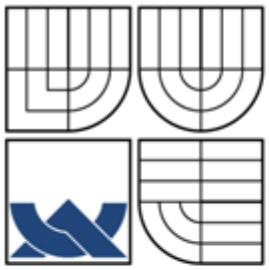
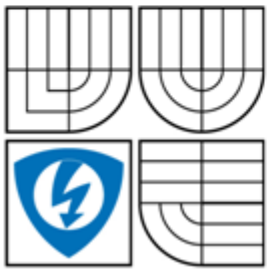


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A
KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

POROVNÁNÍ TECHNOLOGIÍ PRO TVORBU INFORMAČNÍCH SYSTÉMŮ

COMPARISON OF TECHNOLOGIES FOR CREATION OF INFORMATION SYSTEMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

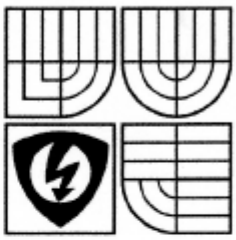
AUTOR PRÁCE Bc. LUKÁŠ HUBENÝ

AUTHOR

VEDOUCÍ PRÁCE Ing. RADOVAN HOLEK, CSc.

SUPERVISOR

BRNO 200



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Hubený Lukáš, Bc. ***ID:*** 54122
Ročník: 2 ***Akademický rok:*** 2007/08

NÁZEV TÉMATU:

Porovnání technologií pro tvorbu informačních systémů

POKYNY PRO VYPRACOVÁNÍ:

Proveďte porovnání technologií používaných při tvorbě informačních systémů. Zaměřte se na technologie JSP, PHP, Perl, DOT NET a jejich spolupráci s databázovými systémy. Výhody jednotlivých platforem demonstrujte na ukázkových příkladech.

DOPORUČENÁ LITERATURA:

Termín zadání: 3.12.2007 ***Termín odevzdání:*** 26.5.2008

Vedoucí projektu: Ing. Radovan Holek, CSc.

prof. Ing. Pavel Jura, CSc.
předseda oborové rady

UPOZORNĚNÍ:

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Lukáš Hubený
Bytem:	Vojtěchov 44
Narozen/a (datum a místo):	11.4.1984 v Chrudimi

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta	Elektrotechniky a komunikačních technologií
se sídlem	Údolní 53, 602 00 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Pavel Jura, CSc., vedoucí ústavu automatizace a měřicí techniky
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

□ disertační práce
■ diplomová práce
□ bakalářská práce
□ jiná práce, jejíž druh je specifikován jako .....
(dále jen VŠKP nebo dílo)

Název VŠKP: Porovnání technologií pro tvorbu informačních systémů _____

Vedoucí/ školitel VŠKP: Ing. Radovan Holek, csc. _____

Ústav: Ústav automatizace a měřící techniky_____

Datum obhajoby VŠKP: _____

VŠKP odevzdal autor nabyvateli v*:

■ tištěné formě – počet exemplářů: 2

■ elektronické formě – počet exemplářů: 2

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ■ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 26.5.2008

……..	……
Nabyvatel	Autor

Anotace

V této diplomové práci jsou porovnávány technologie pro tvorbu informačních systémů. Jsou srovnávány čtyři základní technologie: Php, ASP.NET, JSP a Perl. Porovnání technologií probíhá ve dvou fázích: První fáze je vesměs teoretická, která řeší teoretické výhody a nevýhody jednotlivých platforem a část praktická, kde jsou ukázány na krátkých příkladech výhody platforem při programování informačního systému. Porovnává se i spolupráce s databázemi.

Klíčová slova

Informační systém, Php, ASP.NET, JSP, Perl

Annotation

The topic of this master thesis is a comparison of the technologies for information systems creation. There are compared four fundamental technologies: Php, ASP.NET, JSP and Perl. The comparison of these technologies have two main stages: The first stage is mostly theoretical and demonstrate theoretical advantages and drawbacks of these platforms. The second stage demonstrates on some short examples advantages of platforms during information systems programming. Co-operation with database systems is compared too.

Keywords:

Information system, Php, ASP.NET, JSP, Perl

Bibliografická citace

HUBENÝ, Lukáš. *Porovnání technologií pro tvorbu informačních systémů*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 81s., 1 CD příloha. Ing. Radovan Holek, CSc .

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma "Porovnání technologií pro tvorbu informačních systémů" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne 26.5.2008 Podpis:

P o d ě k o v á n í

Děkuji vedoucímu mé diplomové práce Ing. Radovanu Holkovi, CSc. za udávání směru a za cenné připomínky při vypracování diplomové práce.

V Brně dne 26. 5. 2008 Podpis:

OBSAH

1. ÚVOD

Dnešní doba je dobou spěchu, nových věcí a inovací. Společnosti na trhu vynakládají spoustu energie k tomu, aby přilákaly zákazníka a prodaly svůj produkt. Velkou zbraní k prezentaci, reklamě a prodeji výrobků se stal internet, který se celkem nenápadně začal rozšiřovat a pak nastal jeho velký třesk. Dnes už si spousta lidí neumí život bez internetu představit, protože je jejich nástrojem pro práci a zdrojem informací. S tím, jak rychle se internet rozšiřoval, vznikaly i nové technologie pro prezentaci a sdílení dat na internetu.

Nejdříve to byla potřeba vědců sdílet informace o výzkumech, na což stačily dnes již stařičké Html stránky, kde se obsah prezentoval pouze jako nějak formátovaný text doplněný obrázky a případně tabulkami. Samozřejmě toto ve světě 21. století nemohlo dlouho vydržet, a proto se začaly vyvíjet jazyky, které by umožňovaly interakci s uživateli. Například možnost zobrazení dat z databáze, přizpůsobení vzhledu jednotlivým uživatelům podle jejich potřeb.

Dobrý webový server poskytuje uživatelům vše, co potřebují. K takovému serveru se uživatel vrátí. Pokud chcete mít dobrý internetový obchod, měl by být váš server stále aktuální, ale nejen to. Měl by rovněž obsahovat informace přizpůsobené na míru jednotlivým zákazníkům. Dále by měl umožňovat jednoduché nastavení vlastního uživatelského prostředí.

Výše zmíněné potřeby splňují tzv. dynamické stránky, ale způsobů, jak je vytvořit, je mnoho. Protože řada výrobců věděla, že tato „dynamičnost“ bude vyžadována, začaly tvořit jazyky, které by to umožňovaly.

Úkolem této semestrální práce je zhodnotit několik nejvýznamnějších technologií pro tvorbu dynamických stránek, představit jejich výhody,vady a stanovit „optimální oblast použití“. Jsou to zejména: JSP, Php, Perl a ASP.NET. Dříve než začnu tyto jazyky blíže popisovat, tak si krátce představíme i jejich „webové kolegy“, abychom získali komplexnější přehled o dostupných technologiích dnešních webů.

2. JAZYKY WEBOVÝCH STRÁNEK

2.1 STATICKÉ STRÁNKY

Statické stránky jsou psané tzv. Html kódem, který je souborem značek(tagů), které umožňují autorovy nejen text různě formátovat, zarovnávat, ale také vkládat obrázky, odkazy a další prvky. Html stránky se obvykle skládají z jednoduchého textu, obrázků a hypertextových odkazů na jiné dokumenty.

Pomocí CSS si lze formátování výsledného dokumentu zjednodušit tím, že jej napíšeme jednou do speciálního souboru, který potom vložím do dané stránky, místo psaní formátování u každého. Ale ani CSS nezachrání statické stránky před úpadkem. Jestliže chceme na statické stránce něco upravovat, musíme zasahovat do Html kódu, a pak je aktualizované umisťovat zpět na server, což není zas tak náročné, ale proč to dělat složitě, když to jde i jednoduše. Zkuste si představit, jak by se dal vytvořit internetový obchod jen pomocí statických Html stránek, prostě nemožný úkol pro tuto už dosti starou technologii. A proč nemožné? Statické stránky nemají implementovány nástroje pro práci s databázemi.

2.2 AKTIVNÍ WEBOVÉ STRÁNKY – NA STRANĚ KLIENTA

Tyto technologie jsou relativně často inovovány. Hlavní nevýhodou implementování funkčnosti na straně klienta je to, že správce webového serveru nemůže ovlivnit software používaný pro zobrazení stránky. Jelikož chtějí firmy obsáhnout tolik uživatelů s různými prohlížeči, kolik je možné, je přijímání nových technologií velmi pomalé, protože je podporují pouze nejnovější verze nejrozšířenějších prohlížečů. Na rozdíl od toho, technologie na straně serveru typicky nevyžadují žádný konkrétní prohlížeč, takže jejich přijímání je obecně rychlejší.

2.2.1 Ovládací prvky AktiveX

Ovládací prvky AktiveX jsou samostatné programy, známé jako „komponenty“ , které jsou napsány v jazycích jako je Visual C++ nebo Visual Basic.

Když jsou přidány do webové stránky, poskytují určitý kus funkčnosti, jako jsou sloupcové diagramy, grafy, čítače, ověřování klientů a nebo přístup k databázím. Jsou do stránky přidávány pomocí značek <Object>, které jsou nyní součástí standardu Html. Ovládací prvky AktiveX, vložené do webové stránky, mohou být spouštěny prohlížečem nebo serverem. Byly vyvinuty společností Microsoft, a proto nejsou bez zásuvného modulu podporovány konkurenčními prohlížeči.

2.2.2 Java Applety

Applet je program napsaný v programovacím jazyce Java, který může být vložen do stránky Html téměř stejným způsobem jako třeba obrázek. Když používáme prohlížeč, který má povolenu Javu pro zobrazení stránky, jež obsahuje applet, kód appletu je přenesen do vašeho systému a proveden prohlížečem. Protože je applet napsaný v jazyce Java, má všechny jeho výhody a je nezávislý na platformě.

2.2.3 Skriptování na straně klienta a DHtml

Skriptovací jazyky poskytují nováčkům přístupnější cestu k programování. Skriptování na straně klienta pro použití na webu bylo vyvinuto pro poskytnutí dynamické alternativy ke statickému Html.

Hlavním skriptovacím jazykem na straně klienta je JavaScript (původně LiveScript). Je podporován od Internet Explorer verze 3.0 a výše a v ostatních prohlížečích též. JavaScript není to samé jako Java!

Dynamické Html (DHtml) je podobné skriptování, ale s tím, že je interpretováno na úrovni prohlížeče, který vytváří reprezentaci stránky v Html. Ve skutečnosti se dynamické Html od skriptování liší pouze tím, že umožňuje přístup k některým dalším funkcím, jako je např. animace stránek a přesné umisťování grafiky a textu pomocí absolutních souřadnic. Prohlížeč bude stále vytvářet stránky z čistého Html kódu.

2.3 AKTIVNÍ WEBOVÉ STRÁNKY – NA STRANĚ SERVERU

Před pár lety bylo jediným způsobem přenášení dynamických dat na web něco, co se nazývalo CGI (Common Gateway Interface). Programy CGI poskytovaly relativně jednoduchou cestu pro vytváření webových aplikací, které akceptují uživatelský vstup, posílají dotazy na databázi a vrací nějaké výsledky zpátky prohlížeči. Společnost Microsoft i Netscape vyvinuly vlastní rozhraní API, které lze použít pro vývoj vnitřního kódu pro obsloužení požadavků webu.

2.3.1 CGI (Common Gateway Interface)

Rozhraní CGI je nejstarší a nejlépe prověřenou technologií pro tvorbu dynamických webových stránek. Díky CGI může server odpovídat na uživatelské požadavky spuštěním jakéhokoliv počítačového programu. Programy CGI jsou obvykle napsány v jazyku Perl, ovšem nic nebrání tomu, abyste v případě potřeby použili jakýkoliv jiný programovací jazyk. Tak či onak je obvykle použit skriptovací jazyk.

Programy ve skriptovacích jazycích nejsou kompilovány – zůstávají stále v textovém formátu. Každý uživatelský požadavek způsobí opětovné načtení textu programu, jeho interpretaci a vykonání příslušných instrukcí. Ke zpracování popisovaného procesu je ovšem zapotřebí čas. Tento faktor nabývá na síle především v případě velmi vytížených serverů.

Asi největší nevýhodou programování CGI je to, že není škálovatelné. To znamená, že pokaždé, když je webovým serverem přijat požadavek, je vytvořen celý nový proces. Každý proces se skládá z jeho vlastní sady proměnných prostředí, samotné instance požadovaného prostředí, kopie programu a paměti přiděleného programu. Při přijetí mnoha požadavků je server velmi silně namáhán a to může způsobit i pád serveru.

Zde můžou pomoci technologie FastCGI a mod:Perl serveru Apache, které se zaměřují na výkon.

2.3.2 JavaScript – na straně serveru (Server Side Javascript)

JavaScript na straně serveru (SSJS) je odpovědí firmy Netscape na ASP firmy Microsoft. Stejně jako ASP se stránky, které používají SSJS, se skládají z Html kódu a vložených skriptů. Tento kód je prováděn na serveru a produkuje webovou stránku skládající se z holého Html, která je odeslána prohlížeči.

SSJS má tu výhodu, že používá JavaScript, který je standardním jazykem webu. Nicméně má jednu menší nevýhodu oproti ASP a Php a to, že aplikace, které používají SSJS, musí být před svým spuštěním kompilovány. To zvyšuje složitost modifikování stránek SSJS. Vážnějším nedostatkem je to, že SSJS je nyní podporováno pouze serverem Enterprise Server, který zaostává za Apachem i IIS od Microsoftu. V dnešní době se používá velmi zřídka.

2.3.3 Servlety a JSP

Viz další kapitoly

2.3.4 Php

Viz další kapitoly

2.3.5 Perl

Viz další kapitoly

2.3.6 ASP.NET

Viz další kapitoly

3. VLASTNÍ SROVNÁNÍ

Jelikož chceme srovnávat technologie, které se vyvíjejí takovou rychlostí, že je téměř každého půl roku na světě nová verze, je potřeba si ujasnit hned na začátku, které verze těchto technologií budou srovnávány. Ne všechny verze byly vydány ve stejnou dobu, ale většinou jsou v současnosti (konec 2007) nejpoužívanější.

- Php – bude to verze Php5, která už je „bezpečně zajeta“
- ASP.NET - verze ASP.NET 2 (FrameWork 2.0)
- JSP – verze JSP 2.0
- Perl –verze 6

3.1 INSTALACE

V této kapitolce bude porovnána složitost či jednoduchost instalace programů a serverů potřebných pro tvoření na jednotlivých technologiích. Je to víceméně subjektivní srovnání a beru zde na vědomí i stížnosti a chválu z různých diskuzí. Samozřejmě výhodu, zde mají rozšířené technologie, protože návodů na instalaci je na internetu spousta. Do této instalace zahrnuji i instalaci potřebného serveru a případně i databázového serveru, protože jsou k celkové funkci zapotřebí.

3.1.1 Php

Php nabízí pro začátečníky programátory balíčky (triády), které obsahují, jak Php, tak nastavený server Apache, tak i databázi MySQL. Výhodou těchto balíčků je jejich velmi jednoduchá instalace bez různého nastavování, ale nevýhodou je právě to, že si uživatel nemůže spoustu věcí nastavit sám.

Jinou možností je nainstalovat si Php, MySQL a Apache zvlášť a umístit si je do vlastních složek a nastavit si potřebné parametry v souborech *Php.ini* a *http.conf*. Jak měnit tyto soubory je popsáno na mnoha místech. Php spojuji se serverem Apache, ale samozřejmě to není podmínkou a Php může použít i jiný server a to samé platí o MySQL.

3.1.2 ASP.NET

Technologie ASP.NET je provozována na operačním systému Windows a pro její funkčnost je potřeba komponenta zvaná NET Framework, která se stará o ty nejzákladnější „úkoly“, jako je správa paměti, rušení objektů, spouštění vláken, natahování potřebných knihoven atd.

Další věcí, kterou potřebujeme je webový server. Pro vývoj na tzv. localhost si vystačíme s vestavěným serverem *ASP.NET Development Server.* Pro normální použití je nejlepší použít *Internet Information Serverem* (IIS), který je součástí OS Windows XP Profesional, jinak se musí doinstalovat.

Poslední věcí je databázový server pro ASP.NET. Buď se můžeme pomocí ODBC rozhraní připojit prakticky ke každé databázi, ale víc bych se přiklonil použít „vestavěný“ databázový server *SQL Server 2005 Express Edition,* který je v Express verzi zdarma a dostačující.

Aby bylo vše kompletní, tak při tvorbě (nejen) ASP.NET budeme používat pro programátory velice komfortní nástroj *Visual Studio 2005* (edici *Visual Web Developer 2005 Express Edition*).

Ve výsledku tedy budeme muset instalovat tři „programy“, ale jde vetšinou pouze o mačkání tlačítka *Další.* Většinu nastavení si uživatel zvolí hned při instalaci a není nucen přepisovat již žádné textové soubory.

3.1.3 JSP

Opět budeme potřebovat nějaký server a opět bych doporučil server (kontejner) Tomcat(http://tomcat.apache.org), který umí běžet ve dvou módech a to buď samostatně a nebo ho propojit se serverem Apache. Potom požadavky na JSP stránky zpracovává kontejner Tomcat a zbytek server Apache. Tato souběžná práce je umožněna tím, že Apache poslouchá standardně na portu :80 a Tomcat standardně na portu :8080. Ověření, zda Tomcat můžeme ověřit, když v internetovém prohlížeči napíšeme *http://localhost:8080/* . Bohužel má Tomcat i své neduhy, kterými jsou: poměrně pomalé zpracování požadavků a slabší stabilita, ale na druhou stranu je

Open Source a je plně kompatibilní s J2EE(Java). Je možno použít i jiné servery, například server Jetty nebo Resin, které jsou také Open source.

Před instalací Tomcatu je potřeba nainstalovat *Java SE Development Kit* (JDK), který nainstaluje „Javovské prostředí“. Podobně jako v Php je potřeba přepsat určité cesty a parametry v určitých souborech. (viz návody na internetu)

3.1.4 Perl

Pokud vlastní uživatel systém linux, tak bývá Perl většinou jeho součástí. Instalace pod Windows není také těžká. Stačí stáhnout Aktive Perl pro Windows (http://activestate.com/store/activeperl/?id=ActivePerl) a nainstalovat. Opět je možné provozovat Perl pomocí Apache nebo IIS. Jako databázový zdroj je možné zvolit, kteroukoliv databázi, pro kterou existují databázové ovladače(rozhraní).

3.1.5 Shrnutí

Jako nejjednodušší instalace mi připadá instalace ASP.NET, kde uživatel v průvodci zvolí, co chce a nechce a není nucen v nějakých souborech cosi přepisovat. Výhodou je, že je databázový server implicitně propojen s ostatními součástmi a není potřeba ho nastavovat.

Instalace Php není také většinou problémová, protože na internetu je tolik návodů, že programátor brzy dokáže najít chybu při své instalaci (když k ní dojde).

U JSP a Perlu je situace a něco horší, protože na internetu není tolik informací jako například o Php => případný problém při instalaci se hůře objevuje. JSP navíc vyžaduje instalaci tzv. kontejneru.

3.2 SPOLUPRÁCE S DATABÁZEMI

Komunikace s databází narůstá stále více na důležitosti, zatímco dříve se internet používal spíše pro prohlížení dokumentů a obrázků, tak dnes se přes internet obchoduje, nakupuje a ukládá spousta informací, proto je na místě, se zajímat, s jakou databází bude náš webový výtvor spolupracovat.

Teď je tedy otázka, jak která technologie spolupracuje s kterou databází. To se dozvíme dále. Všechny čtyři technologie sice podporují práci s databází přes rozhraní ODBC, což je dobrá zpráva, ale určitě to není to nejlepší rozhraní, které existuje i když je velmi universální. Proto je lépe zjistit jaké rozhraní mají jednotlivé technologie pro dané databáze a vybrat si technologii a databázi, které jsou pro sebe „jak zrozené“. Také je dobré srovnat, jak jednotlivé technologie s těmito databázemi komunikují.

3.2.1 Php

Php pro databáze nabízí speciální knihovny. Základní distribuce tak přímo podporuje databáze jako MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Firebird/Interbase atd. Samozřejmě zde najdeme také modul pro ODBC a podporován je i zajímavý projekt SQLite. A pro ukázku a srovnání s ostatními je zde typický kód pro připojení k MySQL:

```
<?Php
/* Spojení s databází my_database */
$spojeni = mysql_connect("mysql_host", "mysql_user",
"mysql_password")
or die ("Spojení nelze navázat");
echo ("Spojení úspěšně navázáno");
mysql_select_db("my_database") or die("Nenalezena vybraná
databáze");

/* Dotaz do databáze */
$dotaz = "SELECT * FROM my_table";
$vysledek = mysql_query($query) or die("Chyba v dotazu");

/* Výpis výsledku  */
echo ("<table>\n");
while ($radek = mysql_fetch_array($vysledek, MYSQL_ASSOC))
{
echo ("\t<tr>\n");
foreach ($radek as $sl_hod) {
echo ("\t\t<td>$sl_hod</td>\n");
}
echo ("\t</tr>\n");
}
echo ("</table>\n");

/* Uzavření spojení */
mysql_close($spojení);
?>
```

Zdrojový kód 1: Připojení do databáze MySQL

Ze Zdrojový kód 1 jsou vidět dvě důležité věci: manipulace s databází probíhá strukturovaným voláním funkcí, přičemž reakcí na chybu je příkaz die (ukončení skriptu; syntaxe „připoj se or die“ je ekvivalentem „if (nepodařilo se připojit) then die“).

3.2.2 ASP.NET

ASP.NET k datovým zdrojům přistupuje zcela jinak, a sice přes technologii ADO.NET (fyzicky je reprezentována třídami ve jmenném prostoru `System.Data`). Otázkou je, k čemu další databázové API, když zde máme ODBC, OLE DB, DAO, ADO, RDO a další. Zjednodušeně proto, že žádná ze stávajících technologií není příliš dobře připravena na bezstavovou éru protokolu HTTP. Mezi další důležité cíle

návrhu patří vylepšená podpora XML a snaha oddělit připojení k datům od manipulace s nimi. Asi nejlepší bude malá ukázka.

Nejdříve situace, kdy máme kontakty uloženy v následujícím XML souboru (Kontakty.xml):

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<kontakty>
<kontakt>
<jmeno>Pavel Matlásek</jmeno>
<email>matlas@nekde.com</email>
</kontakt>
<kontakt>
<jmeno>Martina Králová</jmeno>
<email>martinka@emmail.cz</email>
</kontakt>
</kontakty>
```

Do souboru aspx umístíme následující tag:

```
...
<form runat="server">
<asp:DataGrid id="MyDataGrid" runat="server" />
</form>
...
```

DataGrid je serverový ovládací prvek datové mřížky, hodí se tudíž k vypsání 2D dat. Zbývá už jen doplnit kousek kódu do události `Page_Load` (ta se automaticky spouští při načtení stránky).

```
private void Page_Load(object sender, EventArgs e)
{
DataSet ds = new DataSet();
ds.ReadXml(Server.MapPath("Kontakty.xml"));
MyDataGrid.DataSource = ds;
MyDataGrid.DataBind();
}
```

Zdrojový kód 2: Ukázka DataGrid v ASP.NET

Nejdříve je vytvořen objekt DataSet. To je v ADO.NET naprosto klíčový objekt, který si lze představit jako in-memory databázi. Může obsahovat tabulky, vazby mezi tabulkami, integritní omezení apod. Pomocí funkce `ReadXml` do něj načteme celý soubor, datové mřížce určíme, že má jako zdroj dat používat právě naplněný DataSet a poslední příkaz už je jen příkazem vynucujícím svázání dat

definované o řádek výše. Nikde se tedy nevyskytuje žádný kód, který by iteroval přes všechny řádky a sloupce výsledku.

Druhá verze tohoto příkladu bude využívat Microsoft SQL Server 2000 (tento databázový server je samozřejmě upřednostňovaný, nicméně ADO.NET podporuje širokou škálu jiných databází, např. Oracle, MySQL a mnoho dalších, pro které existuje nativní nebo OLE DB datový poskytovatel). ADO.NET sice stále podporuje tzv. data readers, což jsou v podstatě serverové kurzory podobné těm v Php, náš příklad však využije novější a pokročilejší techniku využívající tzv. datové adaptéry (data adapters). Ti jsou prostředníkem mezi databází a objektem DataSet a velmi zjednodušeně řečeno fungují tak, že je DataSet využívá k provádění nízkoúrovňových příkazů. Např. při aktualizaci dat v databázi DataSet pouze požádá o aktualizaci a je na datovém adaptéru, aby vykonal patřičný SQL příkaz.

Kód stránky samotné (soubor aspx) zůstává nezměněn, mění se pouze kód metody `Page_Load`:

```
SqlDataAdapter adapter = new SqlDataAdapter ("select * from
kontakt",
"server=localhost;database=test;uid=uzivatel;pwd=neprustrelneheslo"
);
DataSet ds = new DataSet();
adapter.Fill(ds);
MyDataGrid.DataSource = ds;
MyDataGrid.DataBind();
```

Tato verze se tedy od té předchozí liší jen velmi málo, ačkoliv předtím byla data získávána ze souboru na disku, zatímco teď z relační databáze. Tolik malá ochutnávka toho, jakou abstrakci ADO.NET nabízí a jakým způsobem jsou data prezentována uživateli (žádné iterace, používá se model vázání dat k ovládacím prvkům).

ADO.NET je poměrně šikovným API, jehož podstatnou vlastností je mimo jiné podpora přístupu k datům v nespojitém prostředí. Nejnázorněji to je vidět při tvorbě obyčejného tlustého klienta (i tam se používají stejné objekty DataAdapter a DataSet), kde dojde k jednorázovému načtení dat do objektu DataSet, uživatel se pak klidně může odpojit, hodinu provádět úpravu dat (přidávání, mazání nebo modifikaci

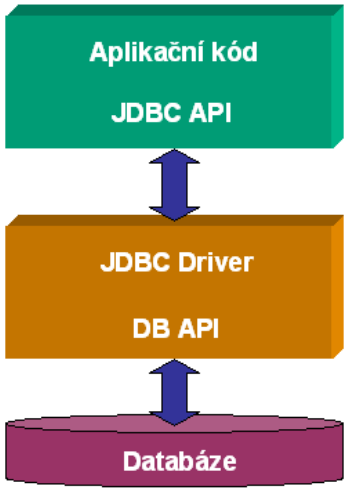
záznamů) a nakonec vše zase jednorázově aktualizovat v databázi. V ASP.NET to už tak názorné není, nicméně i zde tento princip funguje. Už na naší krátké ukázce je vidět, že komunikace s databází se odehraje v rámci jednoho příkazu: `adapter.Fill(ds)`. Ten spojení otevře, získá data a bez okolků zase uzavře. Až potom je s daty dále manipulováno.

3.2.3 JSP

Co se týče práce s databází o operace nad nimi je JSP více podobno Php, protože manipulace s databází probíhá strukturovaným voláním metod jako u Php (viz ukázka níže).

JSP a obecně i JAVA (J2EE) používá pro práci s databází ovladač JDBC (Java Database Connectivity). JDBC API poskytuje základní rozhraní pro unifikovaný přístup k databázím. Základem konceptu JDBC je využití funkčnosti poskytované JDBC ovladačem, který je následně překládá do nativních volání dané databáze. Díky tomu je aplikační programátor odstíněn od specifického API databáze a může se naučit jednotné rozhraní JDBC, které pak použije pro přístup do libovolné databáze, která poskytuje JDBC ovladač. V dnešní době to jsou prakticky všechny hlavní systémy a ovladače optimalizované a vyvíjené samotnými výrobci databázových strojů.

JDBC navíc není určeno pouze pro přístup k relačním databázím, ale k libovolnému formátu dat, ukládaného ve „sloupcové podobě“, což mohou být i datové soubory „spreadsheetů“ a textové soubory.



Obrázek 1: Zjednodušená architektura JDBC

Přestože je schéma na Obrázek 1 zjednodušené, naznačuje základní princip JDBC, jak jej vnímá aplikační programátor. Logika JDBC je ale složitější, v závislosti na vlastnostech JDBC ovladače. JDBC specifikace rozpoznává čtyři typy JDBC ovladačů (viz dokumentace).

JDBC ovladač je specifická třída, obvykle poskytována výrobcem databáze. Předtím, než budou provedeny jakékoli operace nad databází, je nutné ovladač nahrát a registrovat. K tomu slouží příkaz *Class.forName("jméno_JDBC_ovladače")*;. Příkladem může být „JDBC-ODBC bridge“, jehož ovladač se registruje příkazem *Class.forName("sun.jdbc.odbc.JdbcOdbcDriver")*;.

Jakmile byl JDBC ovladač nahrán a správně zaregistrován DriverManagerem, je možné navázat spojení s databází. V případě, že jsme autorizováni k získání připojení a vše je nastaveno správně, metoda *getConnection()* vrátí požadované spojení, které je základem pro další práci s databází a přes které se provádí všechny operace nad databází.

Jakmile máme spojení s databází, můžeme nad ní začít provádět datové operace. K tomu je určena další třída, kterou nám poskytne vytvořený objekt *Connection*, tou je *Statement* (třída implementuje rozhraní *java.sql.Statement* popsané v dokumentaci JDK). Instance této třídy pak slouží pro posílání SQL

příkazů databázi, kde jsou zpracovány a vrací odpovídající hodnoty. Statement nabízí velké množství metod (viz již zmíněná dokumentace). Oproti Php už nemáme univerzální metodu pro dotazy jako mysql_query, ale je zde rozdělena na dvě základní. První je *executeQuery(),* která slouží pro provádění SELECT a druhá metoda, která aktualizuje databázi *executeUpdate(),* která aktualizuje databázi příkazy UPDATE, INSERT a DELETE.

A abychom nebyli ani u JSP připraveni o jednoduchý příklad navázání spojení s databází a jednoduchého dotazu do databáze, zde je (Zdrojový kód 3):

```
<%
/* Nahrání driveru */
Class.forName("org.gjt.mm.mysql.Driver").newInstance();

/* Připojíme se k databázi. */
Connection connection = DriverManager.getConnection("jdbc:mysql://
localhost/test?useUnicode=true&characterEncoding=iso-8859-2",
"user", "password");

/* Statement nám umožňuje odesílat sql
dotaz a získat odpoveď. */
Statement stmt = connection.createStatement();

/* Dotaz SELECT do databáze */
ResultSet rs = stmt.executeQuery("SELECT * FROM employee");

/* Výpis dotazu SELECT */
while(rs.next()) { %>
<tr><td><%= rs.getString(0) %></td><td><%=
rs.getString(1) %></td>
<td><%= rs.getString(2) %></td><td><%=
rs.getString(3) %></td></tr><%="\n"%>
<% }

/* Uzavření všech otevřených objektů. */
rs.close();
stmt.close();
conn.close();
%>
```

Zdrojový kód 3: Navázání spojení u JSP

3.2.4 Perl

Perl je opět velmi podobný Php v přístupu k databázi. Perl pro práci s databází používá rozhraní DBI (Database Interface). Jde o modul a specifikaci poskytující sadu metod pro připojení k databázovému systému, zadání příkazu či dotazu a získání výsledku spolu s transparentním mapováním datových typů mezi Perlem a jednotlivými databázovými implementacemi. Cílem je mít stejný Perlový kód pro stejné akce nad různými databázemi, například pro odeslání dotazu či načtení jednoho záznamu výsledku. Na druhou stranu se DBI nesnaží skrývat a smazávat rozdíly v SQL syntaxích, datových typech, rozšířeních či nedostatcích jednotlivých systémů.

DBI se pak pokusí při volání *connect* (viz příklad níže) natáhnout modul s driverem odpovídajícím dané databázi, v tomto případě *DBD::Oracle* a použít ho pro vytvoření databázového spojení. Pro databázi PostgreSQL slouží driver *DBD::Pg* a za druhou dvojtečkou pak následuje jméno databáze, případně jméno stroje ve formátu *dbi:Pg:host=ax241;dbname=telis*, pro MySQL driver *DBD::mysql*, kde pokračování popisu databázového zdroje je hodně podobné *DBD::Pg*, pro přístup k MS SQL doporučuji *DBD::Sybase* s knihovnou *FreeTDS*, případně *DBD::ODBC*. Je vidět, že i Perl svou cestou dovoluje připojení na poměrně dost databází.

I pro Perl je zde krátký ukázkový příklad (Zdrojový kód 4)spolupráce s databází:

```
/* Navázání spojení */
use DBI;
my $dbh = DBI->connect('dbi:Oracle:prod9i', 'user', 'pass',
        { AutoCommit => 0, RaiseError => 1 });

/* Dotaz select */
my $sth = $dbh->prepare('select name, street, city from tst2 where
name like ?');

/* Výpis záznamů */
$sth->execute('%' . $n . '%');
while (my $row = $sth->fetchrow_arrayref) {
        # v $row je reference na tříprvkové pole
        my $street = $row->[1];
        # ... další zpracování
}
```

Zdrojový kód 4: Navázání spojení v Perlu

3.2.5 Shrnutí

Php, JSP a Perl pracují s databázemi velmi podobně pomocí podobné strategie (navázání spojení, dotazy na databázi, ukončení spojení a uvolnění zdrojů). Každá technologie má svoje specifické rozhraní či knihovny, pomocí kterých se připojují k daným databázím. JSP používá například JDBC, které bylo inspirováno ODBC. Perl používá zase DBI rozhraní. ASP.NET řeší práci s databází kapku netradičně, ale jde pouze o zvyk. Databáze je vytvořena jako objekt a při jednoduchý dotazech na databázi programátor není nucen psát ani kód. Při propojení technologií a databází je možné využít dost známý ODBC ovladač, který je velmi universální, ale na druhou stranu pokud ho programátor nemusí použít, tak je to lepší, protože je to poměrně "úzký klient", který může snížit výkon celé aplikace.

Nakonec tohoto shrnutí bych doporučil používat ke každé technologie databázi pro tuto technologii "vžitou", která je už ověřená a programátor se už nemusí zabývat tím, proč něco nefunguje i když by mělo.

3.3 HISTORIE

3.3.1 Php

Tedy ještě jednou, Php je skriptovací jazyk pro tvorbu dynamického webu a jeho počátky spadají do roku 1994. Tehdy se Rasmus Lerdorf rozhodl vytvořit

jednoduchý systém pro počítání přístupu ke svým stránkám. To bylo napsáno v jazyce Perl. Za nějakou dobu byl systém přepsán do jazyka C, protože „Perlovský“ kód hodně zatěžoval server. Sada těchto skriptů byla ještě později téhož roku vydána pod názvem "Personal Home Page Tools", zkráceně Php.

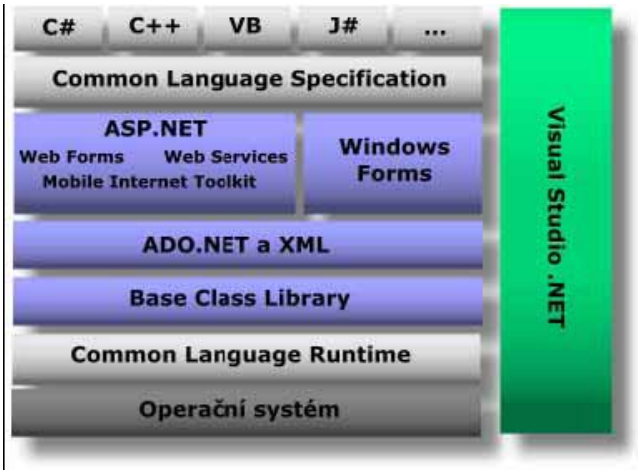
U toho však nezůstalo. V polovině roku 1995 se systém Php spojil s jiným programem stejného autora, a to sice s nástrojem "Form Interpreter" neboli zkráceně FI. Tak vzniklo Php/FI 2.0, systém, který si postupně získal celosvětovou proslulost a byl velmi rozšířen.

Koncem roku 1998 byla již k dispozici verze Php 3.0, která byla mnohem rychlejší a vybavenější než Php 2.0 a která byla k dispozici rovněž pod operačními systémy Windows. Počet webů které používaly Php se zvyšoval, až dosáhl cca 150 000. Verze Php 4.0 a 5.0, které jsou k dispozici nyní, přidávají do jazyka mnoho nových funkcí a rovněž přinášejí přepracované a tudíž podstatně rychlejší jádro Zend. Podle údajů z dubna 2004 běží Php na více než 15 000 000 doménách a je to bezkonkurenčně nejčastěji používaný modul webového serveru Apache.

3.3.2 ASP.NET

ASP.NET je nejnovější technologií firmy Microsoft určenou pro vývoj webových aplikací. První verze ASP (Active Server Pages) byla vydána na konci roku 1996 a kromě podobného data vydání toho měla s Php společného opravdu hodně. Koneckonců, Php i ASP spadají do stejné kategorie, zde pracovně nazvané jako skriptovací technologie.

Koncem devadesátých let minulého století začal Microsoft pracovat na zcela nové platformě, kterou pojmenoval .NET. Je to „sada softwarových technologií společnosti Microsoft pro propojování světa informací, lidí, systémů a zařízení“. Zní to příliš marketingově, ale v podstatě to je pravda, alespoň co se záměru týče. Základem je platforma nazvaná .NET Framework, jejíž struktura je nejlépe vidět z obrázku.



Obrázek 2: Začlenění ASP.NET v .NET Frameworku

Z Obrázek 2 je velmi pěkně vidět, co to vlastně ASP.NET je. Není to nějaká samostatná technologie pro tvorbu webových aplikací, je integrální součástí nové platformy a staví na stejných základech jako její doplněk – Windows Forms (část frameworku pro tvorbu tlustých klientů, ne nutně pro operační systém Windows). ASP.NET tedy není jen další verzí ASP, ve skutečnosti se jedná o zcela novou technologii.

Obrázek 2 dále ukazuje tři základní stavební kameny ASP.NET: Web Forms, Web Services a Mobile Internet Toolkit. To všechno jsou velmi důležité věci, které dávají tušit sílu ASP.NET. Důležitou součástí je rovněž koncept serverových ovládacích prků, kterým se práce věnuje níže.

To, že jsou do ASP.NET integrovány webové služby, je velkým plusem. Jak bylo řečeno, celý .NET je inspirován snahou usnadnit vývoj distribuovaných systémů, a webové služby jsou jasnou cestou, kudy se pravděpodobně integrace heterogenních systémů bude ubírat.

ASP.NET nelze stáhnout a nainstalovat samostatně (vždy je součástí .NET Frameworku), proto jsou verze číslovány shodně s celým Frameworkem.

3.3.3 JSP

Stejně jako u ASP.NET i tady se nedá hovořit přímo o historii JPS, protože Java Server Pages je standardní rozšíření JAVy, které je definováno nad servlety. Z toho plynou stejné výhody jako servletů, ale přidává ještě navíc jednu podstatnou výhodu a to, že JSP se jednoduše vytváří a tak se stává údržba dynamických stránek snadnější a přehlednější než u servletů. JSP umožňují kombinovat Html s kousky kódů Javy, tento kód je uzavřen ve speciálním tagu, v jednom dokumentu. To ale na druhou stranu klade větší nároky na tvůrce, jelikož nalézt chybu v JSP je mnohem obtížnější než v klasickém JAVA programu.

3.3.4 Perl

Název Perl vznikl z akronymu Practical Extraction and Report Language. Autorem je Kanaďan Larry Wall. Napsal jej pro svou potřebu, protože mu chybělo něco jednoduššího než je C a co zároveň zvládne i větší požadavky. Potom v roce 1987 uvolnil Perl verze 1.0 pro veřejnost a sám se divil zájmu, který tím vyvolal. Proto se začal ještě dále vyvíjet. Z nástroje pro zpracování textu se povýšil na programovací jazyk s mnoha doplňky. Dnes se o vývoj Perlu stará skupina lidí v čele s Larrym Wallem. Poslední stabilní verzí je verze 6.0.

3.4 PODPOROVANÉ JAZYKY

Jelikož porovnáváme technologie, tak je vhodné vědět, co která technologie podporuje za jazyky a jestli má programátor možnost si vybrat, či ne. Taky srovnáme "kvalitu" těchto jazyků.

3.4.1 Php

Php je sice technologií pro vývoj webových aplikací, ale zároveň i skriptovacím jazykem této technologie, a to jazykem jediným. Je tady sice možnost integrace s Javou, nicméně nelze říct, že by Php obecně podporovalo více jazyků.

3.4.2 ASP.NET

Jelikož je ASP.NET nadstavbou NET Frameworku a celý .NET Framework je jazykově neutrální – lze používat libovolný programovací jazyk, který vyhovuje určitým pravidlům (Common Language Specification). ASP.NET obsahuje přímou podporu pro Visual Basic .NET, C# a JScript.NET, jinak je možno využívat z nabídky více než 25 jazyků což je opravdový komfort. Dokonce je teoreticky možné jako jazyk ASP.NET používat samotné Php.

Jazyková neutralita ASP.NET je nádherná vlastnost, která má mnoho příjemných důsledků. První je samozřejmě možnost volby. Vývojář ve Visual Basicu se nemusí učit nějaký nový jazyk, který je už od pohledu zcela odlišný, vývojář v Javě má okamžitě k dispozici J#, vývojář v C++ může svůj oblíbený jazyk okamžitě začít používat, prostě si každý vybere, co mu vyhovuje a co nejlépe ovládá.

Tím, že se zdrojový kód kompiluje do MSIL (Microsoft Intermediate Language), je jazyková neutralita dotažena k dokonalosti. Není nejmenší problém napsat některé třídy třeba v COBOLu, jiné v Pascalu a přitom využívat funkčnosti tříd napsaných v C#.

3.4.3 JSP

Jelikož technologie JSP vychází z JAVY je jasné, že hlavní podporovaným jazykem je právě JAVA. Programátor v tomto případě nemá možnost volby.

3.4.4 Perl

Perl není tak ani technologie jako taká, ale spíše "pouhý" programovací jazyk vybavený tak, že zvládá úkony, podobné, jako ostatní porovnávané technologie. Z první věty tedy vyplývá, že programátoři Perlu programují v jazyku Perl.

3.4.5 Srovnání

V této kapitole je srovnání velmi jednoduché a poměrně jasné. Kromě technologie ASP.NET umožňují ostatní technologie pouze jeden programovací

jazyk, takže programátor se musí tento jazyk naučit, jinak má smůlu. ASP.NET vyhrálo tuto kategorii, protože staví na pevném základu, kterým je .NET Framework.

3.5 PODPORA OOP

Zkratka OOP znamená objektově orientované programování. Tato podkapitola bude napsána netradičně, protože bude rozdělena na jednotlivé myšlenky OOP a tam popsány všechny čtyři technologie. Metodika OOP je založená na následujících myšlenkách a koncepci:

3.5.1 Úroveň podpory OOP technologiemi (jazyky)

První důležitou věcí je, že všechny čtyři technologie, respektive jazyky, podporují OOP. Budu zde používat pojmu jazyky místo technologie, protože OOP se týká právě programovacích jazyků.

Jazyk Java je založen na jazyku C++, ale je mnohem více objektově orientován. Lze prohlásit, že se jedná o čistokrevný objektově orientovaný programovací jazyk. V Javě je prakticky vše objektem, tento jednotný přístup umožňuje jednodušší osvojení základů jazyka a jeho používání. ASP.NET nejčastěji používá k programování jazyky Visual Basic, C# a tou jsou též jazyky, které podporují OOP, ale už nelze říci, že by byly "čistokrevné OOP jazyky". To samé platí o Php. Jazyk Perl OOP sice podporuje, ale je na tom s podporou ještě o něco "hůře" než třeba Php, protože některé procedury v Perlu se neshodují s myšlenkou OOP programování.

3.5.2 Objekty

Jednotlivé prvky modelované reality (jak data, tak související funkčnost) jsou v programu seskupeny do entit, nazývaných objekty. Objekt si „pamatuje“ svůj stav (v podobě dat čili atributů) a poskytuje rozhraní operací, aby se s ním mohlo pracovat (nazývané metody). Při používání objektu nás zajímá, jaké operace poskytuje, ale ne, jakým způsobem to provádí - to je princip zapouzdření. Jestli to

provádí sám nebo využije služeb jiných objektů, je celkem jedno. Vlastní implementaci pak můžeme změnit, aniž by se to dotklo všech, kteří objekt používají.

Abstrakce objektu, která v architektuře programu podchycuje na obecné úrovni podstatu všech objektů podobného typu, se nazývá třída. Třída je předpis, jak vyrobit objekt daného typu.

3.5.3 Zapouzdření

Zapouzdření je základním principem objektově orientovaného programování. Objekt zveřejňuje pouze rozhraní pro komunikaci s ním, zatímco vnitřní implementace zůstává skrytá a nedostupná.

Všechny .NET jazyky, Php, JSP i Perl podporují vytváření tříd a objektů. Syntaxe je poměrně podobná.

Jazyk C# podporuje následující modifikátory přístupu: public, protected internal, protected, internal a private (podle stupně omezení). Public přístup neomezuje, protected internal znamená protected or internal, protected znamená omezení na danou třídu a její potomky, internal omezení na danou assembly a private pouze na danou třídu. Jak je vidět, C# (celý .NET Framework) podporuje celou řadu modifikátorů přístupu, navíc princip zapouzdření podporuje tím, že pokud není žádný modifikátor u člena třídy uveden, je implicitně private.

Co se týče Php, syntaxe vytváření tříd je podporována už od verze 3. Až v nové verzi 5.0 se ale objevují modifikátory přístupu public, protected a private, které musí být před proměnnou uvedeny. Význam je prakticky stejný jako v případě jazyka C# (modifikátor ala internal neexistuje, protože Php nezná nic jako nějaké programové celky, komponenty). Malým problémem je zpětná kompatibilita. Jelikož modifikátory přístupu byly přidány až v páté verzi, je nutno se nějak vypořádat s klíčovým slovem var, které bylo dříve používáno pro označení členské proměnné třídy. Aby většina aplikací po přechodu z Php 4 na Php 5 fungovala, klíčové slovo var automaticky znamená ekvivalent modifikátoru public, což jde trochu proti principu zapouzdření.

Jak už bylo řečeno výše, Java je čistokrevně OOP orientovaný jazyk. Podporuje stejně jako Php tři modifikátory přístupu public, protecte a private.

Perl je v tomto pohledu hříšníkem, který jde proti OOP, protože narozdíl od jiných jazyků umožňuje přistupovat k atributům třídy z programu i bez přístupové metody. Což je proti myšlence objektově orientovaného programování a nemělo by se tohoto přístupu využívat.

Závěrem tedy můžeme říct, že C# má o dva modifikátory přístupu navíc (internal a protected internal) oproti Php a Javě, protože Php ani Java nejsou komponentové technologie, proto je ani mít nemůžou a u Perl prakticky nemůžeme ani mluvit o klasickém zapouzdření.

3.5.4 Dědičnost

Dědičnost je schopnost definovat určité třídy jako předky a jiné jako potomky, přičemž potomci po svých rodičích dědí jak členské proměnné, tak metody.

Php, jazyk C# a Java jsou ohledně dědičnosti prakticky totožné. Prvním významným společným rysem je podpora pouze jednoduché dědičnosti oproti jazykům Perl a C++, které podporují vícenásobnou dědičnost, která má své příznivce i odpůrce. Jednoduchá dědičnost je v současnosti jednoznačným trendem. Java navíc uvedla ve známost takzvaná rozhraní, která mají vícenásobnou dědičnost zastoupit. Rozhraní je jakýsi předpis třídy, který určuje, jaké metody má mít třída, která bude toto rozhraní implementovat. Daná třída pak může implementovat libovolný počet rozhraní. Syntaxe i chování v Javě, v Php i v C# se prakticky neliší.

Php zná klíčové slovo final, které je ekvivalentem klíčového slova sealed v jazyce C#. Od takto označené třídy již nelze dědit.

Všechny čtyři jazyky rovněž znají koncept abstraktních tříd. Hlavním znakem abstraktních tříd je, že nemohou být instanciovány a jejich hlavním a víceméně jediným účelem je „sloužit“ jako třídy předků. Mohou (ale nemusejí) obsahovat abstraktní metody, což jsou metody bez implementace. Potomek potom tyto metody musí buďto implementovat, nebo musí být sám abstraktní třídou.

V podpoře dědičnosti se C# a Php neliší prakticky vůbec. U Javy a Perlu je navíc podpora vícenásobné dědičnosti i když Java ji nepodporuje přímo, ale pomocí tzv. rozhraní.

3.5.5 Přetěžování

Co to je přetěžování (overloading) si ukážeme na příkladu. Jde o to, že např. pro sečtení dvou čísel nemusíme mít dvě metody sectiCelaCisla(int a, int b) a sectiDesetinnaCisla(double a, double b), ale prostě vytvoříme dvě metody secti, přičemž první bude přijímat parametry typu int a druhá typu double.

Přesně takto funguje C#, Java i Perl, které přetěžování plně podporují, zatímco Php ho kvůli své dynamicky typované podstatě nepodporuje, respektive nepotřebuje. V jistých případech lze podobné chování velmi jednoduše nasimulovat.

3.5.6 Výjimky

Výjimku chápeme jako důsledek výjimečné, neočekávané či chybové události v rámci vykonávání programu. Je trochu zavádějící uvažovat o výjimkách pouze v intencích chyb. Umožňují tedy reagovat na chybové stavy velmi přehledným a konzistentním způsobem, navíc s využitím objektového přístupu (výjimky jsou objekty).

Php, C# i Java používají prakticky stejný zápis – známou strukturu try/catch (Chráněný blok je uvozen příkazem try a uzavřen mezi složené závorky. Jakoukoli výjimku vzniklou v chráněném bloku je možné ošetřit. K ošetření se používá příkaz catch, za nímž následuje konkrétní datový typ výjimky). U Php nenajdeme blok finally. Všechny tři jazyky definují třídu s názvem Exception, která je předkem všem ostatním výjimkám. ASP.NET to přísně vyžaduje, v Php je to naopak pouze doporučený postup, lze si ale vytvořit vlastní hierarchii výjimek, která nebude mít s třídou Exception nic společného.

- *Skládání* – Objekt může využívat služeb jiných objektů tak, že je požádá o provedení operace.
- *Polymorfismus* – odkazovaný objekt se chová podle toho, jaký je jeho skutečný typ. Pokud několik objektů poskytuje stejné rozhraní, pracuje se s nimi stejným způsobem, ale jejich konkrétní chování se liší. V praxi se tato

vlastnost projevuje např. tak, že na místo, kde je očekávána instance nějaké třídy, můžeme dosadit i instanci libovolné její podtřídy, která se může chovat jinak, než by se chovala instance rodičovské třídy, ovšem v rámci mantinelů, daných popisem rozhraní.

3.6 PODPORA XML

XML (eXtensible Markup Language, česky rozšiřitelný značkovací jazyk) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Umožňuje snadné vytváření konkrétních značkovacích jazyků pro různé účely a široké spektrum různých typů dat.

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Jazyk umožňuje popsat strukturu dokumentu z hlediska věcného obsahu jednotlivých částí, nezabývá se sám o sobě vzhledem dokumentu nebo jeho částí.

Perl podporu XML plně podporuje. Síla Perlu spočívá v tom, že můžeme snadno mixovat elementární postupy, jako jsou regulární výrazy a manipulace s textem, s moduly, které pracují na poměrně vysoké úrovni. Z toho plyne, že máme poměrně velkou kontrolu nad tím, co se děje, na druhou stranu někdo zase radši komfort.

Tento komfort nám nabízí celý .NET, který je na distribuované systémy zaměřen, není tedy divu, že na XML nebylo zapomenuto. Právě naopak, celý jmenný prostor System.Xml obsahuje velké množství tříd pro poměrně komfortní práci s XML. Podporovány jsou standardy XML 1.0 včetně DTD, XML jmenné prostory, XML schémata, výrazy XPath, XSLT transformace a DOM level 1 a 2. Také pokud se podíváme i na další věci, jako konkrétní jmenné prostory a třídy, lze říci, že celý .NET Framework se s XML velmi kamarádí. Příkladem je integrace XML webových služeb, zápis komentářů v XML, různé konfigurační soubory v XML apod.

Php ve verzi 5 výrazně předělalo podporu XML, možná by se dalo říct, že XML konečně začalo plně podporovat. Verze 4 uměla ke XML dokumentům přistupovat pouze přes rozhraní SAX (Simple API for XML), což je sice efektivní, ale málo komfortní. Podpora DOM zůstala až do posledního vydání čtyřkové verze

označena za experimentální a určitě právem, protože několikrát změnila své API a obsahovala chyby.

XML je nepostradatelnou součástí Javy pro zpřístupnění B2B technologie. Nové API pro XML Parsing JAXP umožňuje čtení, manipulaci a generování XML dokumentů přes API JAVY. JAXP podporuje nejnovější standardy XML včetně DOM SAX a XSLT. JSP technologie zahrnují zlepšení kompatibility pro XML ve vícevrstvé architektuře.

Závěrem lze krátce říci, že všechny technologie podporují, dnes velmi rychle se rozšiřující a hojně využívané, XML. Problémy může mít snad jen Php verze 4.0.

3.7 MOŽNOSTI A OBTÍŽNOST UŽIVATELSKÝCH NASTAVENÍ

Možnosti nastavení a konfigurace jednotlivých technologií jsou důležité hlavně pro komfort programátorů a administrátorů. Komu by se taky chtělo něco složitě dva dny nastavovat. Většinou je upřednostňováno grafické prostředí pro nastavování před textovým, kde se přepisují a dopisují příkazy, aby nám fungovali různé moduly.

Pro nastavení Php se používá konfigurace souboru Php.ini, kde si většinou programátor musí na internetu dohledat, co a kde má změnit, aby mu fungovalo to a to. Uživatelsky to není moc příjemné nastavování, ale na internetu je k tomuto tématu tolik informací, že se člověk nemůže prakticky "ztratit".

ASP.NET má podle mé maličkosti konfigurovatelnost vyřešenu velmi dobře. Vše je ukládáno v XML souborech, přičemž ASP.NET zná dva druhy souborů:

- *Machine.config* nastavení pro celý web server
- *Web.config* což je nastavení pro určitou část aplikace.

Soubor Web.config je platný pro skripty v aktuálním adresáři a pro všechny adresáře podřízené. Pokud se v nějakém adresáři vyskytne další soubor Web.config, zdědí všechna nastavení od svého rodiče a má možnost další vlastnosti přidat nebo stávající modifikovat či překrýt. Protože by bylo poměrně složité při každém požadavku rekonstruovat celý hierarchický strom souborů Web.config, aby se

zjistilo, jaké vlastně nastavení platí pro aktuální stránku, je po prvním vyřešení vše cachováno a při příštím požadavku ihned k dispozici. Dojde-li však ke změně konfigurace, ASP.NET znovu celou hierarchii zkontroluje a aplikuje změny. Konfigurační soubory jsou pochopitelně rozšiřitelné (XML = eXtensible ML).

Perl jako takový je "doma" v systému linux a tam má už všechny potřebné nástroje a komponenty nakonfigurovány a programátor může hned začít pracovat. Konfigurace se provádí jen v případě kdy musí komunikovat nebo spolupracovat s pro něj nestandardním nástrojem (třeba IIS).

JSP se samo o sobě nenastavuje, ale kromě serveru potřebuje ke své práci ještě kontejner, který by zpracoval JSP dotazy. Většinou se používá serveru Apache ve spojení s kontejnerem Tomcat. Kontejner Tomcat je potřeba nakonfigurovat pomocí několika souborů. Soubor **server.xml** obsahuje kompletní nastavení aplikace Tomcat. V souboru **tomcat-users.xml** se nastavují uživatelé a přiřazují se jim práva k jednotlivým aplikacím. Po nainstalování je zde třeba vytvořit uživatele pro aplikaci manager (V roles musí mít manager), který pak může spravovat servlety pomocí webového rozhraní. V souboru **web.xml** se nastavují inicializační hodnoty pro všechny webové aplikace. Servlety jsou pak schopny tyto inicializační hodnoty číst a využít.

Uživatelsky nejpříjemnější se mi jeví v možnostech nastavení a konfigurovatelnosti technologie ASP.NET. Php má sice možnosti nastavení, ale pouze v "textovém režimu", který je uživatelsky méně příjemný. Perl a JSP sami o sobě nenabízejí prakticky žádná uživatelská nastavení, takže pracujeme s tím, co a jak je nastaveno.

3.8 BEZPEČNOST

Pojem bezpečnost je v dnešní době skloňována ve všech pádech a u informačních systémů to platí také. Dříve šlo především o funkčnost, ale dnes už jde bezpečnost a funkčnost ruku v ruce. Čím významnější projekt, tím vyšší je kladen důraz na bezpečnost.

Je znát, že si vývojáři technologií perfektně uvědomují význam bezpečnosti a proto můžeme říci, že všechny technologie jsou poměrně bezpečné i když malinké rozdíly tu jsou. Všechny podporují autentizaci / autorizaci (ASP.NET opět poměrně elegantně pomocí souborů Machine.config nebo Web.config, tedy deklarativně), nabízejí funkce pro bezpečné zapisování řetězců do databází (Php např. obsahuje tzv. magic quotes, což je sada funkcí a konfiguračních direktiv, které umožňují automaticky řešit časté problémy s podvrženými SQL vstupy). Zákonitě však zůstanou problémy, které nelze na aplikační úrovni řešit, třeba problém bezpečnosti cookies, sessions apod. Zde musejí přijít ke slovu specializované technologie typu SSL.

Druhou bezpečnostní otázkou je bezpečnost platformy. Co se týče operačního systému, výrazně lepší reputaci mají systémy Unix a Linux, zatímco Windows svou bezpečností rozhodně neprosluly. Microsoft sice v rámci své politiky „trustworthy computing“ dělá co může, nálepku bezpečného systému ale asi jen tak nezíská.

Podobně je to s webovými servery. Apache je uznáván jako bezpečný a kvalitní server, zatímco IIS za bezpečný považován není. Uznávaným bezpečnostním webem je securityfocus.com, který pro IIS 6.0 registruje jedinou nalezenou chybu, zatímco pro Apache 2.0 27. Jiný zdroj, dospívá ke třem chybám IIS – rovněž tedy velmi slušné číslo.

Důležitý je však rozdíl v opravách chyb. Zatímco v Open Source řešeních může být chyba ihned po jejím zveřejnění opravena, a to kýmkoliv, Microsoft vždy vydává opravy s určitým zpožděním. Další výhodou otevřených řešení je, že jsou slabá místa odhalována daleko dříve, protože je dostupný zdrojový kód. Bezpečnostní předpoklady mají tedy výrazně lepší Linux nebo Apache, ačkoliv se třeba aktuálně IIS 6 ukazuje jako bezpečný server.

3.9 VÝKONNOST

Výkonnost je v dnešní době velmi důležitá vlastnost, jelikož získání informací z internetu je dnes pracovní náplní mnoha lidí, proto je důležité, aby nebyli

uživatelé nuceni čekat dlouho na zobrazení stránky. Na velmi frekventovaných portálech je výkonnost asi jednou z nejdůležitějších parametrů.

Je těžké najít optimální podmínky pro porovnání těchto technologií, protože každá má své silné i slabé stránky. Nejspravedlivější by bylo porovnání na různých typech projektů a vyhodnocení, v kterých z nich byla jaká výkonnost. Taky nezáleží pouze na technologii, ale třeba i jako u bezpečnosti na systému, na které platformě běží a jaký server zpracovává její požadavky. Každá technologie by se určitě také jinak vypořádala s "jednoduchým" skriptem a velmi obsáhlým skriptem. Také se většinou skripty dotazují do databáze a ty by měli taky velký vliv na celkový výkon. Jasně a krátce lze říci, že nejde srovnávat výkon samostatných technologií, protože potřebují další "komponenty" ke svému běhu, které také ovlivňují jejich výkon. Poměrně velký dopad na výkon má fakt, jestli je kód kompilovaný nebo interpretovaný.

Kód kompilovaný se zákonitě vykonává rychleji než kód interpretovaný. ASP.NET, jako celý .NET, své skripty kompiluje do MSIL (Microsoft Intermediate Language), což je obdoba bytecode v Javě. Tento pseudokód je potom do strojového kódu kompilován metodou Just-In-Time, tedy při prvním požadavku. Vzniká tak kompilovaný skript, který následně ASP.NET cachuje, aby překlad z MSIL nemuselo provádět při každém požadavku. Na ASP.NET je pěkně vidět rozdíl mezi interpretovaným a kompilovaným kódem – při prvním požadavku na složitou stránku trvá její vygenerování na běžném počítači několik sekund (MSIL je potřeba interpretovat), zatímco druhý požadavek je vyřízen prakticky okamžitě. To je dáno také tím, že JIT kompilace se snaží o co nejoptimalizovanější překlad MSIL do strojového kódu, což má pozitivní výkonnostní důsledky.

Php je většinou označováno za interpretovaný jazyk, není to však tak úplně pravda. Také Php skripty jsou nejdříve kompilovány do něčeho, co se jmenuje opcode a v podstatě se jedná o ekvivalent MSIL. Tento opcode je potom interpretován pomocí Zend Engine. Jedná se tedy o jakýsi hybrid mezi kompilovaným a interpretovaným přístupem. Problémem Php je, že nedochází k žádné optimalizaci kódu ani ke cachování strojových kódů.

Java obsahuje JIT a velice agresivní metody optimalizace, které umí přestavět aplikaci za běhu. Je velmi podobná ASP.NET, kde se kód nejdříve kompiluje, proto první načtení stránky trvá déle.

Perl je, jako Php, jazyk interpretovaný proto je kód, při každém vstupu, znovu překládán, což není úplně ideální.

Lze říci, že z hlediska výkonnosti je na tom nejlépe ASP.NET a JSP, které používají kompilovaný skript a různé metody optimalizace kódu. Php používá vesměs hybrid mezi kompilovaným a interpretovaným jazykem a Perl se „spoléhá“ pouze na interpretovaný jazyk.

3.10 CENA

Tento parametr je pro některé tím nejhlavnější, zejména pro malé firmy, které počítají cenu za každou koupenou licenci. O technologiích Php a Perl lze říct, že jsou prakticky úplně zadarmo, protože pro psaní těchto skriptů se nechají použít nástroje, které jsou „free“. Samozřejmě jsou nástroje, které jsou zpoplatněny, ale to už je věc jiná. Když budu psát skripty v linuxu v textovém editoru a budu je vyhodnocovat serverem Apache nemusím zaplatit ani korunu.

Za ASP.NET a JSP je v současnosti nevyhnutelně nutné zaplatit alespoň cenu operačního systému, což v principu není u linuxu nutné, i když pokročilejší distribuce placené jsou. Také si myslím, že jednorázový výdaj za licenční poplatek není něco nepřekonatelného. Jelikož je potřeba použít většinou pokročilejší nástroje je potřeba s finančním vydáním též počítat.

3.11 ZHODNOCENÍ – TEORETICKÁ ČÁST

V celé kapitole 3 Vlastní srovnání jsem srovnával vesměs teoreticky dané technologie. Po přečtení je zřejmé, že některé technologie jsou vybaveny velmi dobře (jsou tzv. IN) a jiné hůře. Proto jsou v praktické části už porovnávány vesměs pouze tři (IN) technologie: Php, ASP.NET a JSP. Php, které je dnes mezi začínajícími programátory, i obecně, nejvíce rozšířeno a má obrovskou základnu informačních

zdrojů a také je zdarma. ASP.NET se svou inovativní filosofií a JSP se svou ověřeností praxí a také obrovskou programátorskou a informační základnou. To neznamená, že je Perl špatný, nebo že jsem ho zavrhl, jen pro něj nebyla naprogramována ukázková aplikace. Z jakých důvodů je tedy v praktické části Perl „diskriminován“:

- Mnohem menší budoucí perspektiva než ostatní technologie (málo programátorů se rozhodně právě pro Perl)
- Celkový úpadek tohoto jazyka -> málo kvalitních nástrojů pro tuto platformu, málo zdrojů a diskuzí
- Podobnost s Php (od Perlu se odvíjelo) a malá rozšířenost na hostinzích
- Špatná škálovatelnost a malá efektivita při větším vytížení serveru

V praktické části budou porovnány tedy všechny 4 technologie pro tvorbu informačních systému. Php, ASP.NET a JSP opírající se o literaturu, diskuze, články a jednoduchou aplikaci. Perl se opírá „pouze“ o literaturu, diskuze a články s ním související. Takže přikročme k praktickému srovnání.

4. PRAKTICKÁ ČÁST

V teoretické části jsem měl obecně porovnat jednotlivé technologie. V praktické části má být, podle zadání, vyzdvihnutí a demonstrování největších výhod, případně nevýhod jednotlivých platforem, na ukázkových příkladech.

Abych mohl demonstrovat tyto výhody a nevýhody, pokusil jsem se naprogramovat na „významnějších“ třech platformách stejnou aplikaci, abych pak mohl výsledek a složitost porovnat. Hlavním úkolem nebylo ani tak naprogramovat 3x precizní aplikaci, ale při programování na jednotlivých platformách si uvědomit jejich „příjemné vlastnosti“ a přednosti, případně jejich neduhy. Jelikož program, který bude popsán není moc komplikovaný, nejsou z něho vidět některé výhody některých platforem, proto budu využívat ještě další (nevlastní) ukázky příkladů.

4.1 TESTOVACÍ APLIKACE

Program, který jsem tvořil, má být systém pro registraci předmětů jednotlivých studentů. Tento projekt samozřejmě nezahrnuje pouhé programování na jednotlivých platformách, ale také je potřeba databázový nástroj, který by se staral o data. Po konzultaci jsem se rozhodl použít, pro všechny platformy kromě ASP.NET, databázi MySQL. ASP.NET se díky svému Frameworku chová jako jeden celek, proto jsem nechal původní databázový nástroj SQL Server Express 2005.

Platforma	Databázový nástroj
Php	MySQL
ASP.NET	SQL Server Express 2005
JSP	MySQL

Tabulka 1: Databázové nástroje použité pro jednotlivé platformy

Jelikož nebylo hlavním úkolem porovnat databázové nástroje, ale platformy, pokusil jsem se, aby byla závislost na databázovém nástroji co nejmenší a zvolil pro většinu platforem databázi MySQL. Můžou se objevit námitky, že to může být pro

některé platformy výhoda, což nepopírám, ale myslím, že v tomto směru se ideální řešení hledá velmi těžko.

4.1.1 Struktury tabulek aplikace

Jelikož je aplikace poměrně jednoduchá, tak k její funkčnosti bylo zapotřebí vytvoření pouze 5 tabulek.

Sloupec	Typ	Není nutné vyplnit	Klíč	Speciální
Id_predmet	INT(11)	NO	PRI	AI
Nazev	VARCHAR(50)	NO		
Zkratka	VARCHAR(10)	NO		
Garant	VARCHAR(50)	YES		
Ustav	VARCHAR(10)	NO		
Semestr	VARCHAR(20)	NO		
Kredity	SMALLINT(6)	NO		
Typ	VARCHAR(30)	NO		
Kapacita	INT(11)	NO		
Kapacita2	SMALLINT(6)	NO		

Tabulka 2: Struktura tabulky Předmět

V tabulce „Předmět“ jsou uloženy všechny údaje potřebné k uložení záznamů o daném předmětu. Jednotlivá pole nebudou popsána na tomto místě, ale bude na ně odkazováno na místě, kde popisuji, jak aplikace pracuje.

Sloupec	Typ	Není nutné vyplnit	Klíč	Speciální
Id_student	INT(11)	NO	PRI	AI
Jmeno	VARCHAR(40)	NO		
Prijmeni	VARCHAR(40)	NO		
Titul	VARCHAR(10)	YES		
Rodne_cislo	VARCHAR(12)	YES		
Fakulta	VARCHAR(10)	YES		

Obor	VARCHAR(10)	NO		
Prumer	DECIMA(10,2)	NO		
Login	VARCHAR(40)	NO		
Heslo	VARCHAR(40)	NO		
Opraveni	SMALLINT(6)	NO		

Tabulka 3: Struktura tabulky Student

Tato tabulka sdružuje informace o studentech, kteří si zapisují vybrané předměty. Jen upozorňuji, že jsem se nezabýval detaily, proto podle mých tabulek nemůže jeden student studovat na více fakultách. Musela by se tím zavádět další tabulka s relací 1:N, což pro náš účel není potřebné.

Sloupec	Typ	Není nutné vyplnit	Klíč	Speciální
Id_zapis	INT(11)	NO	PRI	AI
Id_student	INT(11)	NO		
Id_predmet	INT(11)	NO		
Stav	VARCHAR(30)	YES		
Datum	TIMESTAMP	NO		
Dr_kolo	SMALLINT(6)	YES		

Tabulka 4: Struktura tabulky Zápis

Jelikož musí být možné, aby si jeden studen mohl zapsat více předmětů a zároveň jeden předmět mohl být zapsán více studenty, bylo potřeba vytvořit relaci M:N, kterou zajišťuje právě tabulka „Zápis“. Další položky této tabulky budou opět vysvětleny níže, protože není na první pohled zřejmé k čemu přesně slouží.

Sloupec	Typ	Není nutné vyplnit
Zac_pr_kolo	DATETIME	YES
Kon_pr_kolo	DATETIME	YES
Zac_dr_kolo	DATETIME	YES

Kon_dr_kolo	DATETIME	YES

Tabulka 5: Struktura tabulky Nastavení

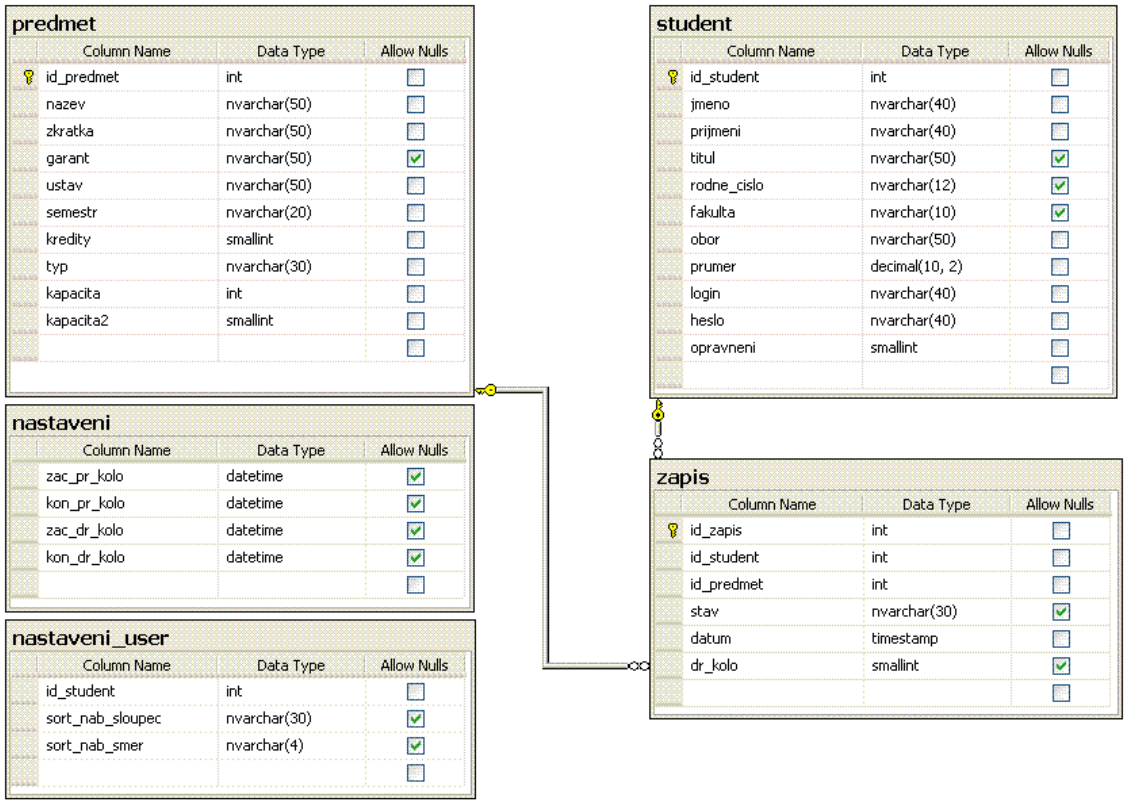
V tabulce „Nastavení“ musí pověřená osoba nastavit datum, kdy začíná a končí první a druhé kolo zápisu předmětů. Interval prvního a druhého kola se nesmí překrývat. Tato tabulka je pouze "klasická" tabulka bez jakýkoliv relací na další tabulky a slouží pouze k uložení dat, která jsou dále využívána.

Sloupec	Typ	Není nutné vyplnit	Klíč
Id_student	INT(11)	NO	
Sort_nab_sloupec	VARCHAR(30)	YES	
Sort_nab_smer	VARCHAR(4)	YES	

Tabulka 6: Struktura tabulky Nastavení_user

Tabulka „Nastavení_user“ by nemusela vůbec existovat, a přesto by aplikace fungovala, ale po konzultaci bylo rozhodnuto, že komfort není vůbec na škodu a uživatel si na něj rád zvyká. Proto byla vytvořena tato tabulka, která "si pamatuje" stav, ve kterém přihlášený uživatel zanechal tabulky a při příštím přihlášení pokračuje uživatel v počaté práci. Je to malá tabulka, protože v malé aplikaci není třeba ukládat příliš mnoho stavů, naopak ve velkém systému by mohla mít i desítky řádků.

Z tabulek výše byl vytvořen datový model, který je znázorněn na Obrázek 3. U jednotlivých technologií nebyl tento model totožný a vyskytovaly se zde drobné odlišnosti.



Obrázek 3: Datový model databáze zapis_predmetu

4.1.2 Testovací data

Abych se přiblížil reálnému modelu registrování předmětů studenty, nechal jsem se inspirovat registrováním předmětů, které je přístupné i mě: Zápis předmětů na VUT v Brně. Při testování bylo využíváno těchto dat z tabulky „Předmět“:

Povinné BMI	Zkr	Úst	Povin	Sem	K1	K2
Biologie člověka	BCL	bmi	p	zimni	2	1
Číslicová analýza a zpracování signálů	CZA	bmi	p	zimni	6	2
Úvod do medicínské informatiky	UMI	bmi	p	letni	2	1
Terapeutická a protetická technika	TPT	bmi	p	letni	6	2

Tabulka 7: Povinné předměty Bio-medicíny

Volitelné oborové BMI	Zkr	Úst	Povin	Sem	K1	K2
Multitaktní systémy	MUT	bmi	n	zimni	2	1
Didaktika	IPD	bmi	n	zimni	4	2
Počítače a programování	PC1	bmi	n	letni	2	1
Rentgeny	XRA	bmi	n	letni	4	2

Tabulka 8: Volitelné předměty Bio-medicíny

Povinné AMT	Zkr	Úst	Povin	Sem	K1	K2
Výpočetní technika v automatizaci	VTA	amt	p	zimni	2	1
Měření v elektrotechnice	MVE	amt	p	zimni	6	2
Praktické programování v C++	PPC	amt	p	letni	2	1
Signály a systémy	SAS	amt	p	letni	6	2

Tabulka 9: Povinné předměty Automatizace

Volitelné oborové AMT	Zkr	Úst	Povin	Sem	K1	K2
Měření fyzikálních veličin	MFV	amt	n	zimni	2	1
Mikroprocesory	MIC	amt	n	zimni	4	2
Modelování a simulace	MOD	amt	n	letni	2	1
Řízení a regulace	RR1	amt	n	letni	4	2

Tabulka 10: Volitelné předměty Automatizace

Všeobecně vzdělávací	Zkr	Úst	Povin	Sem	K1	K2
Podnikatelské minimum	POM		v	zimni	4	2
Němčina	JN1		v	letni	4	2
Angličtina	AN1		v	oba	6	2
Ruština	JR1		v	oba	2	1

Tabulka 11: Všeobecně vzdělávací předměty

Legenda k tabulkám Tabulka 7 až Tabulka 11:

Zkr ..... Zkratka předmětu

Úst ..... Ústav daného předmětu

Povin ..... Je předmět povinný/volitelný pro daný ústav
Sem..... Jaký semestr je předmět vyučován
K1 ..... Kapacita předmětu pro první kolo
K2 ..... Kapacita předmětu pro druhé kolo

Testovací studenti	Ústav	Vážený průměr
Lukáš Hubený	amt	2.18
Marek Doležal	amt	2.78
Karel Kyncl	amt	2.68
Karel Bureš	amt	2.03
Jaroslav Kyncl	amt	2.45
Michal Petr	amt	2.12
Marek Hubený	amt	2.33
Roman Slezák	bmi	2.21
Tomáš Smělý	bmi	2.04
Jaroslav Hlouš	bmi	2.56
Jiří Výborný	bmi	2.87
Martin Štumpf	bmi	1.56
Milan Srdinko	bmi	1.87
Karel Smid	bmi	2.14

Tabulka 12: Studenti pro testování

Tyto tabulky předmětů a studentů musely vzniknout proto, aby bylo ověřeno, že aplikace pracuje správně a automaticky. Kapacity předmětů v prvním i druhém kole (K1 a K2) jsou malé kvůli ověření správného fungování zápisu v druhém kole. Následující tabulka ukazuje, jak se chová systém v případě přihlášení studentů z různých oborů a jak studenti "vidí" jednotlivé předměty, které si mohou zapisovat.

Student-obor	Předmět-ustav	Předmět-typ	Výsledek
Amt	amt	p	povinný
Amt	amt	n	volitelný oborový
Amt	bmi	p	Nenabídne se studentovi
Amt	bmi	n	volitelný mimooborový
Amt	cokoliv	v	všeobecně vzdělávací

Tabulka 13: Zobrazení předmětů studentům

Tabulka 13 ukazuje, co bude vidět student u příslušných předmětů po jeho přihlášení do systému (poslední sloupec). První řádek tabulky říká: Pokud je přihlášen student z Automatizace a předmět patří též pod ústav Automatizace a má zároveň nastaven příznak "p", vidí ho student jako povinný a musí si ho zapsat. Pokud je přihlášen student z Automatizace a předmět patří pod ústav Biomedicíny a má příznak "p" vůbec se studentovi z Automatizace neobjeví a ten si ho nemůže zapsat. Naopak student z Biomedicíny si ho zapsat musí. Zbytek stavů zachycuje Tabulka 13.

Každý student si zapisuje:

- 2x povinný
- 2x volitelný oborový
- 2x volitelný mimooborový
- 2x všeobecně vzdělávací

4.1.3 Funkce a možnosti aplikace

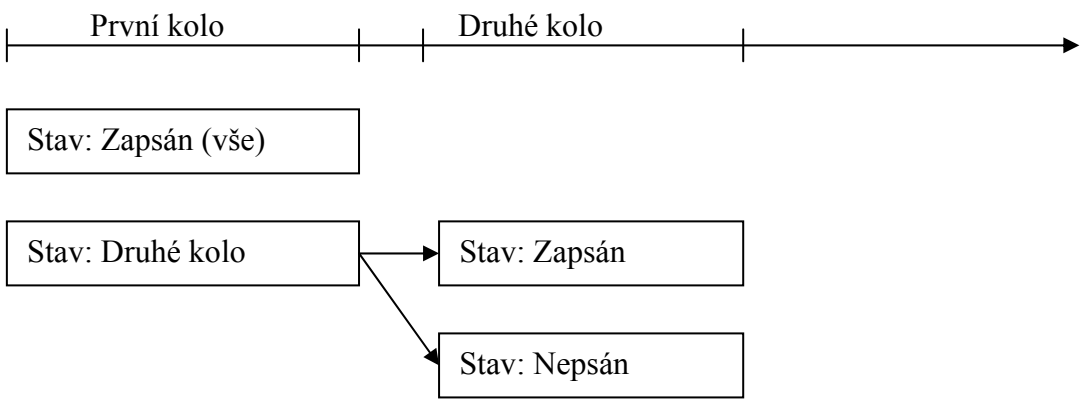
Pokud by byla aplikace napsána, jen tak aby "fungovala" bez dalších vylepšení, nebyly by zřetelné některé výhody a nevýhody jednotlivých platforem. Výhody a nevýhody byly vidět nejlépe právě na nadstavbách systému, tím myslím různé filtry, sortování, navigace atd.

Jelikož jde o zápis předmětů a každý by si měl zvolit předměty sám, musí systém podporovat autentizaci uživatele, která je dána standardním vstupním formulářem(login, heslo).

Po přihlášení je důležité, zda je už doba prvního kola zápisu. Jestliže ano, pak je možné si zapisovat libovolně předměty, přičemž nezáleží na rychlosti, ale pokud je překročena kapacita předmětu, rozhoduje o zapsání v prvním kole vážený průměr studenta. Jak už bylo napsáno výše, studentovi se nezobrazí všechny předměty v databázi, ale pouze ty, které se mu mají zobrazit a s příslušnými příznaky (povinný, volitelný oborový, volitelný mimooborový, všeobecně vzdělávací).

V systému je samozřejmě vidět kapacita předmětu, počet zápisů daného předmětu a aktuální pořadí přihlášeného studenta. Systém dále umožňuje vizuálně kontrolovat, zda se přihlášený student do kapacity zatím vešel (zelený řádek) nebo bude muset projít druhým kolem (červený řádek). Tím, jak se studenti postupně zapisují se ze zapsaného předmětu postupně může stát předmět, který se bude muset zapsat v druhém kole. U těchto předmětů je ve sloupečku „stav“ napsáno "druhé kolo".

Jestliže nastal čas druhého kola, tak studenti, kterým se nepovedl zapsat alespoň jeden předmět v prvním kole, se dostanou do zápisu předmětu v kole druhém. Druhé kolo je už závislé na rychlosti studentů. Pokud je naplněna "dodatečná kapacita" předmětu (K2) rozhoduje o zapsání/nezapsání předmětu pořadí. Pokud se tedy studentovi nepodaří zápis ani ve druhém kole, daný předmět se mu nezapíše a musí si vybrat jiný.



Obrázek 4: Možné stavy u zapisovaných předmětů

Nadstavbové funkce systému:

- Řazení (sortování) záznamů tabulek podle libovolného sloupce
- Při změně řazení záznamů je informace uložena do databáze tak, aby v případě, že student opustí systém, mu příště naskočily "obrazovky" v takovém stavu, jako když je opouštěl
- Filtrování v tabulkách
- Vizuální kontrola, zda byl předmět zapsán nebo se zápis přesouvá do druhého kola
- Navigace, která slouží zároveň k pohybu po aplikaci

4.2 PHP – NEJVĚTŠÍ VÝHODY A NEVÝHODY

Jednoduchost

Je to prakticky jeho největší výhoda zejména pro začínající programátory. Jelikož je tato část, částí praktickou, uvedu zde pár příkladů jednoduchosti zápisu v Php oproti jiným technologiím a uvedu krátký komentář.

```
// Php automaticky rozpozná, že proměnná číslo bude typu
integer

$cislo = 100;

// Pro spojování řetězců slouží operátor tečka. Je zde zároveň
vidět, že řetězce můžou být uzavřeny buď do uvozovek nebo
apostrofů (osobně bych, ale vždy zvolil vždy pouze jednu
variantu)

$text = "Toto je ukázka".', jak jednoduše'." můžu vytvořit
v Php řetězec";


// Do proměnné je možne dát pole, které obsahuje jak čísla,
tak znaky či další pole (opět doporučuji nepřehánět loajalitu
Php)

$pole = array('pes', 'kočka', 44, 2, array('první' => 'ahoj',
'bedo'));


// Vytiskne obsah proměnné pole, aniž bych se musel starat o
typ tisknuté proměnné

print_r($pole);
```

```
// Ukázka asi nejjednoduššího možného zápisu výpisu výstupu

echo 'Jsou stejné';

//ukázka jednoduchého připojení k databázi (2. řádek), výběru
databáze (3. řádek), dotazu na vybranou databázi (4. řádek) a
v cyklu while, který bude probíhat tolikrát, kolik máme
výsledků řádků dotazu, proběhne zpracování

require("mysql_connect.Php");
mysql_connect($_SESSION["host"],$_SESSION["jmeno_c"],$_SESSION
["heslo_c"]);
mysql_select_db($_SESSION["db"]);
$dotaz = mysql_query("select * from nastaveni;");
while($radek = mysql_fetch_row($dotaz))
{
  …
}

//tento dlouhý jednořádkový dotaz na databázi ukazuje, jak
jednoduše pomocí Php vložím do databáze data z formuláře,
pomocí metody POST

$dotaz = mysql_query("insert into predmet
values(NULL,'$_POST[nazev]','$_POST[zkratka]','$_POST[garant]'
,'$_POST[ustav]','$_POST[semestr]','$_POST[kredity]','$_POST[t
yp]','$_POST[kapacita]');");
```

Práce s databází

Na několika příkladech ukážu, jak Php přistupuje do databáze, jak se data mění, vkládají, a jak se naopak „dolují“ pro použití v Php kódu. Naopak od ASP.NET přistupuje Php do databáze, ne pomocí „grafického naklikání dotazu“, ale pomocí několika jednoduchých příkazů, které se o práci s databází starají. Musím se přiznat, že mi novinka, Data-binding (viz. níže) v ASP.NET, učarovala, ale když jsem měl řešit složitější věci a dotazy na databázi, přišla mi daleko lepší práce s těmito několika příkazy Php (mysql_query, mysql_fetch_row, mysql_num_rows atd.).

Jde vždy i o zvyk, ale tady asi nejde jednoznačně říci, zda je lepší používat data-binding nebo šikovné příkazy Php.

Ve Zdrojový kód 5 je vidět, jak Php přistupuje k databázi. Celkově tento kód vypadá poměrně hrozně, ale to proto, že je „nahečmán“ do malého prostoru a hlavně neřeší pouhý výpis předmětů, ale i další věci, které se v kódu testují.

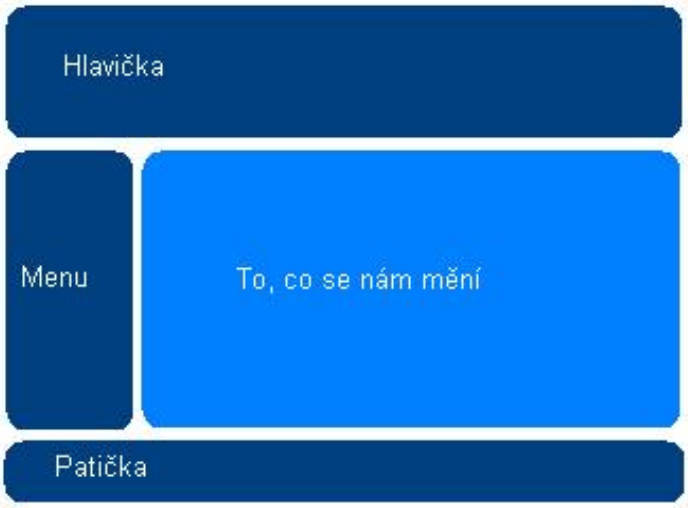
```
//// Výpis předmětů ////
require("mysql_connect.Php");
mysql_connect($_SESSION["host"],$_SESSION["jmeno_c"],$_SESSI
ON["heslo_c"]);
mysql_select_db($_SESSION["db"]);
$dotaz = mysql_query("select zapis.id_zapis, predmet.nazev,
predmet.zkratka, predmet.semestr, predmet.kredity,
predmet.id_predmet, predmet.kapacita, zapis.stav from predmet,
zapis where zapis.id_predmet=predmet.id_predmet and
zapis.id_student=$_SESSION[id_student];");
$nalezen = mysql_num_rows($dotaz);
if($nalezen == 0)
        echo("<br>Nenalezen žádný záznam.");
else {
        echo("<table><tr
bgcolor='#D5D5FF'><td>Název<td>Zkratka<td>Semestr</td><td>Kre
dity<td>Pořadí/Zapsáno/Kapacita<td>Stav<td>Odstranit</td></tr>");
        while($radek = mysql_fetch_row($dotaz)) {
                $dotaz2 = mysql_query("select * from student, zapis
where student.prumer <= $_SESSION[prumer] and
student.id_student=zapis.id_student and
zapis.id_predmet=$radek[5];");
                $dotaz3 = mysql_query("select * from zapis where
id_predmet = $radek[5];");
                $pocet_zapsanych = mysql_num_rows($dotaz3);
                $poradi = mysql_num_rows($dotaz2);
                if($poradi > $radek[6]) {
                        $dotaz5 = mysql_query("update zapis set stav =
'druhe_kolo' where id_zapis=$radek[0];");
                        $dotaz4 = mysql_query("select stav from zapis
where id_zapis=$radek[0];");
                        $aktualizovany_stav = mysql_result($dotaz4,0);
                }
                else
                        $aktualizovany_stav = $radek[7];

        echo("<tr><td>$radek[1]</td><td>$radek[2]</td><td>$radek[3]
<td>$radek[4]<td><table width='100%'><tr><td align='center'
width='35%'>$poradi <td align='center' width='35%'>
$pocet_zapsanych<td align='center'>
$radek[6]</table><td>$aktualizovany_stav</td><td><a
href='zapsane_predmety.Php?id_zapis=$radek[0]'><img border='0'
align='right' src='odstranit.png' title='odstranit'</a></tr>");
        }
        echo("</table>");
}
```

Zdrojový kód 5: Ukázka práce s databází v Php

4.3 ASP.NET – NEJVĚTŠÍ VÝHODY A NEVÝHODY

Master-page

Tato komponenta nebo spíše typ stránky mě potěšil v ASP.NET asi nejvíce, protože je to velmi praktická věc. Dříve než popíšu k čemu tato komponenta vůbec slouží, nastíním, co často programátoři při návrhu stránek řeší:



Obrázek 5: Schéma rozvržení „klasické“ webové stránky

Schéma na Obrázek 5 ukazuje rozvržení klasické webové stránky. Nebudu se zatím zabývat tím, jak vytvořit tyto oddíly a pozicováním, ale obsahem těchto oddílů (DIV). Na většině stránek je obsah hlavičky, patičky a menu statický a nemění se. Mění se většinou pouze obsah (prostřední oddíl). Jak to udělat, ale co nejelegantněji, abych se vlastně mohl starat pouze o „dynamickou část stránky“ při přechodu mezi stránkami? Nejdříve ukážu na příkladu, jak je to možné řešit pomocí Php (méně elegantní metoda):

```
<div class=hlavicka>
<? require("hlavicka.Php");?> </div>
<div class=menu>
<? require("menu.Php");?> </div>
<div class=obsah>
<? require("vyzadana_stranka.Php");?> </div>
<div class=paticka>
<? require("paticka.Php");?> </div>
```

Zdrojový kód 6: Vložení statické a dynamické části stránky u Php

Zdrojový kód 6 ukazuje, co musí většinou programátor Php dělat na každé jeho stránce. Má nadefinovány nějaké oddíly a musí je plnit obsahem, který je sice statický, ale přesto není ušetřen toho, že ho musí na každé stránce „includovat“ např.: pomocí require, což není úplně to nejelegantnější řešení.

Jak to tedy řeší Master-page v ASP.NET? Nejdříve si programátor musí vytvořit stránku typu master page a nadefinovat si třeba pomocí statické části stránky. V maste page se nám objeví oblast uzavřená mezi značky <asp:contentplaceholder> a </asp:contentplaceholder>, což je místo, kam se vždy umístí „měnící se“ část stránky. Vše je pěkně vidět na Obrázek 6. Tím, že jsme si vytvořili tuto master page s jejími statickými částmi a jednou dynamickou částí, můžeme na jejím základě vytvářet nové stránky „ohraničené statickou částí“ (hlavička, menu, ..).

Při vytváření nové stránky stačí pouze zaškrtnout „checkbox – select master page“, čímž si programátor zjednoduší práci tak, že bude psát pouze obsah stránky a „statika“ se přidá. Je to bezesporu komfortnější způsob, než například u Php (JSP), kde se na každé nové vznikající stránce musí statické stránky vždy pomocí include nebo require vkládat.

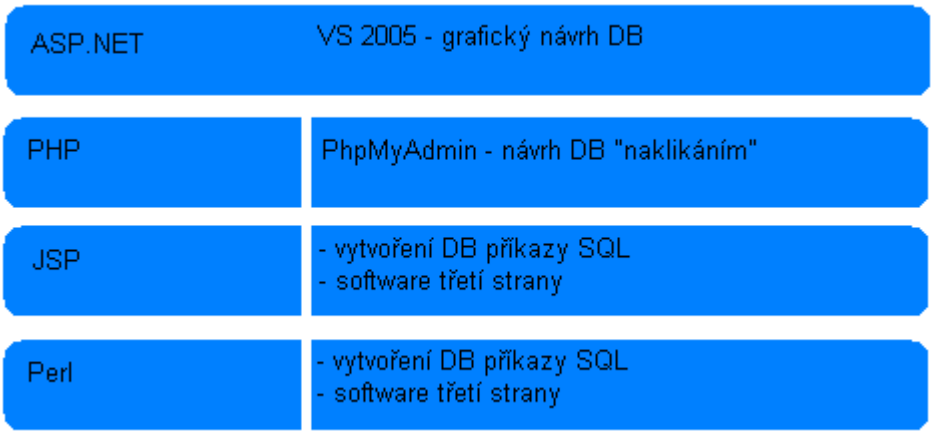
```
5  <html xmlns="http://www.w3.org/1999/xhtml" >
6  <head runat="server">
7      <title>Untitled Page</title>
8      <link rel="Stylesheet" type="text/css" href="format.css" />
9  </head>
10 <body>
11 <div class="hlavicka">...</div>
20 <div class="navigace">...</div>
24
25     <form id="form1" runat="server">
26     <div class="menu">...</div>
29     <div class="variable">
30         <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
31
32         </asp:contentplaceholder>
33     <asp:Menu ID="Menu1" ...>...</asp:Menu>
42     </div>
43     </form>
44
45 </body>
46 </html>
```

Obrázek 6: Master page v ASP.NET

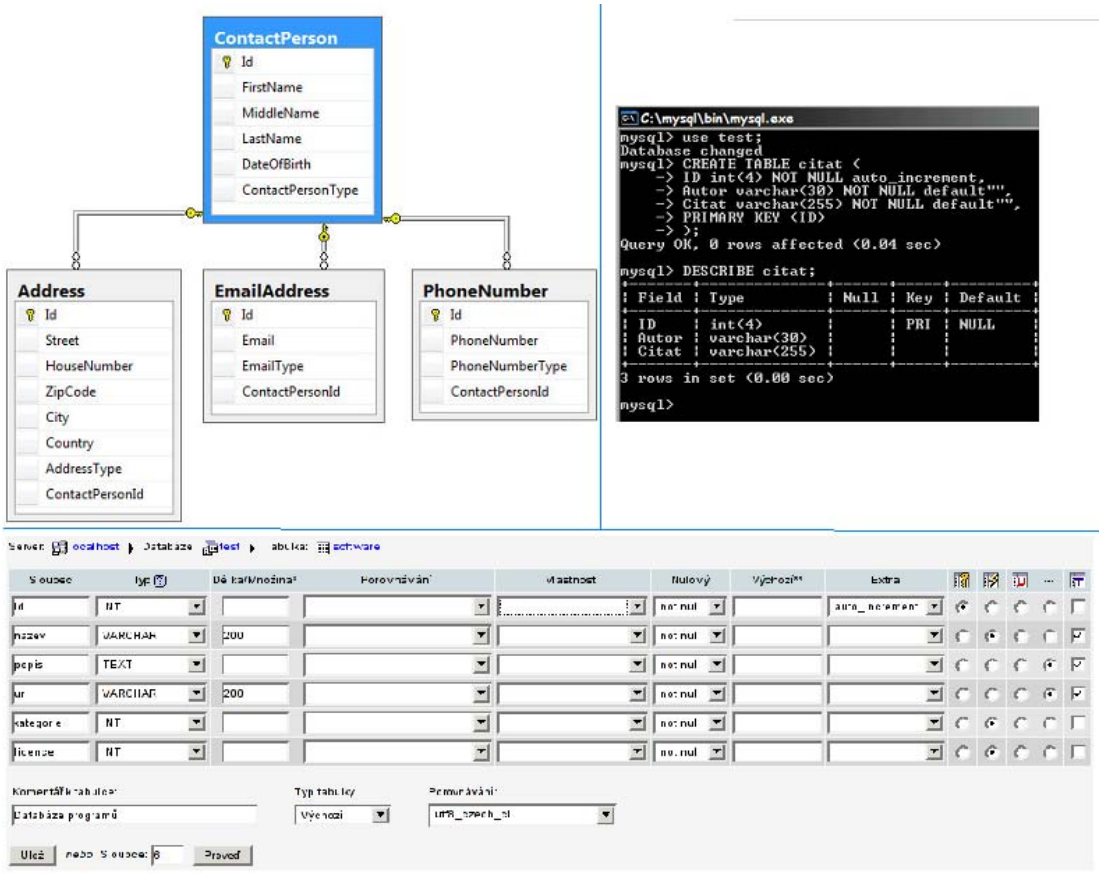
Práce s databází

Práci s databází uvádím jako výhodu, protože se mi líbilo, jako programátorovi, že jsem nainstaloval Framework a Visual Studio 2005 a o databázi jsem se nemusel starat jako o další část nějakého celku, ale hned jsem mohl začít využívat vestavěnou databázi SQL Server Express 2005.

Další výhodou se mi jeví grafické prostředí návrhu databáze. Pomocí nástroje Database Diagrams jsem si jednoduše vytvořil všechny tabulky,jejich sloupce, typy, klíče a ostatní věci týkající se tabulek a nakonec přetažením myši vytvořil relace mezi nimi. Celý datový model vytvořený pomocí VS 2005 je vidět na Obrázek 3. Samozřejmě jsou programátoři, kteří radši vše dělají pomocí příkazové řádky a rádi si stále oprašují příkazy SQL jazyka, ale tento způsob se mi zdá pohodlnější. Na druhou stranu je možné si k Php (patrně i JSP) stáhnou podobný nástroj, kterým je například PhpMyAdmin, ale opět to není už tak konfortní, když musím jako programátor hodinu instalovat a nastavovat další nástroj, abych si mohl jednoduše vytvářet a modifikovat tabulky a záznamy. Pokud uživatel, který chce vytvářet databázi pomocí VS 2005 a nástroje Database Diagrams, dělal někdy v MS Office Access, bude pro něj návrh hračka, protože jsou si tyto nástroje velmi podobny. Na Obrázek 7 jsem se pokusil shrnout své myšlenky, které jsem popisoval výše, ještě graficky. Obrázek 8 ukazuje tři možnosti, jak jednoduše či pracně vytvořit databázi a její tabulky.



Obrázek 7: Práce s databází vyjádřená graficky



Obrázek 8: Prostředí vytváření DB – VS 2005, příkazový řádek, PhpMyAdmin

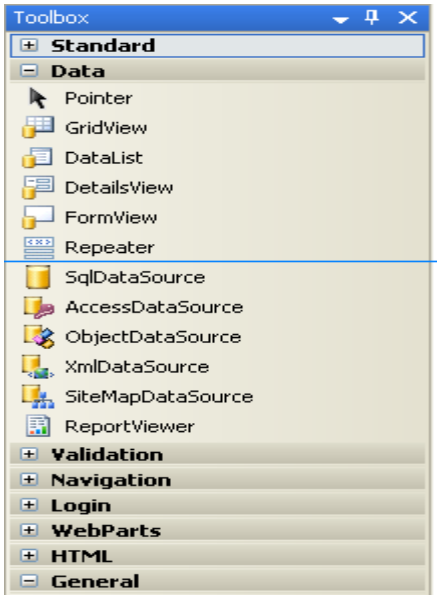
Data binding

Data binding, neboli propojování (napojování) dat je další výborná funkčnost ASP.NET, která slouží k propojení dvou stran. Na jedné straně je to databáze, která poskytuje data a na druhé straně například webová stránka, která tyto data zobrazuje. Toto zobrazování dat si ukážeme zároveň i v Php (JSP obdobně), aby byl vidět rozdíl přístupu k této problematice. Zdrojový kód 7 ukazuje, jak pomocí Php „vydoluji“ data na stránku. Nejdříve se samozřejmě připojím k databázi, položím select a pomocí příkazu mysql_fetch_row postupně vypíši požadované záznamy na stránku do tabulky.

```
require("mysql_connect.Php");
mysql_connect($_SESSION["host"],$_SESSION["jmeno_c"],$_SESSI
ON["heslo_c"]);
mysql_select_db($_SESSION["db"]);
$dotaz = mysql_query("select * from predmet order by
$_SESSION[sort_1];");
echo("<table width='1200px'>“);
while($radek = mysql_fetch_row($dotaz)) {
echo("<tr
bgcolor='#B8FF95'><td>$radek[1]<td>$radek[2]<td>$radek[3]<td>$ra
dek[4]<td>$radek[5]<td>$radek[6]<td>$typ<td><table
width='100%'><tr><td width='45%' align='right'>$pocet_zapsanych<td
align='left'> /  $radek[9]</table><td><td><a
href='Detail_predmetu.Php?id_predmet=$radek[0]'><img title='Více
info k předmětu' align='right' border='0' src='info.png'></a></tr>");
                                    $_SESSION[zapsano] =
$_SESSION[zapsano] + 1;
}
echo("</table>");
```

Zdrojový kód 7: Výpis z databáze pomocí Php

Pro ukázku, jak k tomuto problému přistupuje ASP.NET, nebudu vůbec potřebovat zdrojový kód, protože jak dále uvidíte, je tento problém řešen stylem „naklikej si sám“. Na Obrázek 9 je toolbox ve VS 2005 a rozbalená nabídka data. Nabídku data můžeme rozdělit na takzvané konzumenty dat (prvky po modrou dělicí čáru) a zdroje dat.



Obrázek 9: Toolbox - data v VS 2005

Potřebuji-li zobrazit tabulku předmětů, která je na Obrázek 10, nemusím psát v tomto případě žádný kód, pouze dostat na svou master-page stránku 2 prvky, GridView, který bude konzumentem dat a druhý prvek, SqlDataSource, který bude zdrojem dat. Samozřejmě nestačí pouze prvky přetáhnou, ale je nutné je i nastavit, což se provádí naklikáním, a to včetně dotazu do databáze. Obrázek 11 ukazuje menu prvku GridView, kde si jako zdroj dat volím zdroj se jménem SqlDataSource, což je vytvořený (naklikaný) select do tabulky „Předmět“. Z nabídky prvku GridView je dobře vidět, co vše si můžu nastavit. Mimo jiné se mi líbí nastavení AutoFormat, kterým můžu říci, jak vytvořená tabulka bude vypadat, což se v případě Php musí řešit, buď přímo nastavením tagů <table> <tr> <td> nebo pomocí stylů. U ASP.NET se mi to jeví jednodušší.

Prvek GridView a SqlDataSource nejsou jediné, jak je vidět z Obrázek 9, ale pro princip tvorby seznamu v ASP.NET tato ukázka postačuje.

Zapsané předměty > Nabídka předmětů

Název předmětu	Zkratka	Garant	Ústav	Semestr	Kredity	Typ	Kapacita	Kapacita2	Zapsat
Biologie člověka	BCL		bmi	zimni	5	p	2	1	zapsat
Číslicová analýza a zpracování signálů	CZA		bmi	zimni	5	p	6	2	zapsat
Úvod do medicínské informatiky	UMI		bmi	letni	5	p	2	1	zapsat
Terapeutická a protetická technika	TPT		bmi	letni	5	p	6	2	zapsat
Multitaktní systémy	MUT		bmi	zimni	5	n	2	1	zapsat
Didaktika	IPD		bmi	zimni	5	n	4	2	zapsat
Počítače a programování	PC1		bmi	letni	5	n	2	1	zapsat
Rentgeny	XRA		bmi	letni	5	n	4	2	zapsat
Výpočetní technika v automatizaci	VTA		amt	zimni	5	p	2	1	zapsat
Měření v elektrotechnice	MVE		amt	zimni	5	p	6	2	zapsat

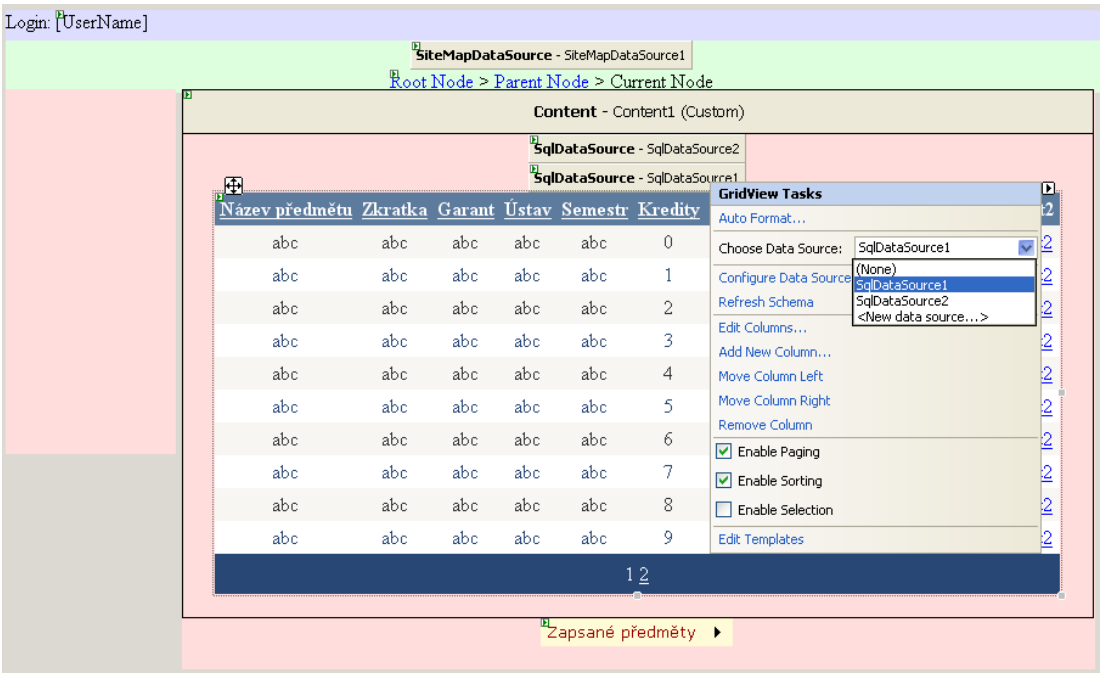
1 2

Obrázek 10: Výpis nabídky předmětů

Abych pouze Data-binding nevychvaloval a byl objektivní, tak musím říci také jeho slabé stránky. Byl vytvořen, aby i programátor bez znalosti Html mohl vytvořit jistou funkčnost[1] a pokud jde o jednoduché vypsání záznamů jedné tabulky, tak to i tento programátor zvládne. Problém nastane tehdy, potřebuji-li vypsat např. data ze dvou tabulek nebo dělat složitější podmínky a další „vylomeniny“, což SQL umožňuje. V tomto případě už nám nestačí pouhé klikání, ale obsluhu už musíme doplnit psaním kódu, což komfortu nepřidává, protože programátor část věcí řeší pomocí grafického prostředí a další část kódem.

Když to celé shrnu, tak je to asi tak: Na jednoduché a středně obtížné výpisy dat do různých tabulek a formulářů je ASP.NET, jak stvořené, protože pomocí pár kliků vytvoří i programátor-začátečník fungující stránku. Na složité dotazy a další složitosti kolem databází tento grafický-klikací způsob už tak „šikovný“ není a dal bych naopak přednost Php, JSP, kde je vše na jednom místě a tím čitelnější.

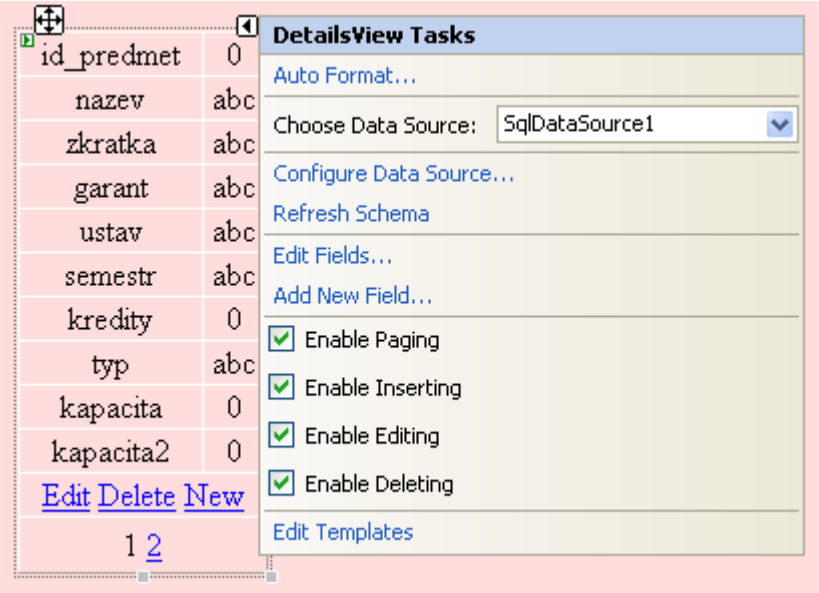
1 Microsoft později uznal, že tato jeho filosofie byla nesprávná



Obrázek 11: Sestavení stránky, aby vznik výsledek, zobrazený na Obrázek 10

Vkládání, mazaní a úprava dat

Častým požadavkem na webových stránkách bývá kromě výpisu dat možnost, data co nejjednodušeji upravovat, mazat a případně i vkládat. Kdo programuje více webových stránek, nebo se touto činností živí, má pravděpodobně napsané určité univerzální šablony, které pouze upravuje, aby stále dokola nemusel psát ty samé kódy pro úpravu, mazání a vkládání dat. ASP.NET se k tomuto problému postavil také čelem a pro programátory je zde možnost využít jisté automatiky. Většina konzumentů dat, jako je GridView, DetailsView atd. má možnost zaškrtnou políčko umožňující vkládání, úpravu a mazání dat. Toto nastavení se dělá při vytváření DataSource v nabídce Advanced. Pokud toto povolím, pak mi stačí zaškrtat příslušná políčka u příslušného prvku, jak ukazuje Obrázek 12, kde povoluji vkládání (Enable Inserting), editování (Enable Editing) a mazání (Enable Deleting).



Obrázek 12: vkládání, úprava a mazání záznamů v prvku DetailsView

Řazení záznamů a stránkování

V případě, že chceme na našich stránkách zobrazovat více dat, se asi nevyhneme potřebě, udělat toto zobrazování komfortní. Ke komfortu zobrazování dat patří podle mě, kvalitní stránkování, řazení záznamů podle kteréhokoliv sloupce a to vzestupně a sestupně a také filtr. Jelikož řazení záznamů a stránkování řadím mezi výhody ASP.NET, porovnám, jak tuto funkčnost řeší Php a ASP.NET.

V Zdrojový kód 8 je část mého kódu pro řazení záznamů v Php. Tímto kódem nechci říci nebo ukázat, jak dobře nebo špatně jsem to napsal, ale to že věc, kterou potřebuje každá druhá webová aplikace, jsem vůbec musel psát. Psát bych tento kód sice nemusel, protože můžu využít toho, že Php je Open Source a na internetu si můžu jednoduše stáhnout řazení, které už napsal někdo jiný, ale opět nebudu oproštěn od práce, přizpůsobit si toto „cizí“ řazení své aplikaci. Pro ty, kdo by nahlédli do tohoto kódu musím ještě napsat, že se jedná o řazení záznamů, kde se do databáze ukládá, který uživatel si jak nastavil toto řazení. To znamená, pokud si uživatel Pepa nastaví řazení záznamů podle semestru vzestupně a opustí aplikaci, při příštím vstupu do aplikace, tam stále bude nastaveno jeho řazení.

V Zdrojový kód 8 je zhruba pouze polovina celého kódu, tak jak to tedy elegantněji řeší ASP.NET. Pokud používám prvky Data-bindingu, tak je situace velice jednoduchá a překvapivě opět nemusím psát žádný kód ☺ Jak je vidět z Obrázek 11 mají obecně konzumenti dat možnost zapnout stránkování (Enable Paging) a řazení (Enable Sorting). Pokud tyto funkčnosti zapneme, tak nám fungují, jak potřebujeme. Pokud nám nevyhovuje jejich nastavení, tak si pomocí Properties vybraného konzumenta dat, můžeme nastavit třeba počet záznamů na stránku, typ řazení a další. Pokud jde o řazení to funguje, jak je většina uživatelů zvyklá: Jednou kliknu na příslušný hyperlink – řazení vzestupně, podruhé kliknu – řazení sestupně.

```
$dotaz = mysql_query("select * from nastaveni_user where
id_student = $_SESSION[id_student];");
$radek = mysql_fetch_row($dotaz);
if($radek[1] == "" || $radek[2] == "") {
        mysql_query("insert into nastaveni_user
values($_SESSION[id_student],'nazev','ASC');");
        $_SESSION[sort_1] = "nazev ASC";
}
else
        $_SESSION[sort_1] = $radek[1] . " " . $radek[2];


if($_GET['sloupec'] != "") {
        if($_GET['sloupec'] == "nazev") {
                if($_GET['sort'] == "ASC") {
                        mysql_query("update nastaveni_user set
sort_nab_sloupec='nazev', sort_nab_smer='ASC' where
id_student=$_SESSION[id_student];");
                        $_SESSION[sort_1] = "nazev ASC";
                }
                elseif($_GET['sort'] == "DESC") {
                        mysql_query("update nastaveni_user set
sort_nab_sloupec='nazev', sort_nab_smer='DESC' where
id_student=$_SESSION[id_student];");
                        $_SESSION[sort_1] = "nazev DESC";
                }
        }
        if($_GET['sloupec'] == "zkratka") {
                if($_GET['sort'] == "ASC") {
                        mysql_query("update nastaveni_user set
sort_nab_sloupec='zkratka', sort_nab_smer='ASC' where
id_student=$_SESSION[id_student];");
                        $_SESSION[sort_1] = "zkratka ASC";
                }
                elseif($_GET['sort'] == "DESC") {
                        mysql_query("update nastaveni_user set
sort_nab_sloupec='zkratka', sort_nab_smer='DESC' where
id_student=$_SESSION[id_student];");
                        $_SESSION[sort_1] = "zkratka DESC";
                }
        }
```

Zdrojový kód 8: Řazení záznamů (s pamětí) v Php

Intelli sense a hierarchické rozdělění stránky

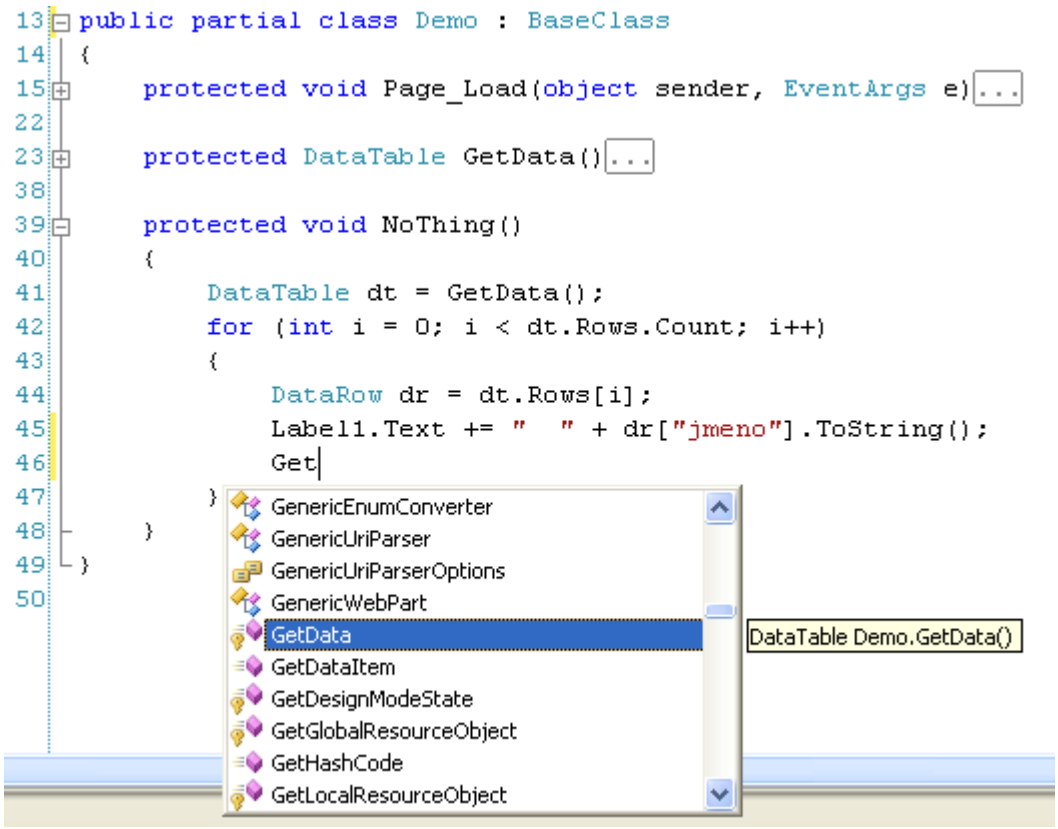
Většina lidí, kteří píší webové stránky, pravděpodobně někdy zkusila psát jednoduché Html stránky třeba i v obyčejném poznámkové bloku a vše fungovalo

perfektně, ale když už programátor musí používat hodně tagů a nastavovat různé jejich atributy, tak už málo komu se chce si všechny tyto věci pamatovat a proto „se sáhne“ po nástrojích, které napovídají atributy tagů, dokončují ukončovací značky tagů a různě pomáhají.

Ve VS 2005 šli v této problematice ještě o kousek dál k tzv. „Intelli sense“, které nabízejí nejen dokončování a atributy běžných Html tagů, ale nechají se pomocí nich zvolit existující třídy a jejich metody, které vám VS 2005 nabízí. Ukázka je vidět na Obrázek 13, kde nám intelli sense nabízí jistou metodu v naší třídě.

Intelli sense, ale nefunguje ve VS 2005 například v xml souborech, které VS 2005 také používá (web.config). Toto vylepšuje VS 2008, kde už je tato funkčnost rozšířena i na tyto soubory.

Dále je vidět na Obrázek 13, že se vlevo vyskytují ikonky v podobě znamének plus a mínus, které nám dovolují celou metodu nebo dokonce celou třídu sbalit a opět rozbalit, čímž můžeme hledání v kódu a i celý kód zpřehlednit. Zvolil jsem ukázku třídy a tří metod, ale tato hierarchie a sbalování funguje i na klasické aspx stránce, kde je hierarchie <body> <table> atd.

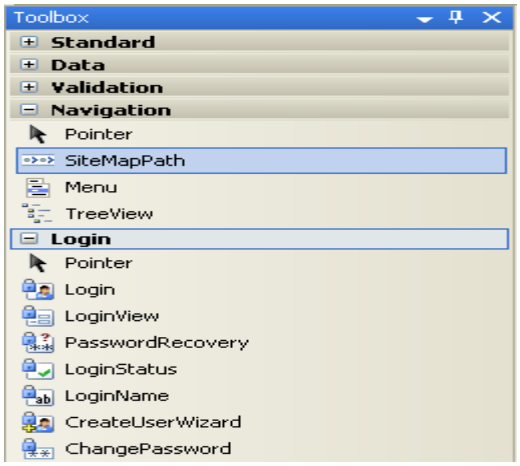


Obrázek 13: Intelli sense a hierarchie ve VS 2005

Navigace na stránkách a logování

Jsem nucen začít opět touto větou: Ve většině IS nebo internetových obchodech je dobré uživatele informovat, v které části webového prostoru naší aplikace se nalézá. Proto VS 2005 umožňuje vložit prvek pro navigaci. Navigaci si vložím na stránku, kde chci zobrazit tuto navigaci (viz Obrázek 14), ale to samozřejmě ještě nestačí. Dále si musím nadefinovat hierarchie svých stránek v xml souboru zvaném Web.sitemat, který slouží jako zdroj dat pro prvek zobrazující navigaci. Příklad nastavení souboru Web.sitemap ukazuje Obrázek 15.

Podobný princip platí i pro prvky, kterými si uživatel přihlašuje do systému, kdy je oproštěn od práce dělat si vlastní přihlašovací formulář a vloží ho přímo do stránky z Toolboxu-Login. Uživatelé, kteří mají přístup do systému se konfigurují přes nabídku Website-ASP.NET Configuration, kde se nechají nastavit i role uživatelů a omezení přístupů jednotlivých uživatelů. To vše se implicitně ukládá do databáze ASPNETDB.MDF.



Obrázek 14: Menu pro navigaci a logování v nástroji Toolbox

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
    <siteMapNode url="~/zapsane_predmety.aspx" title="Zapsané předměty"  description="Zapsané předměty">
        <siteMapNode url="~/nastaveni.aspx" title="Nastavení"  description="Nastavení" />
        <siteMapNode url="~/nabidka_predmetu.aspx" title="Nabídka předmětů"  description="Nabídka předmětů" />
        <siteMapNode url="~/pridej_predmet.aspx" title="Přidej předmět"  description="Přidej předmět" />
        <siteMapNode url="~/pridej_student.aspx" title="Přidej student"  description="Přidej student" />
    </siteMapNode>
</siteMap>
```

Obrázek 15: Web.sitemap v ASP.NET

Stylování objektů (Skinování)

Skinování je velmi podobné kaskádovým stylům, ale rozdíl je v tom, že CSS se týká stylování Html prvků, kdežto skinování celých objektů. Tyto technologie se můžou dobře doplňovat. Skinování můžu použít například v případě, kdy chci ve své aplikaci vypisovat všechna data pomocí prvku GridView a chci, aby tabulky vzniklé pomocí tohoto prvku vypadaly jednotně (stejný vzhled). Potom si nastavím skin, který bude říkat, všechny prvky GridView budou vypadat následovně. Kraťoučký příklad je v Zdrojový kód 9 , který stačí uložit do souboru s koncovkou *.skin a aplikovat na vybranou stránku nebo celý web.

```
<asp:GridView runat="server"> <RowStyle BackColor="White" />
<HeaderStyle BackColor="Blue" ForeColor="Yellow" />
<AlternatingRowStyle BackColor="Blue" /> </asp:GridView>
```

Zdrojový kód 9: Zápis skinu v ASP.NET

4.4 JSP – NEJVĚTŠÍ VÝHODY A NEVÝHODY

Čistokrevné OOP

Proč vůbec zmiňuji OOP u JSP jako výhodu, když OOP podporují všechny 4 technologie? Je to ten důvod, že málokterý programátor, který není zvyklý programovat s OOP je v technologiích jako Perl a Php nepoužije, dokud k tomu není nějak donucen. Naopak v Javě se člověk bez OOP neobejde i když si to ani neuvědomuje. Všude jsou třídy a jejich metody a začátečník programátor se je naučí nějak používat a až potom mu dojde, že programuje stylem OOP.

Java (JSP) programátora donutí takto programovat a myslím si, že je to správná cesta. Malá ukázka prográmku s třídou Vypínač napsaná v Java je na Zdrojový kód 10.

```
public class Vypinac {
  private boolean poloha = false;

  public void zapni() {
    poloha = true;
  }

  public void vypni() {
    poloha = false;
  }

  public void vypisStav() {
    if (poloha)
      System.out.println("Zapnuto.");
    else System.out.println("Vypnuto.");

  }
}
```

Třída Vypinac obsahuje veřejně přístupné (public) metody zapni() a vypni() a soukromou (private) členskou proměnnou poloha. Podle této třídy může program vytvořit teoreticky libovolné množství nezávislých instancí:

```
public class Program {
  public static void main(String[] args) {
    Vypinac prvni = new Vypinac();   // vytvoření vypínače
    Vypinac druhy = new Vypinac();   // vytvoření vypínače

    prvni.zapni();

    prvni.vypisStav();
    druhy.vypisStav();
  }
}
```

Zdrojový kód 10: Ukázka OOP v Javě

Složitost pro začínající programátory

Moc dlouho jsem přemýšlel, jestli mám tento bod uvést, ale jelikož jde o srovnání technologií, tak jsem se rozhodl pro jeho uvedení.. Používání OOP považuji za správný krok v programování. Na druhou stranu se i bez něho nechá vytvořit vyhovující výtvor. Tím že je OOP v Javě „vnucováno“ hned od počátku může spoustu začínajících programátorů od této platformy odradit, což je škoda.

Budoucí směr vývoje

Společnosti vyvíjející Javu (JSP) si uvědomily, že filosofie, kterou uplatnil Microsoft ve svém Frameworku je zdařilá a správná, proto se tímto směrem do budoucna také, zdá se, vydá. Mezi první prvky, které mají vytvořit obdobu nových prvků v .NET jsou v JSP prvky JSF (Java Server Faces). Uvidí se další kroky, kterými se Java vydá, aby trumfla inovativní přístup Microsoftu a její .NET technologie.

4.5 PERL – NEJVĚTŠÍ VÝHODY A NEVÝHODY

Jednoduchost

Pro jednoduché úkony je Perl vhodný nástroj, protože je pro začínajícího programátora dobře „stravitelný programovací jazyk“. Příklady kódu jsem se rozhodl nevkládat, protože si je velmi podobný svým stylem jednoduchého programování s Php. Zkrátka a dobře, jsou zde jednoduché a čitelné příkazy, kterým ovšem chybí

něco, co by je nějak spojovalo a udělalo z nich hezký kompaktní celek bez velkých snah programátora.

Loajalita (volnost)

Psaní kódu v Perlu má sice pravidla, ale jsou poměrně volná, proto programátor může napsat jednu věc mnoha způsoby, kde kód stále funguje. Toho se dosti využívá a díky této volnosti někteří programátoři píší svoje kódy „příšerným stylem“.

Moduly

Nelíbí se mi věc ohledně modulů, které by měli řešit to, že když chci programovat nějakou věc a nemám na to dost prostředků, tak si stáhnu modul a pokračuji dále. Tato filosofie se mi v době, kdy jsou pevné disky o velikostech stovek GB příliš nelíbí, protože jako programátor, bych rád sedl k počítači a hned programoval, a ne abych se stále staral, který modul pro co stáhnout. Tento „modulový“ přístup není pouze u Perlu, ale část této „modulovosti“ je i v Php.

V tomto směru mě vyhovuje více robustnost ASP.NET, kde je vše pohromadě a já můžu v jednom prostředí vše naprogramovat.

Úpadek této technologie

Bohužel musím konstatovat, že Perl byl asi největším oříškem pro porovnání. Moc se o něm nepíše, věci které podporuje, podporují i ostatní technologie a nutno říci, že většinou v lepším provedení. Perl nemůžu ani objektivně zhodnotit, protože ho „mám prozkoumaný“ nejméně, ale přesto jsem nějaké výhody, které jsou popsány výše, našel.

4.6 ZPŮSOB HODNOCENÍ

Závěrem praktické části, bych chtěl objasnit věc, které si asi každý všimne, a to, že praktická část o ASP.NET je o dost delší než ostatní části.

Důvod je jednoduchý: Z těchto čtyř technologií je filosofie technologie ASP.NET o tolik jiná než u ostatních technologií a přistupuje k většině problémů velmi „user-friendly“ metodou, že to jinak nešlo. Někdy mi připadalo, že porovnávám 3 druhy brambor a hlávku zelí. Takže co můžete říct o rozdílech třech typů brambor? Je toho méně než, co můžete říci o rozdílech brambory a hlávky zelí ☺ Jako hlávku zelí jsem tedy bral ASP.NET a proti němu jsem stavěl většinou bramboru v podobě Php, čímž bych ho nerad nadhodnocoval nebo podhodnocoval vůči JSP a Perlu.

Jelikož bylo toto praktické srovnání dost náročné a nerad bych dostal od jistých programátorů „do zubů“ snažil jsem se vždy danou technologii a mnou popisované závěry konzultovat s lidmi, kteří jsou v oboru zkušenější a o dané problematice toho vědí hodně. Bohužel, jak už jsem jednou psal, Perl jako by byl někde na „třetí koleji“ a pomalu upadal. Tudíž pro Perl konzultace neprobíhaly ☹ , přesto jsem se snažil o jistý nástin praktického srovnání.

Dále chci upozornit na další věc: V praktické části jsem zmiňoval pouze výhody a nevýhody jednotlivých technologií vyplývající z mé praktické části práce a z konzultací s programátory, takže už třeba znovu nevyzdvihuji, že Php a Perl jsou zdarma, že jsou Open Source a další výhody, které se nedají popřít. Těmto „teoretickým“ výhodám se věnuji v první půlce této diplomové práce a zde je už znovu neopakuji.

A pro ty, kdo začínají programovat, uvedu ještě tabulku, která by měla vyjadřovat můj postoj k těmto čtyřem technologiím a vhodnost použití v různě rozsáhlých projektech.

Technologie	Malý projekt	Střední projekt	Velký projekt(IS)
Php	10	8	6
ASP.NET	5	9	10
JSP	7	8	9
Perl	9	7	4

Tabulka 14: Vhodnost použitý jednotlivých technologií v různě rozsáhlých projektech

Malé hodnoty ve velkých projektech získaly Php a Perl kvůli tomu, že je mnohem těžší v nich poté něco upravovat nebo doprogramovávat, jelikož je výsledný výtvor „pouze“ skupinka skriptů, které si programátor nějak prováže, ale jinak nejsou nijak vázány. Což na druhou stranu neznamená, že pomocí Perlu se nenechá naprogramovat Informační systém.

Nízké hodnocení získalo ASP.NET u malých projektů pouze z důvodu toho, že pro malé projekty je ASP.NET až příliš robustní technologie a bylo by to asi jako „jít na komára s dělem“. A také tato robustnost nám u malých projektů i kapku brzdí náš malý výtvor.

5. ZÁVĚR

Vybrat univerzálního „šampióna“ vhodného pro tvorbu jakýchkoliv informačních systémů je velmi těžké. Všechny technologie mají svoje pro i proti. Část hodnocení, a proč jsem právě tak hodnotil, můžete najít v podkapitole 4.6způsob hodnocení. Jak jednotlivé technologie dopadly?

Php je poměrně jednoduchá technologie a zároveň programovací jazyk, na kterém se výborně začíná a zároveň obsahuje dostatek funkčnosti pro jakkoliv velký projekt. Mezi jeho další výhody patří to, že není vázáno na konkrétní platformu a také to, že je Open Source, takže jeho kódy můžou využívat další programátoři. Dále pro tuto technologii hraje fakt, že si velmi dobře rozumí z databázemi a hlavně je zdarma. Aby zde nebyla jenom chvála, tak bych si tuto technologii určitě nevybral pro tvorbu obrovských informačních systémů a portálů, protože při změnách v takto velkých projektech už není tato technologie tak „šikovná“. Také používá (částečně) interpretovaný jazyk, který výkonnostně zaostává za kompilovaným.

Pro technologie JSP založené na Javě je největším trumfem perfektní OOP a fakt, že je už tato technologie ověřena praxí, na rozdíl od technologie .NET, která není ještě tolik ověřená praxí. V JSP bych se nebál vytvořit velký IS, protože je na to dobře vybavena. Na druhou stranu, zde není nic převratného, co by ji činilo jedinečnou mezi porovnávanými technologiemi.

Perl není špatnou technologií, ale spíše bych ji doporučil nováčkům pro tvorbu menších projektů, i když samozřejmě dobrý programátor by zvládl naprogramovat i informační systém, ale jeho správa pomocí Perlu by nepatřila k jednodušším variantám. Uživatelům Linuxu by tato technologie mohla vyhovovat více, jelikož se většina věcí spojená s Perlem odehrává v textovém režimu, na což jsou lépe vybaveni „linuxáři“.

Jako poslední jsem si nechal technologie ASP.NET, protože je to technologie, která se od ostatních nejvíce odlišuje, především svojí filosofií. Microsoft před několika lety dosti ztrácel na Javu, Php, protože jeho nástroje nebyly příliš dobré. S tím se nechtěl smířit, a proto vznikla technologie .NET. Musím říci, že má velice málo slabin a nabízí programátorovi velký komfort, od hierarchického členění

stránky po data binding. Zde už znovu nebudu opakovat jeho výhody. Tou největší výhodou je, že systém funguje jako jeden robustní celek, ve kterém má programátor vše, co potřebuje. Jedinou nevýhodou by mohlo být to, že není ještě tolik ověřen praxí, protože je to technologie relativně mladá (ukáže čas).

V semestrálním projektu 2 jsem vyřkl svůj postoj a vyhrála u mě mírně technologie Php, ale mezitím jsem „prozkoumal“ technologii ASP.NET, která mě velice mile překvapila. Tím nechci říci, že ostatní technologie jsou horší. Pokud bych neměl programovat pouze formulář, který odesílá data, vybral bych si jednoduché Php či JSP, ale na kterýkoliv větší projekt bych si pustil zřejmě Visual Studio 2005 (2008). Ještě jednou podotknu, že menší projekty je lepší řešit pomocí Php, případně Perlu a na velké projekty poštvat JSP či ASP.NET.

I přes moje zhodnocení, které bohužel není poskvrněno dlouhou praxí, si musí vybrat každý svého šampióna, protože není možné obecně říci: „tahle technologie je nejlepší na světě ve všech ohledech“, takže hodně štěstí při vybírání ☺

6. POUŽITÉ INFORMAČNÍ ZDROJE

[1] BURD, B.: *JSP: JavaServer Pages Podrobný průvodce*. Praha, Computer Press, 2003

[2] HEROUT, P.: *Učebnice jazyka Java*. České Budějovice, Kopp, 2000.

[3] Kolektiv autorů: *Php programujeme profesionálně – 2. opravené vydání*. Praha, Computer Press, 2001

[4] Kolektiv autorů: *C# Programujeme profesionálně*. Brno, Computer Press, 2003

[5] http://www.interval.cz

[6] http://www.linuxsoft.cz

[7] http://www.linuxsoft.cz/Php/

[8] http://www.zive.cz/default.aspx?searchtext=ASP.NET&sart=&ssec=4%2C3%2C5

[9] ATASOFT: *Srovnání Php a ASP.NET*, [cit. 2005-06-12]. Dostupné na internetu: <http://blog.vyvojar.cz/atasoft/archive/2005/06/12/6107.aspx >

[10] http://www.themidnightcoders.com/doc20/richclientprimer/server/implementation-java.htm

[11] http://atrey.karlin.mff.cuni.cz/~bim/pub/JSP/referat/referat.Html

[12] BERGMAN, N.: *Dynamic Web-based data access using JSP and JDBC technologies*, [cit. 2001-09-01]. Dostupné na internetu: <http://www.ibm.com/developerworks/java/library/j-webdata/>

[13] http://www.fi.muni.cz/~adelton/Perl/europen2004/tutorial.Html

[14] http://www.linuxsoft.cz/article.Php?id_article=1468

[15] ŠEDA, J.: *J2EE, .NET a vývoj rozsáhlých systémů*, [cit. 2003-02-10]. Dostupné na internetu: <http://interval.cz/clanky/j2ee-net-a-vyvoj-rozsahlych-systemu-1>

[16] BERNARD, B.: *Php vs. ASP.NET – pokus o seriózní srovnání*, [cit. 2006-09-4]. Dostupné na internetu: < http://www.borber.com/blog/php-vs-asp-net-pokus-o-seriozni-srovnani>

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

Php	Personal Home Page (dříve) – Osobní domací stránky PHP Hypertext Preprocesor (dnes) – PHP hypertextový preprocesor
JSP	Java Server Pages – Java stránky na straně serveru
Perl	Practical Extraction and Reporting language – praktický skriptovací jazyk
ASP	Active Server Pages – aktivní stránky na straně serveru
Html	HyperText Markup Language – značkovací jazyk
DHtml	Dynamic HyperText Markup Language – dynamický značkovací jazyk
CGI	Common Gateway Interface – obecná brána rozhraní
OOP	Object-Oriented Programming – objektově orientované programování
XML	eXtensible Markup Language – rozšiřitelný značkovací jazyk
CSS	Cascading Style Sheets – kaskádové styly
SSJS	Server Side JavaScript – Javascript na straně serveru
JDK	Java Development Kit – vývojová souprava pro Javu
ODBC	Open DataBase Connectivity – otevřené databázové propojení
JDBC	Java DataBase Connectivity – Java databázové propojení
DBI	Database interface – databázové rozhraní
MSIL	Microsoft Intermediate Language – jazyk do něhož .NET kompiluje
DOM	Document Object Model – objektový model dokumentu
IIS	Internet Information Server – internetový informační server

PŘÍLOHY

Součástí diplomové práce je disk CD, který obsahuje softwarovou verzi této diplomové práce a soubory (skripty) testovací aplikace pro Php, JSP a ASP.NET. Kromě těchto souborů se na disku nacházejí ještě některé nástroje potřebné k vývoji skriptů pro danou technologii. Z důvodů velikosti jsou zde pouze velikostně menší nástroje.

Rozčlenění stromové struktury na CD jsem zvolil následovně: