JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

VYTVOŘENÍ PROGRAMU PRO GENEROVÁNÍ TESTŮ

Vypracoval: Jan Nádvorník
Vedoucí bakalářské práce: Mgr. Jiří Pech, Ph.D

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

V Českých Budějovicích dne…… ……

podpis

Poděkování

Na tomto místě bych chtěl poděkovat vedoucímu mé bakalářské práce Mgr. Jiřímu Pechovi, Ph. D. za ochotu vedení, cenné rady, připomínky a trpělivost.

Abstrakt

Hlavním cílem této bakalářské práce je pomocí jazyka PHP vytvořit fungující aplikaci pro generování testových otázek. K tomu je vytvořena a využívána databáze MySQL, do které se vkládají otázky a odpovědi. Tato data budou na základě uživatelských požadavků pomocí vytvořené aplikace promíchána a náhodně vygenerován test. Výstupem práce bude PDF formát a soubor XML, obsahující vygenerované testové otázky a odpovědi.

Abstract

BA thesis is focused on creating working application that will generate test questions using PHP programming language. For that purpuse I will use MySQL database into which I will insert questions and correct answers. The application will mix those data and create random test based on users requirements. The output of the work will be XML text file consisting of test questions and generated answers.

Obsah

1. Úvod

Pro pedagogy, kteří potřebují na výuku větší množství verzí testů, je nevýhodné vymýšlet tyto testy jednotlivě. Výhodnější možností by bylo vytvořit systém pro jejich generování. Můžeme se setkat s různými nástroji pro generování testů, jejichž použití často limitují různé faktory, ať už je to cena za licenci, nebo nemožnost test vygenerovat do tištěné podoby.

Tato práce se zabývá vytvořením jednoduchého systému pro testování studentů. Systém je složen z návrhu jednoduché databáze, do které se vkládají testové otázky, respektive testy. K této databázi přistupujeme pomocí uživatelského rozhraní navrženého v jazyce PHP. Náhodným vygenerováním získáme test, který se zobrazí jako XML dokument a pro praktické využití lze výsledný test exportovat do PDF formátu.

Část práce se zabývá porovnáním produktů na generování testů, přičemž výsledný projekt bude odstraňovat nedostatky stávajících aplikací. Kapitoly věnující se popisu možných řešení exportu do XML a PHP vysvětlují použití nejlepší varianty pro závěrečný projekt. Vytvoření XML dokumentu zajišťuje třída XML Writer a export do PDF se provádí pomocí třídy FPDF. Návrh databáze popisuje tabulky v databázi a jak postupovat při návrhu databáze. Pozornost je věnována především normalizaci při návrhu databáze ve třetí normální formě. Závěr práce je věnován popisu jednotlivých částí aplikace, jsou zdůrazněny ty části kódu, které jsou důležité, nebo jiným způsobem zajímavé.

2. Cíl práce

Cílem této práce je seznámit se s problematikou generování testů a využít těchto poznatků pro navržení vlastní aplikace usnadňující vytváření testů. Práce si klade za cíl vytvořit jednoduchou aplikaci, která by odstranila nedostatky stávajících řešení generování testů a zároveň umožnila snadné a rychlé generování testových otázek a odpovědí v jednoduchém uživatelském prostředí, které se podle návodu snadno naučí ovládat i člověk bez přílišných zkušeností s používáním počítače. Generátor může výtečně posloužit a usnadnit práci nejen pedagogům na nejrůznějších školách, ale také osobám či firmám zabývajících se vytvářením testů či dotazníků z důvodu shromažďování potřebných statistických údajů.

Chceme vytvořit efektivní aplikaci pro přípravu a generování testů. Vytvořením MySQL databáze, získáme potřebné uložiště pro naplňování tabulek daty. Hlavním úkolem bude vhodně navrhnout databázi pomocí relačního modelu a vytvořit aplikaci v PHP, která bude v této databázi pracovat. Výstupem z programu budou soubory ve formátu XML a PDF. Pro vygenerování PDF dokumentu použijeme vhodnou třídu, kterou lze svobodně šířit a na rozdíl od jiných aplikací nepotřebuje licenční práva. Součástí práce je také stručná uživatelská příručka, podle které by aplikaci měl být schopen ovládat každý běžný uživatel PC. Příručka je přiložena v příloze této práce.

Mezi hlavní výhody patří, že aplikace poběží přímo na serveru, a pro tento případ si uživatel nemusí nic instalovat do počítače. Další výhodou je cena, protože MySQL a PHP jsou distribuovány jako Open Source, a jsou tedy zdarma[1].

Výsledný projekt je uživatelům k dispozici na http://generator.thosting.cz/. Pro použití se uživatel musí na uvedených stránkách zaregistrovat. Po přihlášení pomocí zvoleného emailu a hesla je možné v základním okně ukládat otázky a odpovědi do databáze a posléze z nich vygenerovat test dle momentálních požadavků na kategorii či obtížnost.

3. Metodika

V této práci využívám databázového systému MySQL, který patří k jednomu z nejrychlejších databázovým systémů. K návrhu databáze je využito pět tabulek, které budeme naplňovat daty. Jedná se o tabulky kategorie, odpověď, osoba, otázka, obtížnost.

Do tabulky kategorie se vkládají informace vztahující se k otázkám pod jednou kategorií. Tabulka odpověď uchovává informace o testové otázce, na kterou se pomocí cizího klíče vážou odpovědi z tabulky odpověď. Tabulka obtížnost informuje o obtížnosti otázky a je vázaná přímo na tabulku otázka. Poslední z tabulek osoba zaznamenává uživatele aplikace.

V PHP aplikaci generování testů využijeme funkce pro práci s formuláři, tyto funkce jsou předem vytvořené a umožní tak efektivní využití programového kódu, protože často opakující se kód nahradíme funkcí. Navržením formuláře pro naplnění tabulek otázka a odpověď, je vytvořen základ programu. Pomocí formuláře pro registraci nových uživatelů, lze přidat nové uživatele, kteří budou systém využívat, a údaje o nich se vloží do tabulky osoba. Pro přístup k databázi je vytvořena třída MojeDb, využitím objektového programování je znovu docíleno zpřehlednění kódu. Třída MojeDb také obsahuje funkce pro ladění programu.

K vygenerování testů a zamíchání otázek podle potřeby uživatele pracujeme s funkcemi pro generování náhodných čísel. Takto vygenerované otázky budou základem pro vytvoření souboru ve formátu XML a PDF.

Pro generování souboru XML je nutné si zvolit způsob zápisu, podpora v PHP je značná a záleží na vývojáři, který postup si zvolí. V tomto projektu zápis zajišťuje třída XMLWriter.

Vytvoření PDF dokumentu umožňuje mnoho externích knihoven. V tomto projektu je využita knihovna FPDF. Mezi výhody této knihovny patří, že je zdarma a pro použití nepotřebuje zkompilovat do PDF. Stačí si jí pouze překopírovat do projektu.

4. Současný stav problematiky:

4.1. Softwarová skupina G-SOFT

S generováním testových otázek a posléze jejich exportem do tiskového formátu, se zabývá několik softwarových skupin. Jedna taková se nazývá G-SOFT. Tato skupina vytvořila program na generování testů.

Popis programu:

Program Generátor testů je určen pro generování sestav testů určených pro tisk. Test je generován náhodně z databáze otázek, což zajišťuje originalitu každého generovaného testu. Program je vytvořen pro využití jak pro základní školy, tak pro střední i vysoké školy.[2]

4.2. TestGen Caretta s.r.o.

Další skupinou, která vytvořila program na generování testů, je společnost Caretta s.r.o.. Tato společnost se mimo jiné zabývá tvorbou software na zakázku a vytvořila program s názvem TestGen.

Popis programu

Program TestGen je editor testových otázek a generátor náhodných zkušebních testů pro libovolný obor, v němž je potřeba zkoušet něčí znalosti.

TestGen Editor je editor databází testových otázek. Otázky jsou seskupeny do oborů, mohou obsahovat obrázek, mají čtyři možné odpovědi, z nichž jedna nebo více může být označena jako správná odpověď. Každé otázce je přidělen počet bodů od 1 do 5 a nastavena minimální a maximální obtížnost od 1 do 9. Otázka může být také označena jako nutná (vždy bude v testu) a také jako povinná (musí se na ni správně odpovědět).

TestGen Client je generátor náhodných testů. Na základě definice testu vybere z připravené databáze potřebný počet otázek, zamíchá odpovědi (a,b,c,d) a buď vytiskne test, výsledkovou listinu a správné odpovědi a nebo otevře test k on-line zodpovězení v PC. V případě on-line testu program hlídá časový limit.

Definice testu obsahuje:

- počet otázek na obor
- obtížnost testu (v testu se pak objeví otázky, které mají minimální obtížnost menší nebo rovnu požadované a maximální obtížnost vetší nebo rovnu požadované obtížnosti)
- počet požadovaných otázek za 1 bod, 2 body, 3 body, 4 body a 5 bodů

- počet možných chyb v celém testu
- počet možných chyb v jednom oboru
- minimální počet získaných bodů na obor
- minimální počet získaných bodů na celý test
- způsob vyhodnocování při existenci více správných odpovědí na otázku
- způsob řešení nutných otázek
- způsob řešení povinných otázek

Generované testy včetně jejich výsledků jsou uchovávány v databázi je možné se k nim vracet. Výsledky vytištěných testů je možné do databáze zapsat - program pak test vyhodnotí.

Z jedné z databáze otázek lze nadefinovat několik testů, lišících se obtížností nebo způsobem vyhodnocení.

Registrační poplatek TestGen Clienta je 2000 Kč. - Neregistrovaná verze generuje pouze on-line testy, netiskne.

Registrační poplatek TestGen Editoru je 2000 Kč. - Neregistrovaná verze neumožňuje práci s obrázky v textu a vůbec nepracuje se šifrovanými testy.[9]

4.3. Porovnání programů

4.3.1. Softwarová skupina G-SOFT

Softwarová skupina G-Soft vytvořila program, kterým lze snadno vkládat a upravovat otázky a vytvořit tak test vhodný pro každou skupinu zkoušených. Tyto otázky lze řadit do sekcí a poté v sekci vytvořit předměty s otázkami. Tímto je vytvořen nástroj pro snadnou orientaci testových otázek.

Důležitou výhodou je také export a import otázek, které lze v programu využívat. Uživateli se tak naskýtá možnost vybrat si aplikaci, ve které test bude ukládat a posléze s ním pracovat.

Tento program má spoustu výhod:

- editace otázek
- import otázek do databáze přes MS World a MS Excel
- export otázek do MS World a MS Excel
- zálohování databáze

Ale i jednu nevýhodu:

- pro práci s programem, je nutné ho nainstalovat

4.3.2. TestGen Caretta s.r.o.

Na rozdíl od předešlého programu, TestGen společnosti Caretta s.r.o. umožňuje i testování on-line a posléze automatické vyhodnocení testu. Tento program je z tohoto důvodu propracovanější a skládá se ze dvou částí:

TestGen Client

Program TestGen Client slouží pro generování náhodných testů z připravených databází otázek.

TestGen Editor

Program TestGen Editor slouží pro úpravy databází testových otázek pro sesterský program TestGen Client.

Nevýhodou tohoto programu jsou registrační poplatky za možnost vytisknout test a práci s obrázky. Před použitím tento program musíme také nainstalovat, což není příliš ideální, když program pracuje s on-line testy. Lepší by bylo, kdyby programy tohoto druhu pracovaly přímo na serveru.

5. Zápis do XML

V tomto projektu se zabýváme generováním do XML. Existuje více možností jak vytvářet XML dokument a je důležité, jaký způsob generování si vybereme. Mnoho aplikací pro sdílení informací používá právě formáty založené na XML, a proto je celkem běžné, že webová aplikace potřebuje část dat, se kterými pracuje exportovat do různých podob XML[4].

5.1. Co je to XML

Zkratka XML znamená eXtensible Markup Language v českém překladu rozšiřitelný značkovací jazyk. XML je formát pro ukládání a výměnu dat a umožňuje definovat jiné jazyky. XML přináší možnost strukturování libovolných dokumentů, včetně těch textově orientovaných. XML je značkovací jazyk, což znamená, že jednotlivé části dokumentu označujeme značkami, které přesně specifikují jejich význam[8].

Příklad vygenerovaného testu v XML by mohl vypadat takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<test nazev="Název testu">
 <title>Generator testu - 1</title>
 <otazka id="4" obtiznost="4">
  <text>Na pasece se pase bílý kůň, jakou má barvu? </text>
  <odpoved spravne="ano">bílou</odpoved>
  <odpoved spravne="ne">červenou</odpoved>
 </otazka>
 <otazka id="6" obtiznost="2">
  <text>Jaký tvar má kolo? </text>
  <odpoved spravne="ano">kulatý</odpoved>
  <odpoved spravne="ne">hranatý</odpoved>
 </otazka>
</test>
```

Z kódu je patrné, jakým způsobem se oddělují části dokumentu. Struktura XML dokumentu začíná uvedením verze xml a kódováním jeho obsahu:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Samotná data se ukládají do elementů, kterým ve většině případů odpovídají dva tagy neboli značky:

```
<text>Jaký tvar má kolo? </text>
```

U každého počátečního tagu je možné použít atributy. Atributy se používají k upřesnění významu elementu, k přidání dalších důležitých informací. Hodnoty atributů se uvozují v uvozovkách a u každého elementu lze použít více atributů, které se oddělí mezerou.

Počáteční element *otazka* s atributy *id* a *obtížnost:*

```
<otazka id="4" obtiznost="4">
```

5.2. Ruční generování XML

PHP bylo původně navrženo jako šablonovaní systém, který umožňuje míchat HTML kód s programovým kódem v PHP. Protože oba formáty HTML i XML jsou založeny na prostém textu, lze samozřejmě XML generovat stejným způsobem jako HTML. Skript může obsahovat XML kód přímo, případně je možné jej generovat pomocí příkazů *echo* a podobně. Tato metoda je velmi jednoduchá a přímočará a je analogická tomu, co mnoho PHP vývojářů již zná[4].

Ukázku ručního generování demonstruje následující příklad:

```
<?php
Header("Content-type: text/xml");
echo '<?xml version="1.0" encoding="utf-8"?>', "\n";
echo '<test nazev="Název testu">', "\n";
echo ' <title>Generator testu - 1</title>', "\n";
echo ' <otazka id="4" obtiznost="4">', "\n";
echo '  <text>Na pasece běží bílý kůň, jakou má barvu?
</text>', "\n";
echo '  <odpoved spravne="ano">bílou</odpoved>', "\n";
echo '  <odpoved spravne="ne">červenou</odpoved>', "\n";
echo ' </otazka>', "\n";
echo ' <otazka id="6" obtiznost="2">', "\n";
echo '  <text>Jaký tvar má kolo? </text>', "\n";
echo '  <odpoved spravne="ano">kulatý</odpoved>', "\n";
echo '  <odpoved spravne="ne">hranatý</odpoved>', "\n";
echo ' </otazka>', "\n";
echo '</test>', "\n";
?>
```

Výstup ze skriptu se zobrazí jako text v prohlížeči. Tento skript je poměrně přehledný, nevýhodou je, že nesmíme zapomenout ošetřit escapování nebezpečných znaků a dodržet syntaxi XML – spárovat počáteční a koncové tagy, ukončit hodnotu atributu apod.[4]

5.3. Generování pomocí DOM.

Na rozdíl od ostatních XML rozhraní v PHP není DOM určen jen pro čtení dokumentů, ale i pro zápis. Nabízí proto pro práci flexibilní řešení a to díky tomu, že celý dokument je načten do paměti, jako objekty, které reprezentují jednotlivé části dokumentu. S těmito objekty můžeme dále pracovat[4]. Rozhraní DOM tak můžeme použít i pro generování XML výstupů. Výhoda tohoto přístupu je, že rozhraní se postará o mnoho věcí: escapování dat, doplnění deklarací jmenných prostor, ukončování elementů a podobně.

Rozhraní DOM se také hodí v případech, kdy dokument nevytváříme sekvenčně, ale musíme se vracet a některá data přidávat do již dříve vytvořených částí dokumentu. To je při použití DOMu možné, protože celý dokument je během vytváření držen v paměti a teprve po jeho dokončení je DOM strom převáděn do XML. Tato výhoda samozřejmě může být i nevýhodou při generování velkých dokumentů XML, protože zabírají v paměti zbytečné místo.

5.4. Využití třídy XMLWriter

Kompromisem mezi ručním generováním a použitím rozhraní DOM je třída XMLWriter. Ta by měla usnadnit generování XML a postarat se o věci, které jinak musíme řešit ručně. Přitom zachovává sekvenční přístup ke generování výstupu, takže nemá takové paměťové nároky jako DOM[4].

V této práci je zpracována varianta s využitím generování do XML s pomocí třídy XMLWriter a to zejména pro její jednoduchost. Nejprve si musíme vytvořit její novou instanci.

$xmlwriter = XMLWriter( );

Poté máme na výběr ze dvou možností, kam chceme výstup generovat. Do nějakého souboru metodou *openURI( )*, nebo do paměti *openMemory( )*.

$xmlwriter -> openURI(“php://output”);

Generování dokumentu zahájíme voláním metody *startDocumetn()*, které předáme informaci o generované verzi XML a o kódování.

$xmlwriter -> startDocument(“1.0”, ”utf-8”);

Nyní bychom mohli aktivovat odsazování výstupu pomocí volání.

$xmlwriter -> setIdent(true);

V samotném těle kódu pro vytváření XML dokumentů, voláme metody, které generují počáteční a koncové tagy, atributy, textový obsah elementů, případě komentáře a další prvky dokumentu XML. Využití těchto metod ukáže následující příklad.

```
   // vytvoření nového XMLWriter
   $xmlwriter = new XMLWriter();
   // vytvářené XML se bude zapisovat na výstup skriptu
   $xmlwriter ->openURI("php://output");
   // výstup se bude pro větší přehlednost odsazovat
   $xmlwriter ->setIndent(true);
   // zahájíme dokument a určíme kódování
   $xmlwriter ->startDocument("1.0", "utf-8");
   // vytvoříme kořenový element
   $xmlwriter ->startElementNS(null, "test", null);
     $xmlwriter ->writeAttribute("nazev", $nazev);
   // vytvoříme element title
   $xmlwriter ->writeElementNS(null, "title", null, "Generator
testu - ".$nazev_kategorie[0]);
   // pro každou otázku vytvoříme element otazka
     $xmlwriter ->startElementNS(null, "otazka", null);
     $xmlwriter ->writeAttribute("id", $id_otazka);
     $xmlwriter ->writeAttribute("obtiznost", $row[0]);
     // přidáme element text
     $xmlwriter ->startElementNS(null, "text", null);
     $xmlwriter ->text($row[1]);
     $xmlwriter ->endElement();          // </text>
    // přidáme element odpoved
               $xmlwriter ->startElementNS(null, "odpoved",
null);
               $xmlwriter ->writeAttribute("spravne",
$row_odpoved[1]);
                    $xmlwriter ->text($row_odpoved[0]);
               $xmlwriter ->endElement();          //
</odpoved>
```

```
$xmlwriter ->endElement(); // </otazka>
$xmlwriter ->endElement();   // </test>
$xmlwriter ->endDocument();  // ukončení dokumentu
```

V této ukázce je využit upravený kód přímo z programu generování testových otázek. Pro demonstraci zápisu XMLWriter, byly odstraněny části kódu, které přímo nespadají do metod pro generování pomocí XMLWriter.

6. Generování PDF v PHP

Pro vytváření souboru ve formátu PDF v jazyku PHP existuje mnoho externích knihoven. Tyto knihovny se mohou používat pod různými licencemi a některé z nich se musí nainstalovat a zkompilovat PHP. Postup jak nainstalovat knihovnu je v manuálu PHP, nebo na stránkách tvůrců dané knihovny.

6.1. Knihovna PDFLib

PDF funkce v PHP mohou vytvořit PDF soubory s využitím knihovny PDFlib vytvořené Thomasem Merzem.

PDFlib lze stáhnout na http://www.pdflib.com/products/pdflib/index.html, ale vyžaduje, abyste zakoupili licenci pro komerční použití. Pro nekomerční účely byla vytvořena verze PDFLib lite, která se dá využívat v souladu s podmínkami licenční smlouvy. Tato knihovna nabízí podmnožinu funkcí jakou má komerční knihovna PDFLib[6][7].

Příklad skriptu pro generování PDF v PHP knihovnou PDFLib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_begin_page($pdf, 595, 842);
$font = pdf_findfont($pdf, "Times New Roman", "winansi",1);
pdf_setfont($pdf, $font, 10);
pdf_show_xy($pdf, "Název testu", 100, 50);
pdf_show_xy($pdf, "Otázka 1", 50, 80);
pdf_show_xy($pdf, "Otázka 2", 50, 110);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
?>
```

Tato ukázka vytvoří soubor s názvem test.pdf, ve kterém bude nastaveno písmo na Times New Roman. Na první řádce bude napsáno *Název testu* a na dalších *Otázka 1*, *Otázka2*.

6.2. Třída FPDF

6.2.1. Popis třídy

Další možnost jak vytvořit PDF soubory je třída FPDF. Tato třída umožňuje vytvářet soubory přímo s PHP, bez použití knihovny PDFLib. Výhodou této třídy je to, že je zdarma. Můžeme jí měnit, tak aby vyhovovala našim potřebám.

Hlavní rysy FPDF jsou:

- Volba jednotek, formátu papíru a okraje
- Záhlaví a zápatí stránky
- Automaticky nová stránka
- Automatické řádky a zarovnání
- Podpora obrázků (JPEG, PNG and GIF)
- Barvy
- Odkazy
- TrueType, Type1 a podporu kódování
- Komprese stránek[18]

6.2.2. Základní metody pro práci s FPDF

Konstruktor

```
FPDF([string orientace [, string jednotka [, mixed formát]]])
```

Toto je konstruktor, pomocí něhož se dá nastavit formát stránky, orientaci a jednotku míry použitou ve všech metodách (s výjimkou velikosti písma).

Příklad pro stránku s vlastním formátem 50 na 100

```
$pdf = new FPDF('P', 'mm', array(50,100));
```

Přidání nové stránky

```
AddPage([string orientace ,[ mixed format]])
```

Přidání nové stránky do dokumentu. Parametry jsou orientace a formát stránky.

Orientace

Orientace stránky. Možné hodnoty jsou:

- P nebo Portrait
- L nebo Landscape

Implicitní hodnota je ta, která je v konstruktoru.

Formát

Formát stránky. Může být jeden z následujících hodnot:

- A3
- A4
- A5
- Letter
- Legal

nebo pole obsahující šířku a výšku

Implicitní hodnota je ta, která je v konstruktoru.

Nastavení písma

```
SetFont(string rodina fontu [, string styl [, float velikost]])
```

Nastaví písmo, které bude použito pro tisk. Tato metoda se volá ještě před tiskem, jinak bude dokument neplatný.

Může se použít písmo standardní, nebo přidat jiný font, přes metodu *AddFont()*. Takto nastavené písmo bude mít celý dokument, standardně používá kódování Windows cp1252. Pro použití českých znaků je nutné změnit kódování.

Pro změnu aktuální velikosti písma je vhodné použít metodu *SetFontSize()*.

Příklad:

```
//Times regular 12
$pdf->SetFont('Times');
//Arial bold 14
$pdf->SetFont('Arial','B',14);
//Removes bold
$pdf->SetFont('');
//Times bold, italic and underlined 14
$pdf->SetFont('Times','BIU');
```

Odeslání nebo uložení dokumentu

```
string Output([string jméno, string cíl])
```

Odešle dokument danému cíli: prohlížeč, soubor nebo řetězec. Poté je nezbytnou metodou *Close()*, která se používá k ukončení dokumentu.

Jméno

Název souboru. Pokud není uvedeno, že dokument bude zaslán do prohlížeče (cíl I s názvem doc.pdf.

Cíl

Cíl, kam poslat dokument. Můžete zvolit jednu z následujících hodnot:

- I: poslat soubor do prohlížeče. Pokud je Plug-in k dispozici. Jméno se použije, pokud je vybrána možnost "Uložit jako" pro vygenerovaný PDF dokument.
- D: poslat do prohlížeče a vynutit stažení souboru s názvem jméno.
- F: uložit jako soubor s názvem jméno (může obsahovat cestu).
- S: vrátit dokumentu jako řetězec. Jméno je ignorováno.

Př. : $pdf->Output("nazev.pdf",true) - odešle klientovy soubor nasev.pdf.

Př.2 : $pdf->Output("nazev.pdf") - uloží na serveru soubor nasev.pdf.

Př.3 : $pdf->Output() - prohlížeč načte plug-in a zobrazí obsah dokumentu PDF.

Tisk textové buňky

```
Cell(float šířka [, float výška [, string text [, mixed ohraničení [, int ln [, string zarovnání [, boolean pozadí [, mixed link]]]]]]])
```

Slouží k vytvoření textového objektu (obdélníková oblast), s volbou barvy, hranice a řetězce znaků. Horní levý roh buňky odpovídá běžné poloze. Text může být zarovnaný, nebo vystředěný. Pokud je povolené zalomování stránky a buňka se nemůže zobrazit ne stránku, je automaticky zalomena.

Šířka

Šířka buňky. Pokud je šířka 0, buňka se roztáhne až k pravému okraji.

Výška

Výška buňky. Výchozí hodnota: 0.

Text

Řetězec pro tisk. Výchozí hodnota: prázdný řetězce.

Ohraničení

Označuje ohraničení kolem buňky. Hodnota může být buď číslo

- 0: bez ohraničení
- 1: s rámem

nebo řetězec obsahující některé nebo všechny tyto znaky (v libovolném pořadí):

- L: levý
- T: horní
- R: pravý
- B: spodní

Výchozí hodnota: 0.

Ln

Signalizuje polohu, kam se bude vkládat další objekt. Možné hodnoty jsou:

- 0: doprava
- 1: na začátku další řádky
- 2: dole

Výchozí hodnota: 0.

Zarovnání

Umožňuje centrovat nebo zarovnat text. Možné hodnoty jsou:

- L nebo prázdný řetězec: zarovnání vlevo (výchozí hodnota)
- C: centrovat
- R: zarovnání vpravo

Pozadí

Označuje, jestli má mít buňka pozadí (true) nebo má být průhledná (false). Výchozí hodnota: false.

Link

URL nebo identifikátor vrácený funkcí AddLink().

Příklad:

```
//Nastavý font
$pdf->SetFont('Arial','B',16);
//Posune 8 cm doprava
$pdf->Cell(80);
//Vystředěný text, buňka o velikosti 20*10 mm, zarámovaný
$pdf->Cell(20,10,'Text',1,1,'C');
```

Přehled dalších metod a funkcí

AcceptPageBreak - přijmutí nebo nepřijmutí automatického zalomení stránky

AddFont - přidání nového fontu

AddLink - vytvoření vnitřního odkazu

AddPage - přidání nové stránky

AliasNbPages - definujte alias pro počet stránek

Cell - tisk textové buňky

Close - uzavře dokument

Error - kritická chyba

Footer - patička stránky

FPDF - konstruktor

GetStringWidth - spočítá délku řetězce

GetX - získá aktuální pozici X

GetY - získá aktuální pozici Y

Header - hlavička stránky

Image - vloží obrázek

Line - nakreslí linku

Link - vloží odkaz

Ln - zalomení řádku

MultiCell - tisk textové buňky se zalomením řádků

Output - uloží nebo odešle dokument

PageNo - číslo stránky

Rect - nakreslí obdélník

SetAuthor - nastaví autora dokumentu

SetAutoPageBreak - nastaví automatický mód zalomení stánky

SetCompression - zapne nebo vypne kompresi

SetCreator - nastaví autora dokumentu

SetDisplayMode - nastaví režim zobrazení

SetDrawColor - nastaví barvu linky

SetFillColor - nastaví barvu výplně

SetFont - nastaví font

SetFontSize - nastaví velikost fontu

SetKeywords - sloučí klíčové slova s dokumentem

SetLeftMargin - nastaví levý okraj

SetLineWidth - nastaví šířku linky

SetLink - nastaví cíl interního odkazu

SetMargins - nastaví okraje

SetRightMargin - nastaví pravý okraj

SetSubject - nastaví subjekt dokumentu

SetTextColor - nastaví barvu textu

SetTitle - nastaví název dokumentu

SetTopMargin - nastaví horní okraj

SetX - nastaví aktuální pozici X

SetXY - nastaví aktuální pozici X a Y

SetY - nastaví aktuální pozici Y

Text - tisk textu

Write - tisk "plovoucího" textu [7]

7. Zpracování bakalářského projektu

7.1.1. Hlavní část programu

Hlavní část programu tvoří formulář pro vkládání a editaci otázek. Abychom mohli vkládat údaje, musíme navázat spojení s databází. K tomuto účelu je vytvořená třída MojeDb, ve které jsou umístěny údaje pro přístup k serveru MySQL. Celý zdrojový kód, který je nutno použít pro přístup k databázi, je uložen ve vlastním souboru PHP. Tento soubor obsahuje funkce, které v projektu často potřebujeme. Využíváme přitom všech novinek objektově orientovaného programování programu PHP 5, které dovoluje vkládat zdrojový kód do vlastní třídy.[1] V hlavním formuláři lze vložit otázku a odpovědi k této otázce a zařadit otázku do kategorie, do které bude spadat. Odkazem vyhledávání se dostaneme do formuláře, který je k tomuto určený. Po výběru vyhledané otázky ji lze editovat.

7.1.2. Přihlášení a registrace uživatele

Aby bylo možné nakládat s daty, bude potřeba jednoznačně určit, který uživatel je k tomuto úkonu oprávněn.

Soubor user.php slouží k ověření uživatele, který může dále nakládat s daty v databázi. Na registraci nových uživatelů se přepneme pomocí odkazu registrace. Po úspěšném ověření skript zavolá soubor otazka.php. Prostřednictvím zpracování $_SESSION soubor otazka.php zajistí, aby bylo ověření uživatele úspěšné[1]. Takto přihlášená osoba má možnost vkládat a upravovat otázky a posléze vytvořit test z náhodně vygenerovaných otázek.

7.1.3. Generování do XML

O generování náhodně vybraného testu se stará soubor xmlwriter.php. V tomto souboru je využito třídy XMLWriter. Mezi její hlavní přednosti patří především jednoduchost, ale současná funkčnost třídy XMLWriter má mnohé limity, především v rozsáhlejších projektech a pro opravdovou použitelnost by ji bylo potřeba v mnoha ohledech vylepšit[4]. Pro naše účely však plně vyhovuje.

7.1.4. Export do PDF

Knihoven, které umožňují generování do formátu PDF prostřednictvím PHP je mnoho. V této práci je za tímto účelem použita knihovna FPDF. FPDF je již vytvořená třída. Výhoda použití knihovny fpdf.php spočívá v tom, že nepotřebujeme zakompilovanou podporu pro PDF v naší verzi PHP (např. PDFlib knihovnu). F z FPDF znamená „free“ (zdarma). Tuto

PHP třídu lze volně šířit, využívat a modifikovat. Třída FPDF je uložena v souboru fpfd.php a samotné generování obstarává soubor fpdf_generator.php.

8. Návrh databáze

8.1. Jak postupovat při návrhu databáze

Návrh databáze v tomto projektu, využívá relačního modelu dat. Základy teorie relačních databází formuloval Dr. Codd. Tuto teorii formuloval v „třech podmínkách minimálního relačního modelu“.

- Všechny údaje v databázi jsou uloženy v tabulkách.
- Fyzická struktura údajů a jejich uložení jsou nezávislé a pro uživatele naprosto odstíněné. To znamená, že z hlediska uživatele neexistují žádné viditelné přístupové cesty (včetně přístupu k indexům).
- Pro práci s údaji v databázi předpokládejme existenci databázového jazyka, který umožňuje realizovat minimálně operace výběru, omezení, projekce a spojení. [3]

8.1.1. Normalizace

Normalizace je sada pravidel, které používáme v procesu návrhu databáze. Normalizace by měla vést ke vzniku tabulek, na které se lze snadno dotazovat a udržovat jejich konzistenci. Může nastat situace, kdy vymazáním údajů o všech knihách od stejného autora, se v databázi ztratí i údaj o autorovi. Tomu se snaží zabránit proces normalizace.

Proces normalizace se vytváří a dělí na několik kroků. Tyto kroky se označují jako normální formy.

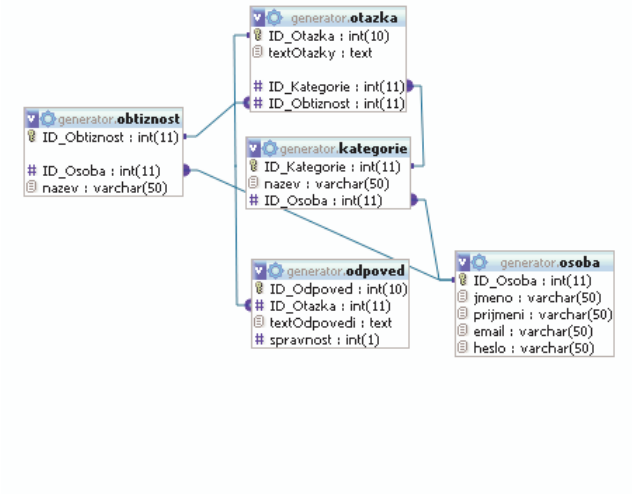
8.1.2. Normální formy

Normální formy jsou souborem testů, které se provádějí nad relačními databázemi. Podle toho, kolika testy databáze projde, můžeme potom říci, že se tato databáze nachází v určité normální formě (NF). Každou normální formu lze jednoduše popsat pomocí aspektu, který je v daném testu zkoumán. Normální formy a tyto aspekty jsou: [11]

- 1NF - atomické atributy
- 2NF - plná funkční závislost
- 3NF - tranzitivní závislot
- BCNF - přísnější 3NF (velmi speciální případy)
- 4NF - multizávislost
- 5NF - multizávislost (zobecnění)

V praxi se při rozumném návrhu struktury databáze většinou spokojíme se splněním 3NF. Navíc z důvodů zlepšení výkonu databáze vývojáři čas od času vědomě NF porušují (např. zmenšení počtu propojení mezi tabulkami při častých dotazech). [11]

8.2. Návrh vlastní databáze



Obrázek 1: Návrh databáze

Na obrázku 1 je zobrazený návrh databáze, který je použit v našem projektu. Pro práci s daty je využita MySQL databáze. Struktura byla vytvořena tak, aby splňovala 3. normální formu – tranzitivní závislosti. Tranzitivní závislost je taková závislost, mezi minimálně dvěma atributy a klíčem, kde jeden atribut je funkčně závislý na klíči a druhý atribut je funkčně závislý na prvním.

8.2.1. Tabulka otázka:

Výchozím bodem pro náš projekt jsou otázky. Data k nim jsou uložena právě v tabulce *otazka*. Přesněji řečeno patří sem přesné znění otázky *textOtazky*, cizí klíč *ID_Kategorie*, který odkazuje na kategorii otázky. Sloupec *ID_Obtiznost* odkazuje na obtížnost otázky.

8.2.2. Tabulka odpověď

Tato tabulka obsahuje seznam možných odpovědí. Cizí klíč *ID_Otazka* odkazuje na tabulku *otazka*, a vytváří relaci mezi otázkou a seznamem odpovědí. Sloupec *spravnost* určuje správnost odpovědi.

8.2.3. Tabulka kategorie

Do této tabulky se ukládají data, která určují kategorii vkládaným otázkám. To znamená množinu otázek, jenž se bere jako celek pro generování jednotlivých testů. Sloupec *ID_Osoba* zastupuje uživatele, který s touto množinou otázek dále může pracovat.

8.2.4. Tabulka osoba

Tato tabulka obsahuje přihlašovací jména a zašifrovaná hesla pro jednotlivé uživatele programu, kteří smí vkládat a editovat otázky z databáze.

8.2.5. Tabulka obtížnost

Do této tabulky lze vkládat text, který bude vyjadřovat obtížnost jednotlivých otázek. Obtížnost identifikuje osoba, která ji vytvořila. Uživatel nepotřebuje vytvářet obtížnosti pod různé kategorie.

9. Struktura zdrojového kódu

Zdrojový kód programu naleznete v souborech popsaných níže a je strukturovaný v následujících složkách a souborech:

font	Složka obsahující písma ke generování PDF
inc/password.php	Přihlašovací údaje pro přístup k souboru mojebd.php
inc/func.php	Soubor s funkcemi, která jsou použity v projektu
css.css	Soubor pro formátování
formfunctions.php	Funkce pro zobrazení formuláře
fpdf.css	Soubor pro formátování knihovny fpdf.php
fpdf.php	Knihovna pro generování PDF souborů
fpdf_generator.php	Generuje PDF soubor
index.php	Soubor pro spuštění programu
logout.php	Odhlášení uživatele
mojedb.php	Třída MojeDb pro přístup k MySQL
odpovedi.php	Odkaz na odpovědi zobrazené v tabulce
otazka.php	Zobrazení formuláře pro vkládání a editaci otázek
registrace.php	Registrace nového uživatele
test_01.php	Vygenerování testu, zobrazí v tabulce
user.php	Přihlašovací soubor pro přístup do programu
vyhledej.php	Formulář pro vyhledávání otázek
xmlwriter.php	Vytvoření XML dokumentu

9.1. Přístup k databázi, vytvoření formuláře

Všechny přístupy k databázi se provádí prostřednictvím objektu třídy MojeDb. Tato třída zabraňuje vzniku nadbytečného zdrojového kódu. Zaručuje tak využití efektivnějšího způsobu vytváření programového kódu.

Další slabostí jsou stále se opakující pasáže zdrojového kódu pro vytvoření formulářů. Proto využíváme pomocných funkcí v souboru *formfunction.php*.

Tento soubor i předešlá třída jsou součástí výukového materiálu, který je použit v knize s názvem: „PHP 5 a MySQL 5 – Průvodce webového programátora“. [1]

9.2. Významné součásti zdrojového kódu

9.2.1. Hlavní struktura programu pro vkládání a editaci otázek soubor *otazka.php*

Soubor otazka.php pro vkládání a editaci otázek načítá soubor obsahující funkce, jímž se dají zobrazit formuláře. Tento soubor *formfunction.php* spolu z dalšími soubory, které jsou přiřazeny *func.php* a *mojedb.php* si představíme v následujících kapitolách.

```
require_once 'mojedb.php';
require_once "inc/func.php";
require_once 'formfunctions.php';
```

Následující kód rozděluje činnost programu na několik větví, podle toho co bylo vybráno za akci. Následující akce zavolají jednotlivé funkce. Mezi funkce patří *precti_otazku*, *build_delete_form, delete_data, save_data*. Tyto funkce jsou nazvány tak, aby programátor pochopil jejich účel. Pro mazání a uložení otázek využíváme transakce. Transakce je vykonání programu, který přistupuje či mění data uložená v databázi. Setkáme se s nimi především ve víceuživatelských systémech řízení báze dat.

```
// upravit ID_Otazka?
if(($ID_Otazka = array_item($_REQUEST, 'ID_Otazka')) &&
is_numeric($ID_Otazka))
  $formdata = precti_otazku($ID_Otazka);
else {

  // zpracovaní dat z formuláře
  $formdata = array_item($_POST, "form");
  if(is_array($formdata)) {
    // vyhnout se magic quotes
    if(get_magic_quotes_gpc())
      while($i = each($formdata))
        $formdata[$i[0]] = stripslashes($i[1]);

    // co se má udělat?
    if(array_item($formdata, "btnClear"))
```

```
      // vymazání formuláře
      $formdata = FALSE;

    elseif(array_item($formdata, "btnDelete")) {
      // dotaz, zda se má obsah formuláře skutečně vymazat
      if(build_delete_form($formdata))
        $showform=FALSE; }

    elseif(array_item($formdata, "btnReallyDelete")) {
      // start transakce
      $db->execute("START TRANSACTION");
      if(delete_data($formdata))
        $db->execute("COMMIT");
      else
        $db->execute("ROLLBACK");
      $formdata = FALSE; }

    elseif(validate_data($formdata)) {
      // pokus o ověření a uložení dat
      $db->execute("START TRANSACTION");
      if(save_data($formdata))
        $db->execute("COMMIT");
      else
        $db->execute("ROLLBACK");
      $formdata = FALSE; }
  }
}
```

9.2.2. Pomocné funkce v souboru *func.php*

Tento soubor obsahuje funkce, které často používáme. Jedná se zejména o funkci *array_item*, která je nepostradatelná při zpracování dat z formulářů. Funkce obsahuje dva argumenty, které, pokud existují, vrací hodnotu ($ar[$key]). V opačném případě vrací false.

```
// otestování existence funkce array_item a její vrácení
function array_item($ar, $key) {
```

```
    if(is_array($ar) && array_key_exists($key, $ar))
      return($ar[$key]);
    else
      return FALSE;
  }
```

V souboru nalezneme i další funkce, jako *html_start* a *html_end*. Tyto funkce nahrazují ve všech souborech html hlavičku a uzavření html kódu.

```
  // hlavicka dokumentu HTML
  function html_start($title="no title", $css="css.css") {
    echo '<?xml version="1.0" endcoding="UTF-8"?>',
           '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"',
         '"http://www.w3.org/TR/html4/loose.dtd">', "\n",
         '<html><head>',
         '<meta http-equiv="Content-Type"',
         ' content="text/html; charset=UTF-8" />', "\n",
       '<meta http-equiv="content-language"',
         ' content="cs" />', "\n",
       '<meta name="language"',
         ' content="czech" />', "\n",
       '<meta name="country"',
         ' content="cz" />', "\n";
    if($css)
      printf('<link href="%s" rel="stylesheet"
type="text/css" media="screen">', $css);
    printf("<title>generator &ndash;
%s</title>\n</head><body>\n",
           htmlentities($title));
  }

  // paticka dokumentu HTML
  function html_end() {
```

```
    echo "</body></html>\n";
  }
```

Následují funkce pro vkládání odkazů *build_href* a *build_href_odpoved*. Význam těchto funkcí není až tolik podstatný. Zajímavější funkce jsou *get_userID* a *get_jmeno_prijmeni*. Tyto funkce spolu s funkcí *show_menu* zobrazují hlavičku stránky. A to login, jméno a příjmení uživatele.

9.2.3. Funkce pro vytváření formulářů *formfunctions.php*

Všechny formuláře, které využíváme v projektu, vytváříme pomocí tabulek. Funkce *form_start* vytvoří html tag představující začátek tabulky a tag pro vytvoření začátku formuláře. Stejně tak funkce *form_end* tabulku ukončuje. Další jednotlivé typy formulářových prvků, jsou koncipovány do buněk tabulky.

Ukázka zdrojového kódu:

```
  <tr><td align="right" class="myformtd"><span
class="red">Heslo</span></td>
  <td class="myformtd" colspan="3" ><input class="mycontrol"
type="password" name="form[heslo]" size="60" maxlength="30"
/></td>
  </tr>
```

A ukázka funkce, která tento kód vytvořila:

```
  // vytvoření políčka pro heslo ve formuláři
  function form_password($name, $value, $size=40,
$maxlength=40, $colspan=1) {
    if($colspan>1)
      echo '<td class="myformtd" ',
        html_attribute("colspan", $colspan), '>';
    else
      echo '<td class="myformtd"> ';
    echo '<input class="mycontrol" ',
      html_attribute("type", "password"),
      html_attribute("name", "form[$name]"),
      html_attribute("size", $size),
```

```
      html_attribute("maxlength", $maxlength);
    if($value)
      echo html_attribute("value", $value);
    echo ' /></td>', "\n";
  }
```

9.2.4. Vyhledávání otázek, soubor *find.php*

Abychom mohli editovat otázku, musíme otázku najít. K tomuto účelu byl vytvořen soubor *find.php*. Máme na výběr ze tří možností, jak otázky filtrovat: podle textu otázky, podle ID otázky a podle kategorie. Pro výběr podle kategorie slouží jiné tlačítko, než pro dvě předešlé možnosti výběru.

Jakmile dojde k výběru, zobrazí se otázky, které prošly filtrem, do tabulky. V pravé části tabulky je odkaz na editaci. Po výběru otázky, kterou chceme editovat, se zobrazí již zmíněná hlavní stránka programu s vyplněnými údaji o otázce.

V tomto souboru je důležitá funkce *vytvor_otazka_dotaz*. Její hlavní náplní je vytvořeni SQL dotazu, který omezuje výběr pro otázku. Další zajímavou součástí funkce je zobrazení otázek na stránku, který je limitován proměnou *$pagesize*.

```
  function vytvor_otazka_dotaz($otazkaText, $ID_Otazka,
$ID_Kategorie, $page, $pagesize) {
    global $db;
    $sql = "SELECT ID_Otazka, textOtazky FROM otazka ";
    if($ID_Otazka)
      // už jsme udělali
      return $sql . "WHERE ID_Otazka=$ID_Otazka ORDER BY
ID_Otazka";

    // přidání podmínek pro vyhledávání podle sekcí a
vyhledávání podle vzorů
    if($ID_Kategorie && $ID_Kategorie!="none") {
      $catsql = "SELECT ID_Kategorie FROM kategorie";
      $rows = $db->queryObjectArray($catsql);
      foreach($rows as $row)

    // přidání omezující klauzule
```

```
    $sql .= "LIMIT " . (($page-1) * $size) . "," . ($size +
1);
    return $sql;
  }
  }
```

9.2.5. Vygenerování otázek pro export v souboru *test_01.php*

Soubor *test_01.php* je určen pro náhodné vygenerování testových otázek. Tento krok zajišťuje SQL dotaz:

```
  $dotazy[] =  "SELECT otazka.ID_Otazka, otazka.textOtazky,
kategorie.nazev AS kategorie, obtiznost.nazev AS obtížnost
FROM otazka
  INNER JOIN kategorie ON kategorie.ID_Kategorie =
otazka.ID_Kategorie
  INNER JOIN obtiznost ON obtiznost.ID_Obtiznost =
otazka.ID_Obtiznost
  WHERE kategorie.ID_Kategorie = '".$formdata["kategorie"]."'
AND obtiznost.ID_Obtiznost = '".$row_obtiznost[1]."'
  ORDER BY rand() limit
".$formdata["pocet_".$row_obtiznost[1]];
```

Z kódu je patrné podle jakých kritérií, se otázky generují. V první řadě je to podle kategorie otázek. Dále pak podle obtížnosti. Limit vygenerovaných otázek určuje počet, který je vázán k jednotlivým obtížnostem.

Náhodně vygenerované otázky k testu by si uživatel určil podle toho, kolik otázek se má vygenerovat s obtížností například těžká, kolik s obtížností střední a kolik s obtížností lehká. Pro jednotlivé obtížnosti je potřeba spustit jeden SQL dotaz s jinými údaji. Proto se SQL příkazy ukládají do pole. Pro současné spuštění SQL dotazů existuje v PHP metoda, která se nazývá *multi_query*. Do této metody se vkládá řetězec obsahující všechny dotazy oddělené středníkem. Metoda vrací hodnotu *TRUE*, pokud se správně vykoná první příkaz, v opačném případě vrátí hodnotu *FALSE*. Návrat metody *TRUE* nedokazuje, že se bez chyby vykonali i následující příkazy.

Pro ověření vykonání všech SQL dotazů si musí programátor pomoci čítačem, který v cyklu vyhodnocení přičte jedničku, v tomto případě je čítačem proměnná *$uvolneno*.

Pomocí metody *store_result()*, se zobrazí všechny objekty typu *result* a výsledek se zobrazí v tabulce. Pomocí metody *more_results()* můžeme zjišťovat, zde je k dispozici více výsledků.

```
//SPOJENI DOTAZU POMOCI STREDNIKU
$dotazyString = Implode(';', $dotazy);
//ODESLANI DOTAZU A ZPRACOVANI VYSLEDKU
$ok = $db->multi_query($dotazyString);
$vysledek_all = Array();
$uvolneno = 0;
if($ok){
show_table_start();
do{ $result_sql = $db->store_result();
if($result_sql){
$otazky = show_table_midle($result_sql);
$result_sql->close();
$uvolneno++;
  }
} while($db->more_results() && $db->next_result());
show_table_end();
} else {
echo "Chyba v prvním dotazu ".htmlspecialchars($db-
>error());
}
```

Vytvoření odkazu, který obsahuje id ke každé zpracovávané otázce, tak abychom převzaly proměnné, pomocí kterých se bude vytvářet XML a PDF formát, zařizuje následující kód:

```
$idecka = show_id($otazky);
// odkaz na XML a PHP
$odkaz="";
for($i=0;$i<count($idecka);$i++)
{
  $id=$idecka[$i];
  if($i==0) $odkaz.="id$i=$id";
  else $odkaz.="&amp;id$i=$id";
```

```
    //echo $odkaz;
  }
  build_href_odpoved("xmlwriter.php",$odkaz, "Xml
Generator");
  build_href_odpoved("fpdf_generator.php",$odkaz, "PDF
Generator");
```

Pro vkládání údajů do formuláře musí ověřit, zda jsou kolonky vyplněné a obsahují správná data. K tomuto účelu slouží metoda *validate_data($formdata)*, ve které jsou vstupem hodnoty ve formuláři. Následující ukázka je část kódu z této metody, která ověřuje údaje a počet obtížností. Součet otázek jednotlivých obtížností se musí rovnat počtu otázek v testu.

```
  // Obtížnosti
    $sql = /*"SELECT nazev, ID_Kategorie, ID_Osoba FROM
kategorie ORDER BY nazev";*/
     "SELECT nazev, ID_Obtiznost FROM obtiznost".
               " WHERE ID_Osoba = ".$db-
>sql_string($_SESSION['ID_Osoba']);
    $result = $db->query($sql);
    $citac = 0;
    while($row_obtiznost = $result->fetch_array()) {
    $citac = $citac + $formdata["pocet_".$row_obtiznost[1]];
    }
    if($formdata["pocet"] != $citac){
    show_error_msg("Musíte zadat správný počet otázek
jednotlivých obtížností!");
      $result = FALSE;
    }
```

9.2.6. Vytvoření XML dokumentu v souboru *xmlwriter.php*

Soubor *xmlwriter.php* vytváří dokument ve formátu XML. Tento dokument si lze poté uložit na disk a jeho význam je posléze vázán při dalším zpracování testu. Tato práce s dalším využití tohoto dokumentu nepočítá. Je důležité uložit test ve formátu, který umí zpracovávat ostatní programovací jazyky.

Při vytváření XML se využívá třídy XMLWriter. Její hlavní funkce jsou: *startDocument, endDocument, startElementNS, endElement, writeAttribute, writeElementNS.* Těmito funkcemi se docílí stromové struktury XML dokumentu.

Dokument vytvořený právě tímto projektem vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<test nazev="Název testu">
 <title>Generator testu </title>
 <otazka id="4" obtiznost="těžká">
  <text>Otazka 0 - </text>
  <odpoved spravne="ano">Odpoved 1</odpoved>
  <odpoved spravne="ano">Odpoved 2</odpoved>
   </otazka>
 <otazka id="1" obtiznost="lehká">
  <text>Otazka 1 - </text>
  <odpoved spravne="ano">Odpoved 1</odpoved>
  <odpoved spravne="ne">Odpoved 2</odpoved>
  <odpoved spravne="ano">Odpoved 3</odpoved>
 </otazka>
</test>
```

9.2.7. Soubor pro generování PDF dokumentu *fpdf_generator.php*

V této části se seznámíme s tím, jak funguje knihovna FPDF. Po vytvoření instance třídy FPDF je důležité nastavit font, kterým bude psán text v dokumentu.

Adresář font obsahuje několik základních písem, které je možné pro generované PDF používat a přepínat se mezi nimi. Tyto fonty nepodporují české znaky. Proto je potřeba vygenerovat vlastní písmo s podporou českých znaků. Generovat vlastní písmo můžete přímo pomocí přiložené třídy MakeFont, nebo jej vygenerovat na internetu například na adrese fpdf.fruit-lab.de (tato, druhá, možnost je jednodušší). Konvertor je schopný vygenerovat písmo použitelné pro FPDF knihovnu z jakéhokoli TTF písma[10].

Ukázka funkcí pro vytvoření a nastavení písma s českými znaky.

```
//Instance dědičné třídy
$pdf=new FPDF();
$pdf->AliasNbPages();
```

```
    $pdf->AddPage();
    //$pdf->SetFont('Times','',12);
    $pdf->AddFont('muj_font', '',
'86578d8876402ecd3f28cdcb3fe3520e_sabiuteemed.php');
    $pdf->SetFont('muj_font','',12);
```

Další funkcí, která je využívána v tomto souboru je funkce *Cell*. Tato funkce tiskne textovou buňku do řádků. Má několik atributů, kterými se dají řádky definovat.

```
    Cell($w, $h=0, $txt='', $border=0, $ln=0, $align='',
$fill=false, $link='')
```

Příklad vkládání odpovědí po jednotlivých řádcích, si nejprve pomocí jazyka SQL vybereme požadované odpovědi z databáze a následující cyklus zobrazí odpovědi pomocí funkce *Cell()* v řádcích:

```
    $odpoved_result = $db->query("SELECT odpoved.textOdpovedi,
odpoved.spravnost FROM odpověď WHERE odpoved.ID_Otazka =
'".$id_otazka."'ORDER BY ID_Odpoved ASC");

    while($row_odpoved = $odpoved_result->fetch_array()) {
        $pdf->Cell(20);
        $x = $pdf->GetX();
        $y = $pdf->GetY();
        $pdf->Rect($x+150, $y, 4, 4);
        $pdf->Cell(0,5, iconv("UTF-8", "WINDOWS-1250",'* )
'.$row_odpoved[0]),0,1);
         }
```

Další významnou funkcí je *Rect()*, s jejíchž pomocí lze nakreslit obdélník. Této funkce využijeme při vytváření kolonek pro označení správné odpovědi.

Nejprve si zjistíme pozici x a y a posléze zavoláním funkce *Rect ()* vložíme tyto hodnoty jako parametry funkce. Následují parametry pro výšku a šířku obdélníku.

```
$x = $pdf->GetX();
$y = $pdf->GetY();
$pdf->Rect($x+150, $y, 4, 4);
```

Popis funkce:

```
Rect(float x, float y, float šířka, float výška [, string styl])
```

Parametry:

x

Souřadnice x levého horního rohu.

y

Souřadnice y levého horního rohu.

šířka

Šířka.

výška

Výška.

styl

Styl vykreslování. Možné hodnoty jsou:

- D nebo prázdný řetězec: kreslení. Toto je výchozí hodnota.
- F: výplň
- DF nebo FD: kreslení a výplň [7]

10. Závěr

Cíle práce se podařilo splnit, takže výsledkem této bakalářské práce je fungující aplikace pro generování testových otázek a odpovědí. Uživatel má možnost nastavit generování podle předem definovaných kritérií a určit si tak rozsah obtížnosti jednotlivých otázek, což dává větší možnosti při vytváření testů.

Projekt byl koncipován tak, aby uživatelům usnadňoval práci s aplikací, byl pochopitelný i pro uživatele s pouze základními znalostmi ovládání počítače a zamezoval nadbytečnému klikání. Proto a také kvůli přehlednosti se například ve formuláři pro vkládání otázek dají vložit i odpovědi a přiřadit kategorie k otázce, včetně definování nové obtížnosti.

Prozatím aplikace nepracuje s grafickými objekty a neumožňuje vkládat obrázky k jednotlivým otázkám. Vložení obrázku by z části vyřešilo další nevýhodu, kterou je nemožnost vložení matematické rovnice s lomenými výrazy. Dosavadní řešení dále nepočítá s importem otázek, což znesnadňuje přenositelnost uživatelem již vytvořených testů v jiných programech do této aplikace.

I přesto je však program vhodným pomocníkem nejen pro přípravu výuky se zaměřením na otestování znalostí studentů, ale také pro vytváření dotazníků sloužících k zjišťování statistických údajů.

11. Literatura

- [1] KOTLET, Michal; OGGL, Bernd. *PHP 5 a Mysql 5 : Průvodce webového programátora*. [s.l.] : Copputer Press, a. s. , 2005. 607 s. ISBN 978-80-251-1813-9.
- [2] Manuál k programu Generátor testů G-SOFT. In [online]. [s.l.] : [s.n.], 2004 [cit. 2010-11-04]. Dostupné z WWW: <http://www.g-soft.wz.cz/files/products/programs/manual_testy.pdf>.
- [3] LACKO, Luboslav. *SQL : Kapesní přehled*. Brno : CP Books, a.s., 2005. 96 s. ISBN 80-251-0788-4.
- [4] KOSEK, Jiří. *PHP a XML*. První vydání. Praha 7 : Grada Publishing, a.s., 2009. 367 s. ISBN 978-80-247-1116-4.
- [5] Manuál PHP : XCVII. PDF funkce. In [online]. [s.l.] : [s.n.], 2010 [cit. 2010-11-05]. Dostupné z WWW: <http://jonatan.spse.pilsedu.cz/doc/php-man/ref.pdf.html>.
- [6] *Source Code of PDFlib Lite 7* [online]. 2010 [cit. 2010-11-05]. PDFLib. Dostupné z WWW: <http://www.pdflib.com/download/pdflib-family/pdflib-lite-7>.
- [7] FPDF knihovna - PDF generátor. In [online]. [s.l.] : [s.n.], 2008 - 2010 [cit. 2010-11-05]. Dostupné z WWW: <http://www.fpdf.cz/>.
- [8] KOSEK, Jiří Lehký úvod do XML. In [online]. [s.l.] : [s.n.], 2001 [cit. 2010-11-05]. Dostupné z WWW: <http://www.cstug.cz/slt/01/plne_texty/13.pdf>.
- [9] TestGen : Popis programu. In [online]. [s.l.] : [s.n.], 2010 [cit. 2010-11-05]. Dostupné z WWW: <http://www.caretta.cz/software/testgen/default.asp?str=200511262302270>.
- [10] FPdf : jak na české znaky. In [online]. [s.l.] : [s.n.], 2008 [cit. 2010-11-08]. Dostupné z WWW: <http://www.zaachi.com/cs/items/fpdf-jak-na-ceske-znaky.html>.
- [11] Normální formy : Co jsou normální formy. In [online]. [s.l.] : [s.n.], 2002-2010 [cit. 2010-11-08]. Dostupné z WWW: <http://www.eamos.cz/amos/kat_inf/modules/low/kurz_text.php?id_kap=3&kod_kurzu=kat_inf_68219>.

Výčet příloh

Součástí této práce je přiložené datové DVD s následujícím obsahem:

- kopie této práce ve formátu PDF
- uživatelská příručka ve formátu PDF
- adresář se zdrojovými kódy generátoru testových otázek