Webové aplikace – principy, nástroje. Vícevrstvé aplikace. Zabezpečení aplikace.

1. Principy a nástroje

Webová aplikace:

- aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť Internet nebo její vnitropodnikovou obdobu intranet
- aplikace založena na systému komunikace klienta (prohlížeč klientského počítače) a aplikace běžící na webovém aplikačním serveru
- komunikace zpravidla probíhá přes protokol HTTP/HTTPS
- klient vysílá požadavek (HTTP REQUEST), server zpracuje adekvátním způsobem a posílá nazpátek odpověď (HTTP RESPONSE)
- zpracování požadavku na serveru má na starosti mechanismus, který je příslušným způsobem vyhodnotí a předá dále do aplikace. Zasílá nazpátek odpovědi klientovi tak, že generuje výsledné webové stránky
- protokol je bezstavový, nemá stálé spojení s kleinty a nemůže je proto identifikovat - velké komplikace pro webové aplikace, které vyžadují stavovou informaci, např. nákupní košík; řešení:
    i. přenášení údajů v URL a skrytých polí formuláře
    ii. cookies
    iii. session proměnné
- populární především pro všudypřítomnost webového prohlížeče jako klienta
- schopnost aktualizovat a spravovat webové aplikace bez nutnosti šířit a instalovat software na potenciálně tisíce uživatelských počítačů je hlavním důvodem jejich oblíbenosti
- statické stránky (do ½ 90. let)
    i. text, obrázky
    ii. akademická sféra, odborná veřejnost
    iii. uživatelé mohou obsah webových stránek ovlivnit jen minimálně
- dynamické stránky (od ½ 90. let)
    i. multimediální obsah
    ii. přístupné pro nejširší veřejnost
    iii. interaktivita s uživatelem

- aplikační (business) logika a datová část je umístěna na straně serveru
- webový aplikační server je kontejner, umožňující spouštět skripty příslušného jazyka
- používané technologie na straně klienta – XHTML, CSS, JavaScript, VB.Script atd.
- používané technologie na straně serveru
    i. interpretované – ASP, PHP, Perl, Java… (skriptovací jazyk se nepřekládá do strojového kódu, ale je přímo vykonáván interpretorem)
    ii. kompilované – ASP.NET (aplikace jsou předkompilovány do jednoho či několika málo DLL souborů)
- Common Gateway Interface (CGI)
    i. je protokol pro propojení externích aplikací s webovým serverem
    ii. definuje jak má webový server komunikovat s ostatními aplikacemi (CGI-skripty)
    iii. CGI skript je externí program, který je na požadavek od uživatele spuštěný WWW serverem jako samostatný proces. CGI skripty přebírají data zadaná uživatelem, zpracovávají je a jako výsledek vytvářejí většinou HTML stránky. Tyto dynamicky vytvořené stránky pak WWW server posílá zpět klientovi.

	Linux	Windows	Apache	IIS	Lighttpd	PHP	ASP.NET	Perl	Python	Java	MySQL	SQL Server	Memcached
Flickr	x		x			x		x		x	x		x
YouTube	x		x		x				x		x		
PlentyOfFish		x		x			x					?	
Digg	x		x			x					x		x
TypePad	x		x					x			x		x
LiveJournal	x		x					x			x		x
Friendster	x		x			x		x			x		
MySpace		x		x			x					x	
Wikipedia	x		x		x	x					x		x

Vícevrstvé aplikace

První softwarové aplikace využívaly jednoduchou monolitickou architekturu, která se snadno implementovala a neměla náročné požadavky na technické vybavení. V tomto případě byla databázová, aplikační i prezentační logika soustředěna v jednom monolitickém bloku. Koncept vícevrstvých architektur datuje své počátky do období rozvoje lokálních sítí. Programy využívající síť pro přístup k informacím v databázích jsou podle tohoto konceptu rozděleny do vrstev podle funkce. Uspořádání architektury může mít vrstvy dvě nebo tři.

Model třívrstvé architektury je přímou evolucí, z dnešního pohledu již koncepčně zastaralé, dvouvrstvé architektury. Dvouvrstvá architektura vychází z vrstvy klientské a vrstvy datové. Klient obsahuje většinu aplikační logiky, se kterou pracuje přímo nad datovým zdrojem. Nevýhody tohoto modelu se ukázaly při vzrůstající komplexnosti klientských aplikací. Se složitostí aplikací vzrůstaly výkonové nároky na klientské počítače, s masovým rozšířením aplikací vrůstaly nároky na sdílení zdrojů, omezení datových přenosů apod.

Řešení se našlo v podobě přidání třetí - střední vrstvy. Díky nově definované aplikační vrstvě bylo možné aplikační logiku, která do té doby ležela na klientu, přesunout na aplikační server. Díky tomuto tahu se klientům značně ulevilo, protože veškerý výpočetní výkon byl přesunut na výkonné servery. Díky přesunutí aplikační logiky na jedno místo bylo možné dosáhnout jejího sdílení a lepší správy a dostupnosti. Významnou měrou se podařilo redukovat datový přenos, jehož těžiště se přesunulo na trasu mezi aplikačním serverem a datovým zdrojem.

2. Zabezpečení aplikace

Zabezpečení přístupu

- pro identifikaci uživatele přistupujícího k aplikaci se nejčastěji používá zadání jména a hesla. Tyto údaje by měli být navíc před odesláním nebo uložením šifrovány (např. MD5, SHA1), aby je nebylo možné odchytit při posílání např. po síti nebo získat z úložiště na disku. Po zašifrování je manipulováno pouze s tzv. otiskem hesla. Šifrovací funkce je jednosměrná, určení původního hesla z jeho otisku je nemožné (téměř). Pro zvýšení odolnosti proti slovníkovým útokům se přidává k šifrovanému heslu náhodný řetězec (hash = MD5 (heslo . retezec))
- mezi další možnosti patří ověření uživatele nebo ověření totožnosti serveru pomocí certifikátů. Server posílá klientovi certifikát. Certifikát spojuje dohromady počítač s reálně existující osobou (fyzickou či právnickou). Certifikát vydává certifikační autorita (CA) – ta by měla ověřit skutečnou identitu žadatele o certifikát. Prohlížeč automaticky věří certifikátům od CA, které zná (umí ověřit podpis na certifikátu).
- na úrovni databázové vrstvy lze pro uživatele aplikace nastavit příslušnou úroveň oprávnění – přístup ke konkrétní databázi nebo tabulce. S vytvářením oprávnění se váže princip nejnižšího uživatelského práva: Uživatel (nebo proces) by měl mít nejnižší úroveň práv, která mu umožňuje provést patřičný úkol.

Zabezpečení dat

- validace dat – kontrola správnosti, úplnosti a konzistence vkládaných dat. Lze ji provádět na úrovni klienta (JavaScript), na úrovni serveru (PHP, Java Servlet), na úrovni databáze (SQL omezení), nebo kombinací těchto technik.
- šifrování přenášených dat – zvýšení bezpečnosti pomocí šifrování přenášených dat je jednou ze základních vlastností aplikací jako jsou online bankovnictví, business komunikace apod. Pro zabezpečení přenosu v rámci internetu mezi serverem s webovou prezentací a prohlížečem (uživatelem) se používají protokoly SSL (Secure Sockets Layer) nebo TLS (Trasport Layer Security), přičemž SSL je předchůdce TSL. SSL je protokol, resp. vrstva vložená mezi vrstvu transportní (např. TCP/IP) a aplikační (např. HTTP), která poskytuje zabezpečení komunikace šifrováním (a autentizací) komunikujících stran. Každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci rozšifrovat jen majitel použitého veřejného klíče svým soukromým klíčem. Ustavení SSL spojení (SSL handshake) pak probíhá následovně:
    i. klient pošle serveru požadavek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.),
    ii. server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru,
    iii. podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru,
    iv. na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude kódovat následná komunikace. Ten zakóduje veřejným klíčem serveru a pošle mu ho.,
    v. server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč,
    vi. klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze „handshake“ tímto končí,
    vii. je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
    viii. aplikace od teď dál komunikují přes šifrované spojení. Například POST požadavek na server se do této doby neodešle.