 s. ISBN 80-863224-46-X.

[2] DOSTÁL, Martin. *Základy tvorby uživatelského rozhraní.* Olomouc: Katedra informatiky Přírodovědecká fakulta Univerzita Palakcého v Olomouci, 2010. 72 s.

[3] Google Inc. *Gears* [online]. 2008. Dostupné z: <http://gears.google.com>.

[4] GARRETT, Jesse James. *Adaptive Path* [online]. 2010-02-18 [cit. 2011-02-23]. Ajax: A New Approach to Web Applications. Dostupné z: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>.

[5] INSTONE, Keith. *Location, Path & Attribute Breadcrumbs* [online]. 2002 [cit. 2011-03-27]. Dostupné z: <http://instone.org/files/KEI-Breadcrumbs-IAS.pdf>.

[6] *Wikipedie* [online]. 2010-12-12 [cit. 2011-02-16]. Man in the middle. Dostupné z: <http://cs.wikipedia.org/wiki/Man_ in_ the_ middle>.

[7] *Wikipedie* [online]. 2011-2-3 [cit. 2011-02-17]. HTTP cookie. Dostupné z: <http://cs.wikipedia.org/wiki/HTTP_ cookie>.

[8] *Wikipedie* [online]. 2011-2-24 [cit. 2011-03-15]. Internetová horečka. Dostupné z: <http://cs.wikipedia.org/wiki/Internetová_ horečka>.

[9] World Wide Web Consortium. *HTML 4.01 Specification: W3C Recommendation* [online]. 1999-12-24. Dostupné z: <http://www.w3.org/TR/html401/>.

[10] World Wide Web Consortium. *XHTML^{TM} 1.1 – Module-based XHTML, Second Edition: W3C Recommendation* [online]. 2010-11-23. Dostupné z: <http://www.w3.org/TR/xhtml11/>.

[11] World Wide Web Consortium. *W3C Confirms May 2011 for HTML5 Last Call, Targets 2014 for HTML5 Standard* [online]. 2011-2-14. Dostupné z: <http://www.w3.org/2011/02/htmlwg-pr.html/>.

A. Obsah přiloženého DVD

Struktura složek DVD

`bin/`
: `phpmc.zip` – Zdrojové soubory modelové webobé aplikace.
`readme.txt` – Instrukce k instalaci.

`doc/`
: `belaska.*` – Text této práce v digitální podobě.
`napoveda.pdf` – Krátká uživatelská příručka.

`data/`
: `form.html` – Formulář zobrazený na obrázcích 6. – 11.

Struktura složek modelové aplikace

`attachment/`
: Prázdná složka; slouží k ukládání souborů s přílohami zpráv.

`classes/`
: PHP třídy.

`configs/`
: `limits.php` – Administrátorská nastavení.
`mysql.php` – Nastavení připojení k MySQL databázi.
`smarty.php` – Nastavení Smarty.

`css/`
: Kaskádové styly a obrázky.

`includes/`
: Vkládaný PHP kód.

`install/`
: Instalátor.

`js/`
: Soubory s JavaScript kódem.

`langs/`
: Adresáře s jazykovými mutacemi aplikace.

`libs/`
: Externí knihovny.

`requests/`
: Aplikací volaný PHP kód.

`templates/`
: Smarty šablony prezentační vrstvy.

  `compile/`
  : Prázdná složka; slouží k ukládání zkompilovaných šablon.

`.htaccess` – Soubor pro vypnutí RewriteEngine.

`index.php` – Spouštěcí soubor aplikace.

`login.php` – PHP kód přihlašovacího formuláře.

`mail.php` – PHP kód zavedení uživatelského rozhraní.

`redirect.php` – PHP kód ověření přihlášení.