ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta strojní - Ústav přístrojové a řídicí techniky

Nová metoda převodu obecného XML na čitelný PDF formát

DIPLOMOVÁ PRÁCE

prosinec 2006 **Kamil Mrázek**

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího diplomové práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků diplomové práce nebo její podstatné části, pokud budu uveden jako její spoluautor.

V Praze dne 11. prosince 2006

Děkuji vedoucímu diplomové práce Doc. Mgr. Ing. Petru Klánovi, Csc. za odborné vedení diplomové práce a vytvoření podmínek (webserver a XML rozhraní) pro tvorbu praktické úlohy.

Anotace:

Použitím existujících knihoven v PHP 5 na převod do PDF a čtení XML dokumentů byla vytvořena nová metoda automatického převodu obecného XML na čitelný PDF přes webový server. Zahrnuje varianty čtení XML pro jednotlivé funkce. Dále převod, vkládání textu a obrázků do PDF, tvorba grafu (obrázku) pomocí GD Graphics Library z daných hodnot, jeho vzorkování. Aplikace metody popsána na konkrétní úloze měření teploty, kde se využívá SimpleXML a knihovna FPDF. Doplňkovou praktickou úlohou je převod XML souboru struktury MathML do PDF taktéž přes PHP5.

Summary:

By using existing libraries in PHP 5 format to transfer to PDF format and to read XML documents there is a new method of file transfer from general XML format to readable PDF format via web server, includes the XML reading options for individual uses. Also transfer, text and images inserting to PDF, charts (images) creating by the help of GD Graphics Library from given values, sampling. Application of method is described on particular case of temperature measuring, where SimpleXML and FPDF library is used. As a supplemental practical case is a XML file (MathML structure) transfer to PDF, similarly through PHP5.

Obsah

1 Úvod

Metod a aplikací na převod do PDF je mnoho. Dokumenty, tabulky, formuláře, to vše a mnoho dalšího se dnes převádí do tohoto univerzálního formátu. V diplomové práci chci poukázat na další možnou aplikaci převodu do PDF sloužící k technickému použití.

Cílem této diplomové práce je vytvořit novou metodu automatického převodu obecného XML do čitelného PDF formátu, navrhnout a popsat možné postupy, jak přes webový server s PHP 5 číst XML dokumenty, vytvářet PDF, to vše pomocí existujících funkcí a knihoven. Proč zrovna XML, PHP, PDF?

XML je nejslibnější jazyk pro uchování a výměnu informace na webu, je prezentován jako obecný jazyk. Umožňuje vytvářet vlastní elementy, atributy a strukturu dokumentu, to znamená, že se může použít k popsání prakticky jakéhokoli typu informace – od jednoduchých postupů až po komplexní databáze. Využívá se k ukládání informace třeba z měřicích jednotek proto, aby následně mohly být zveřejněny údaje například na internetu.

PHP je nejrozšířenější skriptovací jazyk na webu. Autoři tohoto jazyka si uvědomili velké možnosti a rozmach aplikací na bázi XML a zabudovali od verze 5 nativní podporu jazyka XML a jeho čtení několika různými způsoby, které jsou popsány v této práci.

Formát PDF znají všichni, jeho výhodou je ukládání dokumentů nezávisle na softwaru i hardwaru, na kterém byly pořízeny. Soubor typu PDF může obsahovat text i obrázky, přičemž tento formát zajišťuje, že se libovolný dokument na všech zařízeních zobrazí stejně a je uživatelsky čitelnější než XML. Využívá se prakticky všude.

Jedna z aplikací metody může vypadat tak, že měřicí jednotka ukládá hodnoty do XML na webový server. Tato struktura se může v pravidelných intervalech z jiného serveru ukládat do souborů, které jsou následně zpracovány.

Je několik variant, jak zpracovat dané údaje:

1. přečíst hodnoty v XML souborech a napoprvé je všechny uložit do databáze, poté pracovat s hodnotami v databázi a vytvořit výstup do souboru PDF.

2. bez použití databáze a to třeba ukládat hodnoty z XML souboru do pole a následně vytvořit PDF výstup s hodnotami v poli. Výhody této varianty jsou jasné, funkčnost bez instalace databáze a rychlost zpracování. Nevýhodou jsou trochu

komplikovanější algoritmy a také při velkém množství údajů je výhodné si vše ukládat jak do databáze, tak do PDF, pro budoucí zpracovávání.

Další varianty jsou ukládat data např. do textového souboru.

V této diplomové práci je volena varianta druhá, bez použití jakéhokoliv dalšího mezikroku, tzn. bez databáze. V praktické části je vytvořena webová aplikace, kde se navržené funkce a metody aplikují na konkrétní měření a regulaci vybraného laboratorního modelu. Další praktickou úlohou je převod MathML do PDF pro ilustraci, že navržené funkce a metody mají široké a obecné uplatnění.

2 Seznámení s XML, PDF

2.1. Historie XML

XML vzniklo z dnes existujícího standardu SGML (Standard Generalized MarkUp Language). Jde o nejobecnější značkovací jazyk. SGML je však jazykem komplikovaným na implementaci a obsahuje mnoho rysů, které se využijí jen málokdy. Podpora SGML pro rozdílné znakové sady je mizivá, způsobuje problémy tam, kde lidé používají rozdílné jazyky. Rovněž interpretace SGML dokumentu je složitá v případě, kdy nemáme k dispozici definici značkovacího jazyka (tzv. DTD). Na základě těchto okolností byla vyvinuta zjednodušená verze jazyka SGML, kterou nyní nazýváme XML. Tento jazyk není majetkem žádného komerčního subjektu. Je to otevřený a veřejný standard, který byl vytvořen a je nadále spravován a rozvíjen konsorciem W3C - více na www.w3.org/xml.

Toto konsorcium popsalo jazyk takto: „XML je podmnožina SGML... Jeho cílem je umožnit použití SGML k posílání, přijímání a zpracování dat na webu, stejně tak jako je tomu u HTML. XML je navržen pro snadnou implementaci a spolupráci SGML a HTML.“

2.2. Co je XML?

Zkratka XML znamená eXtensible Markup Language neboli rozšiřitelný značkovací jazyk. Slovo jazyk předjímá, že se jedná o strukturovaný formát, kterým je možné zpracovávat, uchovávat a předávat informace. To, že je značkovací, vyjadřuje způsob záznamu, který jednotlivé části informace umisťuje mezi speciální významové značky, tzv. tágy. Přívlastek rozšiřitelný znamená, že na rozdíl například právě od HTML není jeho syntaxe pevná a předem daná, ale naopak velmi pružná a přizpůsobitelná. U HTML jsou významové značky dány příslušnou normou a jejich význam je přesný a jednoznačný.

2.3 Základní struktura XML

Důležité části dokumentu se označují pomocí tzv. tágů. V terminologii XML se jednotlivým označeným částem dokumentu říká elementy. Elementy do sebe mohou být navzájem vnořené a tím dle potřeby zachycovat strukturu informace uložených v dokumentu.

2.3.1 Elementy

Většině elementů odpovídají dva tágy – počáteční a ukončovací.

Příklad:

```
<misto>Praha</misto>
```

V příkladu je ukázka definování elementu „misto“.

Struktura XML dokumentu začíná uvedením verze xml a kódováním jeho obsahu:

```
<?xml version="1.0" encoding="windows-1250" ?>
```

Pokud element nemá žádný obsah, nemusí se zbytečně zadávat počáteční a ukončovací tág, ale za jméno počátečního tágu se uvede znak „/“.

Příklad:

```
<misto>Praha <br></br> Dejvice </misto>
```

Příklad bez použití ukončovacího tágu:

```
<misto>Praha<br/> Dejvice</misto>
```

2.3.2 Atributy

U každého počátečního tágu je možné použít ještě atributy. Atributy se používají k upřesnění významu elementu, k přidání dalších důležitých údajů, hodnota je vždy v uvozovkách nebo apostrofech. U jednoho tágu je možné použít více atributů najednou, stačí je oddělit mezerou.

```
<misto autor="Kamil" mereni="ano">Praha </misto>
```

V příkladu je uvedena k atributu „autor“ hodnota „Kamil“ a k atributu „mereni“ hodnota „ano“

2.3.3 Znakové entity

Vzhledem k tomu, že se znaky `<' a `>' používají pro oddělení tágů od okolního textu, není možné tyto znaky zapsat do dokumentu jen tak. Pro jejich zápis musíme použít tzv. znakové entity. Například pro zápis znaku `<' je určena entita < a pro `>' to je >. Pokud potřebujeme uvnitř hodnoty atributu použít zároveň uvozovky i apostrofy, s výhodou využijeme odpovídající entity " a '.
Příklad:

```
<teplota rano="15&deg;"/>
```

Použití entit " a ' se někdy můžeme vyhnout použitím apostrofů pro oddělení obsahu atributu:

```
<monitor uhlopricka='15"'/>
```

Seznam veškerých znakových entit je uveden v Příloze 6.

2.3.4 Kořenový element

Každý XML dokument musí být celý obsažen v jednom elementu. Následující ukázka tedy není správný XML dokument, protože se skládá z několika samostatných elementů.

```
<misto>Praha</misto>
<rano>5</rano >
<odpoledne>23 </odpoledne>
<vecer>16</vecer>
```

Stačí však přidat kořenový element, který vše „obalí“ a dokument je již v pořádku.

```
<teplota>
<misto>Praha</misto>
<rano>5</rano >
<odpoledne>23 </odpoledne>
<vecer>16</vecer>
</teplota>
```

Splňuje-li dokument všechna výše uvedená pravidla, je syntakticky v pořádku a říká se o něm, že je správně strukturovaný (well-formed). Takový dokument se může směle vypustit do světa, protože si s ním poradí všechny aplikace podporující formát XML.

2.4 Binární data v XML

Binární data jsou informace, které se uchovávají v počítači jako 1 a 0. Binární soubor se skládá z osmibitového uskupení, dobře známého jako byte. Při surovém prohlížení daného binárního souboru se jeho obsah jeví jako nesmyslný mix znaků, je to dáno tím, že počítač se tyto data snaží převést do textové podoby. Binární soubory jsou těžko přenositelné mezi různými platformami, nedají se přečíst bez podpůrného softwaru a každá platforma je zobrazuje jinak. Mezi binární data patří např. obrázky.

Naproti tomu je standardní text. Data ve formě textu se vyjadřují ASCII kódem, což je americký standard pro přenos informace. Textová data obsahují znaky bez diakritiky a číslice. Takovéto vyjádření dat je snadno přenositelné, čitelné a neměnné při přenosu a zobrazování mezi různými platformami.

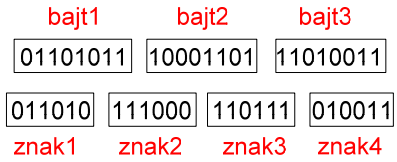
Existují dva základní způsoby jak v XML vyjádřit binární data. XML dokumenty jsou však čistě textové soubory, proto nikdy nelze binární data ve své původní podobě uložit přímo do XML dokumentu. Existují tedy následující možnosti jak asociovat s XML dokumentem binární data:

- zakódovat binární data do textového řetězce
- použít externí binární entitu

Zakódování binárních dat do textového řetězce

Data mohou být zakódována různými způsoby – např. kódováním Base64 či hexadecimálně. Pro názornost následuje princip nejčastěji používaného kódování binárních dat v XML.

Kódování označované jako Base64 slouží k transformaci (mapování) binárních dat do čistě textové podoby. Proud dat vzniklý zakódováním binárního obsahu je pak možné přímo vložit do XML dokumentu. Princip tohoto kódování je jednoduchý. Skupina 24 po sobě jdoucích bitů binárního obsahu (3 bajtů, oktetů) je rozdělena do 4 šestibitových skupin. Tyto šestibitové bloky jsou poté vyjádřeny jim odpovídajícím textovým znakem. Proto se tento přístup v literatuře někdy označuje jako „3-to-4".



obr. 1 - princip Base64 kódování

XML Schéma obsahuje datové typy nazvané „base64Binary" a „hexBinary", které vyjadřují, že hodnota je kódována příslušným kódováním:

```
<xs:element name="mapa" type="xs:base64Binary"/>

<dnes>
   <misto>Praha</misto>
<mapa>d2hhdCB3aWxsIHByaW50IG91dAbCAgIHByaW50ICAgICBvdXQgIC
AgICbCAgIHBhdCB3aWxsIHByyaW50ICAgICBvdXQgICAgICAbChdCB3aWx
sIHByAgIHByaW50ICAgICBvdXQgICAgICAbCAgIHBhdCB3aWxsIHByyaW5
0ICAgICBvhdByaW50ICAgICB....  </mapa>
</dnes>
```

Tímto způsobem je tedy možné, aby byl binární obsah uložen přímo v XML dokumentu. Otázkou často je, zda by binární obsah nemohl být uložen v XML dokumentu v sekci CDATA. Odpověď na tuto otázku zní, že nemohl. Veškeré znaky, které může binární proud dat obsahovat totiž nejsou validními znaky pro použití v XML. Navíc by tato binární data mohla obsahovat sekvenci znaků „]]>“, která slouží pro ukončení CDATA sekce. V CDATA oblasti by binární data mohla být pouze, pokud by předtím byla zakódována do textového řetězce, např. způsobem, který zde byl demonstrován.

Ukládání binárních dat přímo do dokumentů by se mělo používat spíše pro malé objemy dat. V opačném případě je lepší využít externích entit, které jsou pro tento účel vhodnější.

Binární entity

XML umožňuje informace obsažené v dokumentu rozdělit na menší části, kterým se říká entity. Každá entita má své jméno, pomocí kterého může být jednoznačně identifikována. XML podporuje několik druhů entit, které se liší svými vlastnostmi. Entity mohou obsahovat buď data ve formátu XML nebo v jiném formátu. Podle toho je dělíme na entity textové a entity binární. Dále se mohou entity rozlišovat podle toho, zda jsou uloženy přímo v hlavním dokumentu nebo v externím souboru. Tudíž existují entity interní textové, externí textové a externí binární. Interní binární entity neexistují, protože se těžko uloží nějaká binární data do XML dokumentu, protože jeho struktura je textová.

Pro připojení binárních dat k dokumentu slouží v XML tedy externí entity a datový typ NOTATION. Do XML dokumentu se pak vkládá pouze URI (Uniform Resource Identifier - obecně použitelná množina všech jmen/adres, které se vztahují k nějakému zdroji) odkazující na zdroj binárních dat. V této souvislostí je potřeba si vysvětlit co je DTD?

Pomocí XML je možné vytvářet vlastní jazyky, které používají syntaxi XML. V jazyku odvozeném od XML se definuje, které elementy a atributy budou k dispozici, a jak se mohou navzájem kombinovat. Této definici se říká DTD – definice typu dokumentu. Má odlišnou strukturu než jednoduchý XML dokument, viz ukázkový příklad na následující stránce.

Entita je definována v DTD a do dokumentu se vkládá pomocí odkazů na požadovanou entitu. V textu může být použito několik odkazů na stejnou entitu a její obsah nahradí každý výskyt odkazu. U deklarací entit je jedno, zda jsou umístěny v lokálním či externím DTD dokumentu.

V DTD by definice entity vypadala takto:

```
<!—definovani entity-->
<!ENTITY mapa SYSTEM ″http://server/mapa.jpg″ NDATA ″jpg″>

<!—definovani typu entity-->
<!NOTATION jpg ″urn:mime:img/jpg″>

<!—definovani elementu obrazek -->
<!ELEMENT obrazek EMPTY>

<!—definovani atributu -->
<!ATTLIST obrazek zdroj ENTITY #REQUIRED>
```

Typ dat v entitě je určen v definici entity za textem „NDATA“. Tento typ musí být někde definován jako tzv. notace (NOTATION). Dále je v tomto příkladu element „obrazek“, který obsahuje ENTITY atribut „zdroj“.

V XML dokumentu by na tuto entitu bylo odkazováno následovně:

```
<obrazek zdroj=″mapa″ />
```

Podobnou funkcionalitu jako poskytují externí entity a notation v DTD se nalezne i v XML Schématech. Zde je přítomen datový typ „anyURI“, který reprezentuje absolutní či relativní URI. V XML Schématu se tedy nadefinuje element či atribut, který by byl typu anyURI a do obsahu tohoto elementu/atributu se uvede URI externího zdroje (binárních) dat.

2.5 Historie PDF

V roce 1993 přišla společnost Adobe s PDF (Portable Document Format) coby formátem, jež umožňuje věrnou reprodukci dokumentů mezi různými platformami (tzv. kancelář bez papíru). Cílem bylo poskytnout formát, který by umožňoval obdobné vyjadřovací možnosti jako Postscript (první verze PDF se jmenovala Interchange PostScript), ale na rozdíl od uvedeného formátu byl skutečně nezávislý na aplikaci a operačním systému, nevyžadoval náročný interpret pro vykreslení a měl výrazně menší objem souborů. Pro práci s PDF nabídla Adobe postupně různé nástroje, které posléze vykrystalizovaly do trojice Acrobat (prohlížení, tisk a editace PDF souborů), Distiller (tvorba PDF převodem z PostScriptu) a Reader (prohlížení a tisk PDF na různých platformách). Schopnost PDF reprodukovat tentýž obsah shodně bez ohledu na platformu a i skutečnost, že Reader byl posléze poskytnut uživatelům zdarma (původně šlo o komerční aplikaci), vedly k tomu, že se uvedený formát stal oblíbeným prostředkem zejména pro výměnu dokumentů v prostředí Internetu.

2.6 Vlastnosti PDF

PDF byl navržen tak, aby při jeho výstupu či zobrazení nehrály prakticky žádnou roli na zařízení závislé charakteristiky. Kontejnerová struktura formátu pak dovoluje do jeho dokumentů vkládat použitá písma, takže PDF dokument je možno v odpovídajícím prohlížeči shodně zobrazit na libovolném počítači bez ohledu na použitý operační systém. Odpovídající nástroje dovolují PDF vytvořit z prostředí libovolné aplikace, takže příjemce PDF dokumentu může prohlížet a tisknout obsažené informace bez toho, že by disponoval aplikací, ve které byl původně dokument vytvořen - stačí příslušný PDF prohlížeč (Adobe Reader aj.).

PDF nabízí velmi bohaté vyjadřovací vlastnosti pro text i grafiku. V případě písem jsou podporovány všechny základní standardy daného typu (Type 1 a 3, TrueType, OpenType), formát pak dokáže nést jak bitmapové obrázky, tak vektorové objekty různých typů, má širokou podporu barev. PDF nabízí také silnou podporu interaktivních prvků pro elektronické publikování, zejména odkazů a záložek, velmi propracovaná a pro mnohé aplikace formátu zásadní (podniková sféra) je podpora elektronických formulářů, k dispozici jsou rovněž funkce pro elektronické prezentace. Dále je možné v PDF využít nejrůznějších komentářů a zvýraznění, dovolujících (spolu s příslušnými

aplikacemi) skupinovou spolupráci nad dokumenty v uvedeném formátu. Formát dokáže nést prakticky libovolný další obsah (kontejnerová struktura), čehož je dnes u něj využito zejména v případě multimédií, souborových příloh a metadat (formát XMP, PJTF či JDF jobtikety). V rámci PDF je pak použitelné i zabezpečení, dovolující různým způsobem omezit zpracování dokumentu (čtení, tisk, úpravy, kopírování obsažených dat aj.), spolu s tím je podporována technologie elektronického podpisu. K dispozici je i technologie tágovaného PDF, dovolující využít PDF na mobilních zařízeních nebo čtecích zařízeních pro zrakově postižené osoby.

3 XML a PDF v PHP 5

PHP je serverový skriptovací jazyk (server-side) navržený pro potřeby webových stránek, což znamená, že vše co PHP dělá, probíhá na straně serveru, ne na straně klientských stanic. PHP je Open Source, tedy volně šiřitelná technologie, není závislé na platformě a není vázané s žádným konkrétním serverem, může tedy běžet kdekoli.

Tuto kapitolu věnuji pozornosti nové podpory XML v PHP od verze 5. PHP sice podporovalo XML již ve verzi 4, ale zpočátku šlo pouze o interface založený na SAXu (Simple API for XML), který neumožňoval více, než parsování XML dokumentů. Další rozšíření, jako byla HTML, XSLT (Extensible Stylesheet Language Transformations) a DOM validace přišly s rozšířením domxml. Bohužel domxml používalo namísto jmenných prostorů definovaných W3C své vlastní metody a nikdy se nepropracovalo k stabilní verzi. Během své existence domxml rovněž několikrát změnilo API rozhraní, a proto nebylo běžně aktivováno. Na celé řadě serverů tedy nikdy nebylo nainstalováno.

Podpora XML byla pro PHP5 téměř kompletně přepsána. Všechny XML rozšíření jsou nyní založeny na libxml2 knihovně projektu GNOME. Podpora SAX známá z PHP4 zůstává, ale navíc je dodána podpora DOM modelu podle W3C a XSLT transformace pomocí libxslt enginu. Defaultně podporovány jsou DOM, SAX a SimpleXML (zvláštní formát PHP). SimpleXML umožňuje přistupovat k XML datům jako do pole nebo k objektům. Iterace se provádí za pomoci foreach konstrukcí a změna hodnoty pouhým přiřazením nové hodnoty.

3.1 DOM (Document Object Model)

Aplikace využívající SAX mohou zůstat beze změny, ale podpora DOMu byla kvůli problému s jmennými prostory zcela přepracována a proto je nutné aplikace předělávat. Práce s DOM modelem v PHP5 může vypadat např. takto (vzorový XML soubor.xml):

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<mereni>
    <item>
        <misto>Praha</misto>
        <teplota>23</teplota>
```

```
    </item>
    <item>
        <misto>Brno</misto>
         <teplota>25</teplota>
    </item>
</mereni>
```

Následujícím kódem pak je možné získat všechny místa:

```
$dom = new DomDocument();
$dom->load("soubor.xml");
$mista = $dom->getElementsByTagName("misto");
foreach($mista as $node) {
   print $node->textContent . "\n";
}
```

Vlastnost textContent však není standardem W3C, ale vlastností, která má za cíl ulehčit přístup ke všem textovým položkám elementu. DOM (Document Object Model) lze nyní nejen číst a dotazovat se na něj (do něj), ale také s ním manipulovat a zapisovat do něj. Stejně jako u chybové třídy Exception lze i DOM třídy, např. DomDocument rozšiřovat a vkládat do nich vlastní kód.

Validace

Validace XML dokumentů může být životně důležitá. PHP5 podporuje validaci pomocí tří důležitých standardů.

- DTD - Starší standard z doby SGML, postrádá jmenné prostory a jiné novější funkce. Není zapsán v XML, takže není příliš snadno parsovatelný.
- XML Schema - Standard W3C. Velmi obsáhlý.
- RelaxNG - Reakce na XML Schema, vytvořen nezávislou skupinou. Velice oblíbený, protože se implementuje snadněji než XML Schema.

Syntaxe pro validaci je velice jednoduchá:

```
$dom = new DOMDocument;
$dom->load('soubor.xml');
$dom->validate('soubor.dtd');
$dom->relaxNGValidate('soubor.rng');
$dom->schemaValidate('soubor.xsd');
```

Tyto funkce nyní vrací pouze true nebo false. Do dalších verzí PHP se však chystá se rozšíření spektra chybových hlášek pro parsovací chyby.

3.2 SimpleXML

SimpleXML je nejnovějším přírůstkem do XML v PHP. Cílem rozšíření SimpleXML je poskytnout snadný přístup k XML dokumentům pomocí standardních objektových vlastností a iterátorů. SimpleXML neobsahuje příliš metod, ale je poměrně mocným nástrojem. Kratičký příklad se stejnou funkčností (viz. výše):

```
$sxe = simplexml_load_file("soubor.xml");
foreach($sxe->item as $item) {
    print $item->misto ."\n";
}
```

K získání elementu title nám v SimpleXML stačí skutečně minimální kód. Mimo klasické metody projít všechny nody nabízí SimpleXML i XPath rozhraní. Stejně jako XML a HTML dokumenty v DOM lze i SimpleXML dokumenty také měnit a ukládat.

SimpleXML též vychází z libxml2, takže lze velice snadno konvertovat SimpleXML objekty na DomDocument objekty a naopak. Můžeme si tak vybrat vždy nejvhodnější nástroje a to bez podstatného ovlivnění rychlosti. Konverze jsou velmi jednoduché:

```
$sxe = simplexml_import_dom($dom);
$dom = dom_import_simplexml($sxe);
```

3.3 PDF v PHP

Podpora formátu PDF je v PHP umožněna pomocí externích knihoven, kterých je velké množství. V manuálu PHP je vysvětlena knihovna PDFlib vytvořena Thomasem Merzem a ClibPDF firmy FastIO.

Mezi další knihovny podporující tvorbu PDF dokumentů v PHP jazyku patří například FPDF a ezPDF.

Knihovny pro práci s PDF v PHP a Internet:
PDFlib – http://www.pdflib.com/

ClibPDF - http://www.fastio.com/

FPDF - http://www.fpdf.org/

ezPDF - http://www.ros.co.nz/pdf/

Po stažení dané knihovny z internetových stránek se samozřejmě dané knihovna musí nainstalovat a zkompilovat PHP. Postup instalace je v manuálu PHP nebo na stránkách tvůrců dané knihovny. Po úspěšné instalaci se mohou vytvářet PDF dokumenty.

3.4 Knihovna PDFlib

Jedna z nejrozšířenějších knihoven, podporována i v manuálu PHP je PDFlib. Knihovna obsahuje velké množství příkazů práce s poznámkami, texty, grafikou pro vložení do PDF dokumentu.

Příklad tvorby PDF dokumentu:

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "vysledky.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, "Praha", 50, 50);
pdf_show_xy($pdf, "23 stupnu", 80, 50);
pdf_show_xy($pdf, "Brno", 50, 80);
```

```
pdf_show_xy($pdf, "25 stupnu", 80, 80);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
?>
```

Tento skript vytvoří soubor s názvem vysledky.pdf, ve kterém bude napsáno písmem typu Times New Roman: „Praha 23 stupnu“, na druhém řádku: „Brno 25 stupnu“.

4 Analýza a návrh nové metody

Ve formátu XML se uchovávají v dnešní době veškeré naměřené informace, s kterými je dále zamýšleno zpracovávat je. Výhodou XML je také jejich nenáročnost a nezávislost.

Například měřicí soustavy jsou dnes navržené s webovým serverem, který má podporu pouze HTML, což předurčuje k tomu, aby výstupy byly ve formátu XML. Tento formát nepotřebuje žádný skriptovací jazyk jako PHP, je to prostě jen textový soubor se svou vlastní strukturou, která je dána určitými pravidly (viz Kapitola 2).

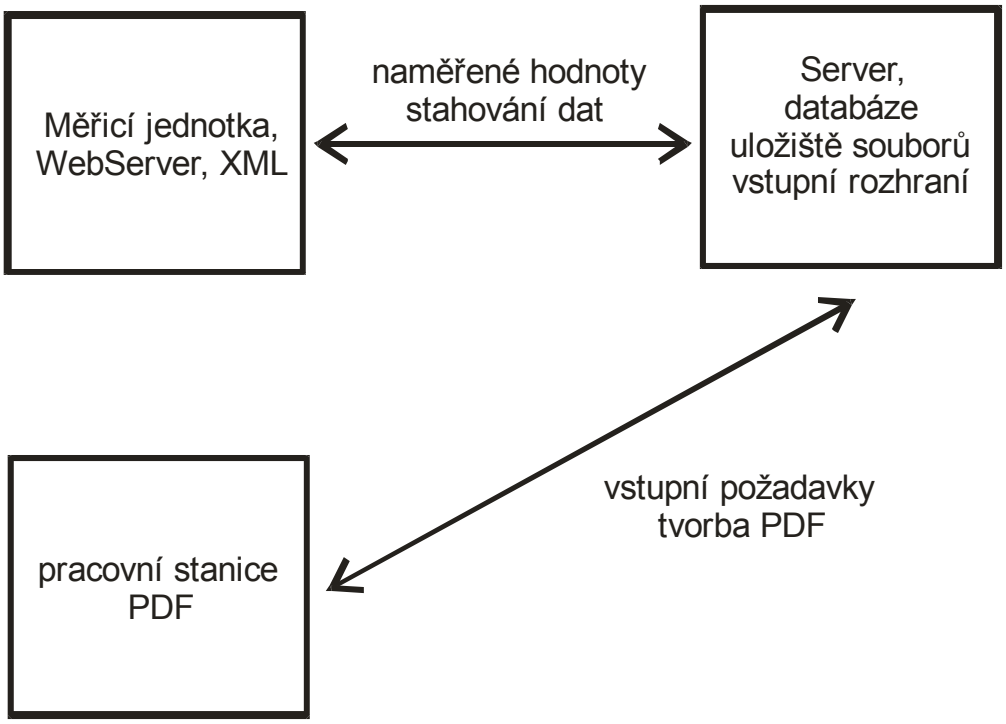
Existují soustavy, které měří nebo regulují určitý laboratorní model, své výsledky ukládají do XML dokumentu. Nová metoda umožňuje periodicky uchovávat výstupy těchto měření a vytvářet z nich čitelné výsledky pro další zpracovávání.

Cílem metody je XML výstupy ukládat a po zadání vstupních podmínek je automaticky převést do PDF formátu jako např. přehled naměřených hodnot, technickou zprávu.

Tyto dokumenty se mohou dále zpracovávat, uchovávat, odesílat, zveřejňovat, protože již jsou pro člověka snadno čitelné. V Příloze 1 je příklad výstupu PDF dokumentu z jednotky, která měří teplotu a výstup byl získán metodou popsanou níže. Samozřejmě každá soustava může mít vlastní XML strukturu, což znamená, že její výstup je odlišný. To vše závisí na tom, co má daná soustava uchovávat, měřit nebo regulovat. Každý PDF dokument může vypadat také úplně jinak, někdy bude jako technická zpráva, někdy jako přehled výsledků, někdy jako graf a tak dále.

V této kapitole je vysvětlen postup nové metody, jak číst XML dokumenty, jak je ukládat a jak vytvářet PDF výstupy.

Na obrázku 2 je znázorněno schéma, jak tato metoda funguje. Z měřicí jednotky s XML výstupem se údaje periodicky stahují do serveru, který je uchovává jako XML soubory nebo je ukládá do databáze. Na serveru je vstupní rozhraní. Uživatel se k němu přes webový prohlížeč připojí, zadá vstupní parametry. Po jejich odeslání se vytvoří požadovaný PDF dokument s danými údaji.



obr. 2 – schéma navržené metody

4.1 Čtení dat ze vzdáleného měřicího serveru

V měřícím server se data zobrazují v XML struktuře. Pro uchovávání těchto hodnot se z jiného serveru periodicky spouští skript, který danou XML strukturu uloží do souborů, případně jsou data pravidelně ukládána do databáze. Pro linuxové servery k pravidelnému spouštění úloh slouží cron, pro definici plánovaných činností příkaz crontab popsaný v Kapitole 5.1.2. V crontabu se nadefinuje spouštění skriptu, který se připojí na měřicí server a uloží aktuální XML strukturu např. do definovaného XML souboru. Skript je uveden v Příloze 2.

4.2 Čtení XML souboru

Ve verzi PHP 5, jak už bylo zmíněno v Kapitole 3.2, je nová nativní knihovna SimpleXML, ta umožňuje jednoduše získat údaje z XML souboru.

Například existuje soubor udaje.xml, který zobrazuje čas měření a hodnoty osmi teplotních čidel, z nichž funguje jen teplotní čidlo „te0“:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<data>
  <time>
```

```
    <h>15</h>
    <m>59</m>
    <s>59</s>
  </time>
  <te0>
    <id>000000E061B328</id>
    <val> 15.2</val>
  </te0>
  <te1>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te1>
  <te2>
     <id>FFFFFFFFFFFFFF</id>
     <val>  0.0</val>
  </te2>
  <te3>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te3>
  <te4>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te4>
  <te5>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te5>
  <te6>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te6>
  <te7>
    <id>FFFFFFFFFFFFFF</id>
    <val>  0.0</val>
  </te7>
</data>
```

Pro získání údaje o teplotě 15.2, který se nachází v této struktuře:

```
<te0>
  <id>000000E061B328</id>
  <val> 15.2</val>
</te0>
```

je PHP skript navržen následovně:

```
<?php
$xml = simplexml_load_file("udaje.xml");
$udaj = $xml->te0->val;
?>
```

Do proměnné „udaj“ byla vložena hodnota 15.2.
Počáteční základní element „data“ se neuvádí, jen druhý element „te0“ a v něm vnořený element „val“, který obsahuje danou hodnotu.

V případě, že elementy mají stejné pojmenování, pak se k ním přistupuje přes stejnou strukturu jakou má pole. Např. pro XML soubor se strukturou:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<data>
     <an0>
          <cnt>  15364</cnt>
          <val>      1</val>
     </an0>
     <an0>
          <cnt>  15363</cnt>
          <val>      1</val>
     </an0>
     <an0>
          <cnt>  15362</cnt>
          <val>      1</val>
     </an0>
     <an0>
```

```
            <cnt>  15361</cnt>
            <val>      1</val>
      </an0>
</data>
```

je PHP skript pro získání údaje 15363 navržen následovně:

```
<?php
$xml = simplexml_load_file("udaje.xml");
$udaj = $xml->an0[1]->cnt;
?>
```

V proměnné „udaj" se nachází 15363, pokud bychom chtěli jinou hodnotu, pak stačí pozměnit cestu:
$udaj = $xml->an0[3]->cnt;
Do této proměnné teď bude vložena hodnota 15361.
Hodnota „an0" je polem, začíná pozicí 0, v tomto konkrétním případě má jen 4 hodnoty a končí pozici 3.

4.3 Zpracování souborů

V experimentální úloze se teplota z webového serveru měřicí jednotky stahuje šestkrát denně vždy v 0:00, 4:00, 8:00, 12:00, 16:00, 20:00 do souboru.

Soubory jsou pojmenovány podle datumu uložení ve formátu yyyymmddhhmm.xml, tzn. soubor uložený v 11. 10. 2006 ve 12:00 se jmenuje 200610111200.xml

U tohoto systému jmen souborů je řazení bezproblémové. Soubory se automaticky řadí podle jména od nejstaršího po nejnovější.

V poli „rok" se uchovávájí první čtyři znaky z jména souboru.
V poli „mesic" se uchovávájí další dva znaky z jména souboru, čili znaky na pozici 5, 6.
V poli „den" se uchovávájí znaky na pozici 7 a 8.
V poli hodina se uchovávají znaky na pozici 9 a 10
V poli minuta se uchovávají znaky na pozici 11 a12.

Funkcí SimpleXML (viz Kapitola 4.2) se zároveň čte údaj uvnitř XML souboru a ukládá do pole s názvem „hodnota“

První hodnota v těchto polích nese údaje o prvním souboru, čili pole je řazeno vzestupně.

4.4 Tvorba grafu

Pro práci s obrázky v PHP je potřeba mít nainstalovanou GD knihovnu, která se nalézá na adrese http://www.boutell.com/gd/

Po úspěšné instalaci do PHP již lze používat grafické funkce, vytvářet libovolné obrázky, v tomto případě se funkce využijí na tvorbu grafu.

Vzorkování grafu

Při vytváření grafu se musí mít na paměti, že graf je omezen svou velikostí, v našem případě i šířky stránky v PDF.

To znamená, že se do něho nemůže nanést kompletní naměřené hodnoty třeba za posledních deset let. Je mnoho postupů, jak určit, které údaje se budou do omezeného prostoru grafu nanášet a které jsou pro daný graf nepodstatné. To vše závisí na druhu měření, periodě měření a co chceme v grafu zviditelnit.

V praktické úloze se zjišťuje teplota každé 4 hodiny, to je šestkrát denně. Takže jsem volil následující postup.

V grafu se nanáší hodnoty spolu s popisem časového údaje, pokud je mezera na ose x mezi jednotlivými hodnotami větší než 12 bodů. Je-li menší mezera, pak časový údaj se už na osu x nevleze, proto se vypouští, a zůstává tam jen den. Ale hodnoty jsou vyneseny do grafu všechny.

Fáze vzorkování grafu nastává, pokud je mezera mezi hodnotami v grafu menší než 9 bodů.

Pak se do grafu nanáší jen jedna hodnota daného dne. Standardně je to medián (prostřední hodnota) a ignoruje se hodnota naměřená v 0:00. Při standardním měření šestkrát denně (0:00, 4:00, 8:00, 12:00, 16:00, 20:00), se v grafu zobrazuje hodnota naměřená ve 12:00. Pokud je hodnot sudý počet, nepočítá-li se hodnota v 0:00, pak je vynášena do grafu hodnota na pozici nejbližší středu zleva. Tím je zaručeno, že

v případě výpadku nějakého měření bude graf souvislý. Totéž platí, když vypadne celý den nebo část období.

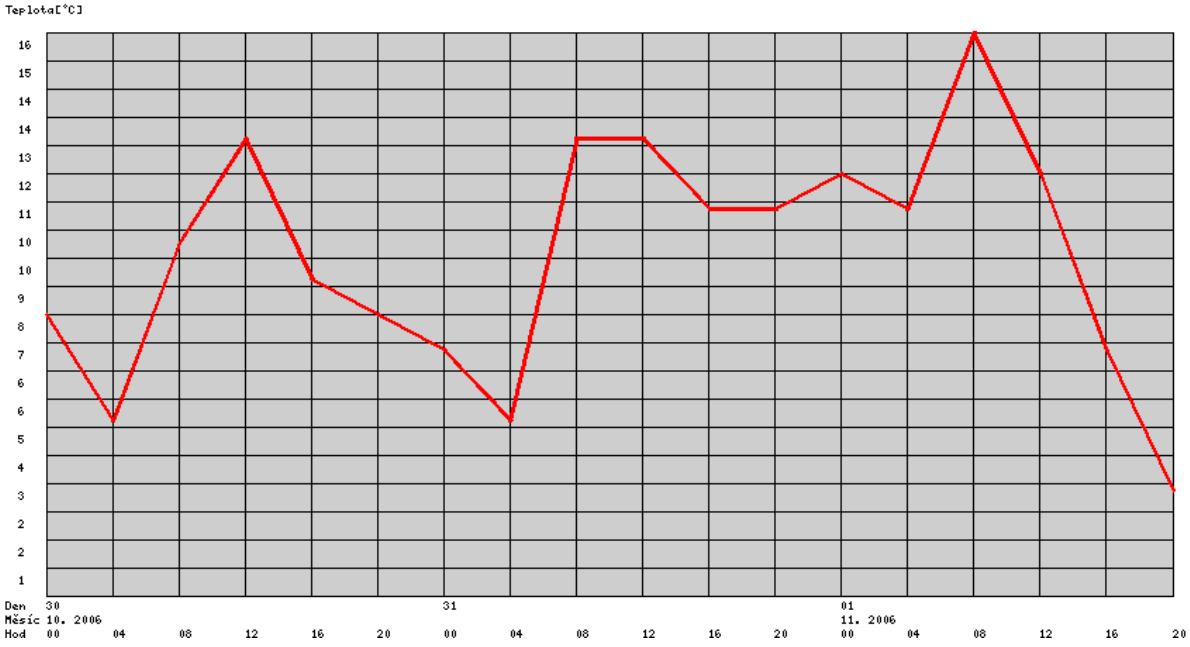
Podmínky pro převzorkování grafu jsou dále řešené podle počtu souborů vstupního výběru, tzn. počtu hodnot v grafu. Pokud je počet souborů do 720, pak se nanáší každý den, pokud je hodnot více než 720, nanáší se do grafu každý druhý den a poté podmínka funguje podle násobků, tzn. jestliže je počet souborů větší než 1440 nanáší se každý třetí den a tak dále.

Popisy u osy y a pomocné vodorovné čáry se nanáší podle zjištění maxima a minima, takže mezera mezi vodorovnými čáry se počítá podle vzorce:

$$mezera\_y = \frac{výška\_grafu}{\max - \min}$$

Vytvořený graf se do PDF vkládá jako obrázek (obr. 3). Nejvýhodnější formát grafu je v tomto případě PNG.

Nastavení grafu a jeho chování se nachází v souboru settings.php (Příloha 5)



obr. 3 – ukázka grafu

4.5 Základní funkce knihovny FPDF

FPDF je již vytvořená PHP třída, která dovoluje generovat PDF. Výhoda použití knihovny fpdf.php spočívá v tom, že nepotřebujete zakompilovanou podporu pro PDF ve vaší verzi PHP (např. PDFlib knihovnu). F z FPDF znamená „free“ (zdarma). Tuto PHP třídu lze volně šířit, využívat a modifikovat.

Zde jsou uvedené základní funkce:

FPDF([string orientation [, string unit [, mixed format]]])

Vytvoří nový objekt a nastaví základní vlastnosti.

Parametry:

nepovinný "orientation" - určuje zde bude stránka na výšku "P" nebo na šířku "L".

nepovinný "unit" - nastavý velikostní jednotku - "pt", "mm", "cm", nebo "in".

nepovinný "format" - určí formát stránky - "A3", "A4", "A5", "Letter", "Legal".

Př. : $pdf=new FPDF("P","mm","A4");

AddPage([string orientation])

Slouží k vytvoření nové stránky.

Parametry:

nepovinný "orientation" - určuje zde bude stránka na výšku "P" nebo na šířku "L"

př. : $pdf->AddPage("L");

SetFont(string family [, string style [, float size]])

Slouží k volbě stylu a velikosti písma.

Parametry: povinný "family" - určuje typ fontu (př.: Courier, Helvetica, Arial,...)

nepovinný "style" - určuje styl písma - tlusté "B", kurzíva "I" a podtržené "U". Je možné i kompinovat - př. "BI" = tučné + kurzíva.

nepovinný "size" - určuje velikost v bodech.

př: $pdf->SetFont("Arial","BU",14) - Arial, tučné a podtržené, 14 bodů.

Output([string file [, boolean download]])

Odešle vygenerované PDF prohlížeči.

Parametry:

nepovinný "file" - určí jméno souboru. Pouze pokud chceme soubor uložit na srveru

nebo odeslat klientovy ke stažení. V tomto případě nedojde k načtení plug-inu a zobrazení.
nepovinný "download" určuje zda bude PDF dokument uložen na serveru(false) nebo u klienta(true).
Př. : $pdf->Output("test.pdf",true) - odešle klientovy soubor test.pdf.
Př.2 : $pdf->Output("test.pdf") - uloží na serveru soubor test.pdf.
Př.3 : $pdf->Output() - prohlížeč načte plug-in a zobrazí obsah dokumentu PDF.

Cell(float w [, float h [, string txt [, mixed border [, int ln [, string align [, int fill [, mixed link]]]]]])
Slouží k vytvoření textového objektu.
Parametry:
povinný "w" - šířka textového objektu.
nepovinný "h" - výška textového objektu.
nepovinný "txt" - text, který bude vložen do textového objektu.
nepovinný "border" - určuje zda bude viditelný okraj. 0 = bez okraje, 1 = s okrajem. Je možno použít i "L" = levý, "T" = horní, "R" = pravý a "B" = dolní + kombinace.
nepovinný "ln" - Určuje kam se bude vkládat další objekt. 0 = v pravo 1 = na začátek následující řádky, 2 = pod tento objekt.
nepovinný "align" - nastavuje zarovníní textu. "L" v levo, "C" na střed, "R" v pravo.
nepovinný "fill" - určuje zda použít (1) nebo nepoužít (0) barvu pozadí.
nepovinný "link" - URL nebo identifikátor vrácený metodou AddLink(). K této funkci se vrátíme v dalších dílech tohoto článku.
Př. : $pdf->Cell(50,10,"TEST01","LBR",1,"c",0) - Vytvoří textový objekt 50x10 s textem TEST01, který bude z leva, z prava a ze spodu orámován a který bude zarovnán na střed, bez barvy na pozadí. Další objekt bude vložen na další řádku.
Př.2 : $pdf->Cell(100) - vytvoří tvrdou mezeru o dané šířce.

Output([string file [, boolean download]])
Odešle vygenerované PDF prohlížeči.
Parametry:
nepovinný "file" - určí jméno souboru. Pouze pokud chceme soubor uložit na srveru nebo odeslat klientovy ke stažení. V tomto případě nedojde k načtení plug-inu a zobrazení.

nepovinný "download" určuje zda bude PDF dokument uložen na serveru(false) nebo u klienta(true).
Př. : $pdf->Output("test.pdf",true) - odešle klientovy soubor test.pdf.
Př.2 : $pdf->Output("test.pdf") - uloží na serveru soubor test.pdf.
Př.3 : $pdf->Output() - prohlížeč načte plug-in a zobrazí obsah dokumentu PDF.

MultiCell(float w, float h, string txt [, mixed border [, string align [, int fill]]])
Parametry:
povinný "w" - určuje šířku objektu MultiCell. Pokud zadáte 0 bude objekt roztažen na celou šířku dokumentu.
povinný "h" - určuje výšku jednoho řádku. Čím větší čislo, tím větší mezera bude mezi řádky. POZOR: 0 způsobí NULOVÉ odsazení řádků - všechny řádky textu budou vygenerovány do JEDNÉ řádky.
povinný "text" - Text, který má být vypsán. Může obsahovat znak "\n" pro odřádkování.
nepovinný "border" - okraj objektu. Viz Cell()
nepovinný "align" - zarovnání textu v textovém objektu. "L","R","C","J" - viz Cell()
nepovinný "fill" - Určuje zda použít barvu pozadí. Viz Cell()
Př.: $pdf->MultiCell(0,13,$data["popis"],0,"J",0);

Image(string file, float x, float y, float w [, float h [, string type [, mixed link]]])
Parametry:
povinný "file" - Jméno souboru s obrázkem. Jsou podporovány formáty JPEG a PNG. Není podporován INTERLACED režim obrázků.
povinný "x" a "y" - X-ová a Y-ová souřadnice levého horního rohu obrázku.
povinný "w" a nepovinný "h" - Šířka a výška obrázku. Pokud není výška zadána je automaticky dopočítána tak, aby byl zachován poměr stran obrázku.
nepovinný "type" - Určuje typ obrázku - "JPG", "JPEG", "PNG". Pokud není zadán, je určen automaticky (z přípony souboru).
nepovinný "link" - URL nebo identifikátor vrácený metodou AddLink(). K této funkci se vrátíme v dalších dílech tohoto článku.

Přehled dalších funkcí:
AddFont – přidat nový font
AddLink – vytvořit odkaz

AddPage – přidat novou stránku

Cell – zobrazit buňku

Close – zavřít dokument

Error – fatální chyba

Footer - zápatí

GetStringWidth – zjistit délku řetězce

GetX – získat aktuální pozici na X

GetY - získat aktuální pozici na Y

Header - záhlaví

Image – vložit obrázek

Line – nakreslit čáru

Link – vložit odkaz

Ln – nový řádek

MultiCell – vytvořit text v buňce

Output – uložit nebo odeslat dokument

PageNo – číslo stránky

Rect – nakreslit obdelník

SetAuthor – vložit autora dokumentu

SetCompression – zapnout kompresi

SetCreator – vložit tvůrce

SetDisplayMode – nastavit mod obrazovky

SetDrawColor – nastavit barvu

SetFillColor – nastavit barvu výplně

SetFont – nastavit font

SetFontSize – nastavit velikost fontu

SetLeftMargin – nastavit odsazení zleva

SetLineWidth – nastavit tloušťku čáry

SetLink – definuje cíl odkazu na pozici a stránce

SetMargins – nastavení okrajů

SetRightMargin – nastavení pravého okraje

SetSubject – nastavení předmětu dokumentu

SetTextColor – nastavení barvy textu

SetTitle – nastavení nadpisu

SetTopMargin – nastavení horního okraje

SetX – nastavení X aktuální pozici

SetXY - nastavení X a Y aktuální pozice

SetY – nastavení Y aktuální pozici

Text – vložení textu

4.6 Vložení obrázku do PDF

Pro převod obrázku do PDF se používají výše popsané funkce FPDF následujícím způsobem:

```
require('fpdf.php');
$pdf=new FPDF("P","pt","A4");
$pdf->Open();
$pdf->AddPage();

$img="image.png";
$info=getimagesize($img);
$y=30;
$x=30;
$pdf->Image($img,$x,$y,$info[0]-10,$info[1]-10);
$pdf->Output();
```

První řádek skriptu zjednodušeně označuje, že je využívána knihovna fpdf.php.

Vytvoří se stránka s orientací na výšku, základní jednotka se použije bod a velikost stránky bude A4.

Do proměnné „img“ se uloží název souboru (obrázku) „image.png“. Do proměnné „info“ se uloží rozměry obrázku.

Tento obrázek se v PDF dokumentu umístí tak, že jeho levý horní roh bude od levého horního rohu stránky umístěn ve vzdálenosti 30 bodů na ose X a 30 bodů na ose Y.

Pravý dolní roh obrázku bude umístěn, tak aby šířka a výška obrázku byla o 10 bodů menší než vkládaný originál. Výstupem je načtení plug-inu v prohlížeči a zobrazení obsah dokumentu PDF.

4.7 Čeština v PDF

Knihovna FPDF využívá své interní fonty. Pokud tyto fonty nestačí je možné použít vlastní fonty typu TTF nebo Type1. Před jejich použitím je ale nutné převést je do podporovaného formátu. PDF ukládá fonty přímo jako součást dokumentu a používá kompresi GZIP. FPDF ještě potřebuje speciální definiční soubor pro každé písmo. Obecně stačí pouze vytvořit definiční soubor a písmo TTF nebo Type1 zkomprimovat. K tomu slouží funkce MakeFont(). Celý postup, ale zas tak jednoduchý není.

TTF fonty obsahují sadu znaků pro každé kódování, proto je vhodné odstranit ze souboru TTF znaky, které není potřeba v tomto případě stačí jen kódování: Windows-1250.

Pro generování definičního souboru je potřeba speciální soubor *.afm, který obsahuje informace o jednotlivých znacích. K vytvoření tohoto souboru se použije program ke konverzi TTF na Type1.

Na příkazovou řádku se napíše:

```
ttf2pt1 -b -L cp1250.map arial.ttf arial
```

"arial.ttf" je název TTF fontu.

"arial" je název vygenerovaného Type1 fontu.

"cp1250.map" je mapa znaků pro kódování windows-1250.

Poté stačí vygenerovat definiční soubor a příslušný archív s daty:

```
<?php
include "makefont.php";
$font="arial";
MakeFont("$font.pfb", "$font.afm" , "cp1250");
?>
```

Tento PHP kód vygeneruje soubor "arial.php" a "arial.z". To jsou soubory, které jsou potřeba, ostatní odstraníme. Tyto dva soubory zkopírujeme do adresáře fpdf_font.

Vygenerovaný font Arial s českými znaky se použije takto:

```
<?php
define('FPDF_FONTPATH','/fpdf_font/');
require('fpdf.php');
$pdf=new FPDF("P","mm","A4");
$pdf->Open();
$pdf->AddFont('arial','','arial.php');
$pdf->AddPage();
$pdf->SetFont('arial','',18);
$pdf->Cell(0,15,"zkouška háčků a čárek","0",1,"C");
pdf->Output();
?>
```

Na prvním řádku se nastaví cesta k definicím fontů - "/fpdf_font/". V tomto adresáři budou umístěny soubory "arial.php" a "arial.z".
Pomocí $pdf->AddFont() zaregistrujeme nový font.

AddFont(string family [, string style [, string file]])
povinný "family" - název fontu. nepovinný "style" - styl funtu: "B" - tlustý, "U" - podtržený, "I" - kurzíva, "" - normální. nepovinný "file" - určuje název souboru s písmem.
Př.:
$pdf->AddFont('arial','','arial.php');
$pdf->AddFont('arial','B','arialbold.php');
$pdf->AddFont('arial','I','arialitalic.php');

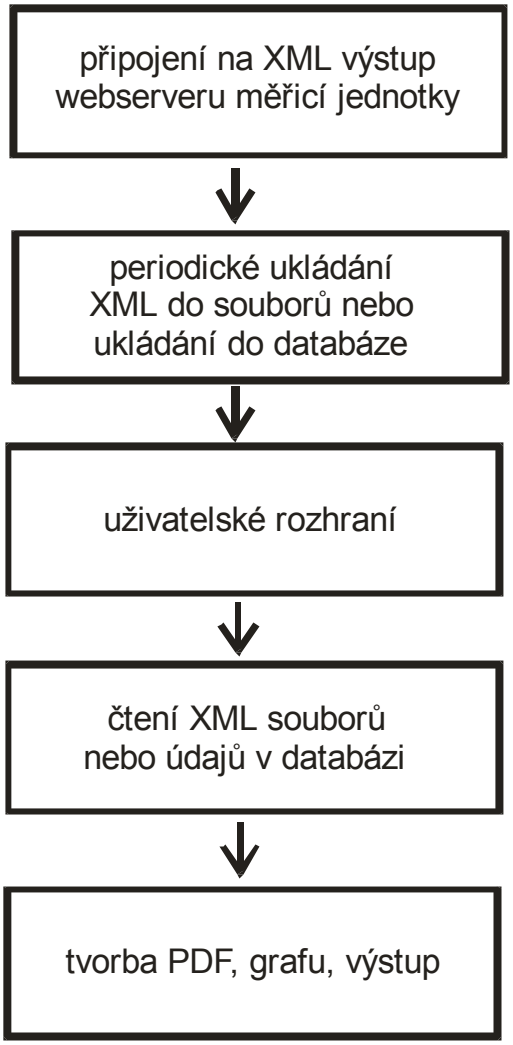
Tato sada zaregistruje písmo "arial" ve třech variantách "normální, tlusté a kurzíva".

Pro každý styl písma musí být samostatný definiční soubor a samostatný archív s daty. Ve Windows je také pro každý styl samostatný soubor TTF. Každý další styl zabere další místo a dokument PDF roste a roste.

Výše popsaný nástroj na tvorbu písma s českými znaky najdete na přiloženém CD (obsah v Příloze 7).

4.8 Shrnutí metody

Výše popsané body návrhu nové metody shrnuje schéma na obrázku 4. Server se periodicky připojuje k měřící jednotce, kde čte XML výstup, který ukládá na svém disku do XML souborů, případně databáze. Uživatel se přes webový prohlížeč dostane na rozhraní, kde zadá potřebné vstupní parametry, podle kterých se v další fázi načtou uložené XML soubory (případně údaje z databáze) a vytvoří se programátorem definovaný výstup v PDF dokumentu.



obr. 4 – schéma navržené metody

5 Řešení praktické úlohy

Typický regulačním problémem je měření regulovaných veličin. Měřicí a řídicí jednotka s vlastním webovým serverem umístěna na Ústavu informatiky Akademie věd (viz obr. 5, 6), na kterou jsou napojeny mimojiné snímače fyzikálních veličin, zpracovává údaje z těchto senzorů a zobrazuje je v XML dokumentu.

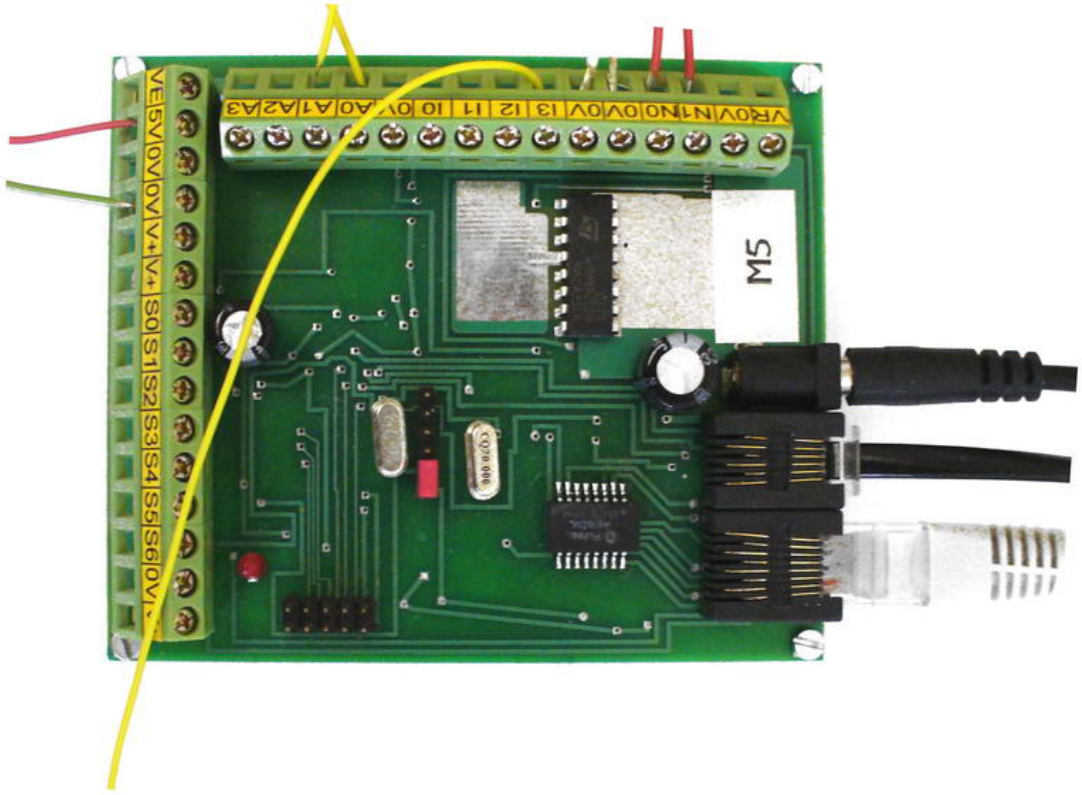
Cílem je naměřené teplotní hodnoty, jejichž výstupem je XML dokument periodicky číst a uchovávat. Uložena data se podle vstupních podmínek vytvořeného webového rozhraní dále zpracují do tiskové podoby ve formátu PDF spolu s teplotním grafem.

Cílem této kapitoly je ilustrovat navrženou metody na praktickém příkladu.

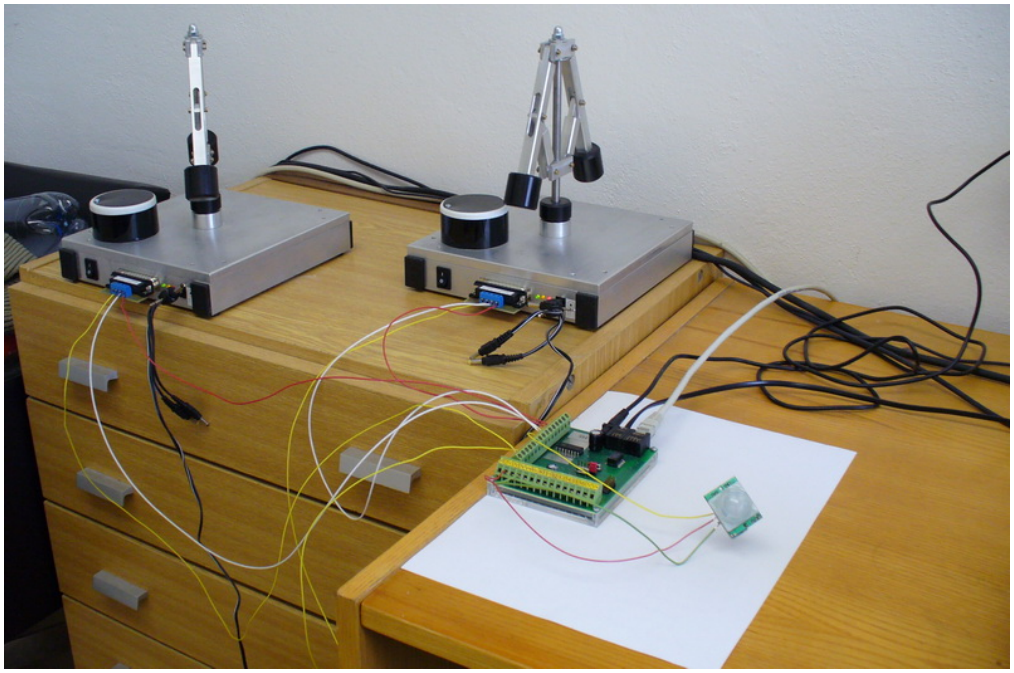
5.1 Měření teploty

Webový server na obrázku 5 obsahuje senzory teploty, pohybu… Údaje z těchto senzorů se ukládají do XML dokumentu, který je volně přístupný na internetu.

Úkolem této praktické úlohy je aplikace výše popsané teorie, metod, poznatků a vytvoření webové aplikace, která převede hodnoty měření teploty do PDF.



obr. 5 - webový server



obr. 6 – zapojený webový server

5.1.1 Ukládání XML souborů

XML dokument s naměřenou hodnotou teploty je umístěn na adrese: http://ctrlv4.cs.cas.cz/temper.xml a je neustále aktualizován.

Pro získání historie naměřených hodnot se tento XML dokument musí pravidelně ukládat do vybraného úložiště. V této praktické úloze byl zvolen interval každé čtyři hodiny, čili soubor se stahuje z „ctrlv4.cs.cas.cz“ a ukládá na jiný webový server v 0:00, 4:00, 8:00, 12:00, 16:00, 20:00 denně. Obsah souboru je stejné XML struktury, jen název souboru nese časový údaj jeho uložení ve formátu [yyyymmddhhmm].xml, např. soubor s názvem 200611100400.xml, byl stažen a uložen 10. 11. 2006 ve 4:00.

Důvody a výhody tohoto ukládání popsané v Kapitole 4.3.

K výše uvedenému kroku slouží popsaný skript v Příloze 2, který je spouštěn pomocí příkazu crontab:

0 0,4,8,12,16,20 * * * php4 /home/people/mrazekk/public_html/skriptteplota.php

5.1.2 Crontab – pravidelné spouštění skriptů

Na linuxových serverech existuje služba cron, která se ovládá příkazem crontab. Jeho použití je následovné:

crontab [volby] [soubor]

Zobrazí, nastaví nebo odstraní váš crontab soubor (tzv. "cron tabulka"). Superuživatel může pomocí volby –u uživatel spustit crontab pro jiného uživatele. Crontab soubor je seznam příkazů, které budou automaticky spuštěny v daný okamžik, přičemž každý příkaz je zapsan na jednom řádku.

Před každým příkazem je uvedena řada čísel, která určuje čas spuštění. Tato řada obsahuje pět skupin čísel s následujícím významem:

Minuta	0-59
Hodina	0-23
Den v měsíci	1-31
Měsíc	1-12
	Jan, Feb, Mar, …
Den v týdnu	0-6, přičemž 0 značí Neděli
	Sun, Mon, Tue, …

Pro rozdělení hodnot v rámci jedné skupiny je možné použít čárku nebo pomlčku, která značí interval hodnot, nebo znak hvězdička značící libovolnou hodnotu. Například uvažte následující položky crontab souboru:

59 3 * * 5	find / -print | zalohovaci program
0 0 1,15 * *	echo "Timesheets due" | mail uzivatel

První příkaz provede zálohu systému souborů vždy v pátek ráno ve 3:59. Druhý příkaz zašle upozornění vždy 1. a 15. v měsíci

Volby:

-e	Spustí editaci aktuálního crontab souboru současného uživatele. (Případně vytvoří nový soubor.)
-l	Vypíše uživatelův crontab soubor na standardní výstup.
-r	vymaže uživatelův crontab soubor z příslušného adresáře.
-u uživatel	určuje, který uživatel bude brán v potaz.

5.1.3 Rozhraní

K zadání vstupních parametrů od uživatele slouží skript index.php, který je popsán v Příloze 3.

Pro uživatele se na dané stránce zobrazuje datum prvního měření a datum posledního měření, tyto údaje jsou informativní a slouží, aby se vědělo, jaké rozmezí do políček „od“ a „do“ je možné napsat, resp. od kdy do kdy jsou údaje na serveru uchovávané.

Rozhraní bylo vytvořeno user-friendly. Uživatel zadává datum v českém, standardním formátu, čili např. 22. 10. 2006, vstupní parametry jsou ošetřene, kritériem znaku tečky „.“. Dále je akceptováno, když den má jednu číslici nebo dvě číslice, nemusí se psát 05.01.2006, ale klidně 5. 1. 2006. Mezery jsou také odstraněny, pokud autor napiše např. 1. 10. 2006, je to totéž jako by napsal 1.10.2006.

Další sympatickou vlastností je, když uživatel zadá jen 22.10. a nezadá rok, automaticky skript zpracuje data v aktuálním roce, čili 22.10.2006. totéž platí, když zadá např. od „20.“ do „25.“, skript automaticky zpracuje údaje v aktuálním měsíci a aktuálním roce, tzn. pokud to zadává uživatel v listopadu 2006, získá tak údaje od 20. 11. 2006 do 25. 11. 2006.

Důležité je nezapomenout ukončit číslo dne, měsíce tečkou, tak jak je to standardně v českém jazyce s psaním dnů a měsíců, jsou to řadové číslovky, které vždy končí tečkou, čili nic nezvyklého.

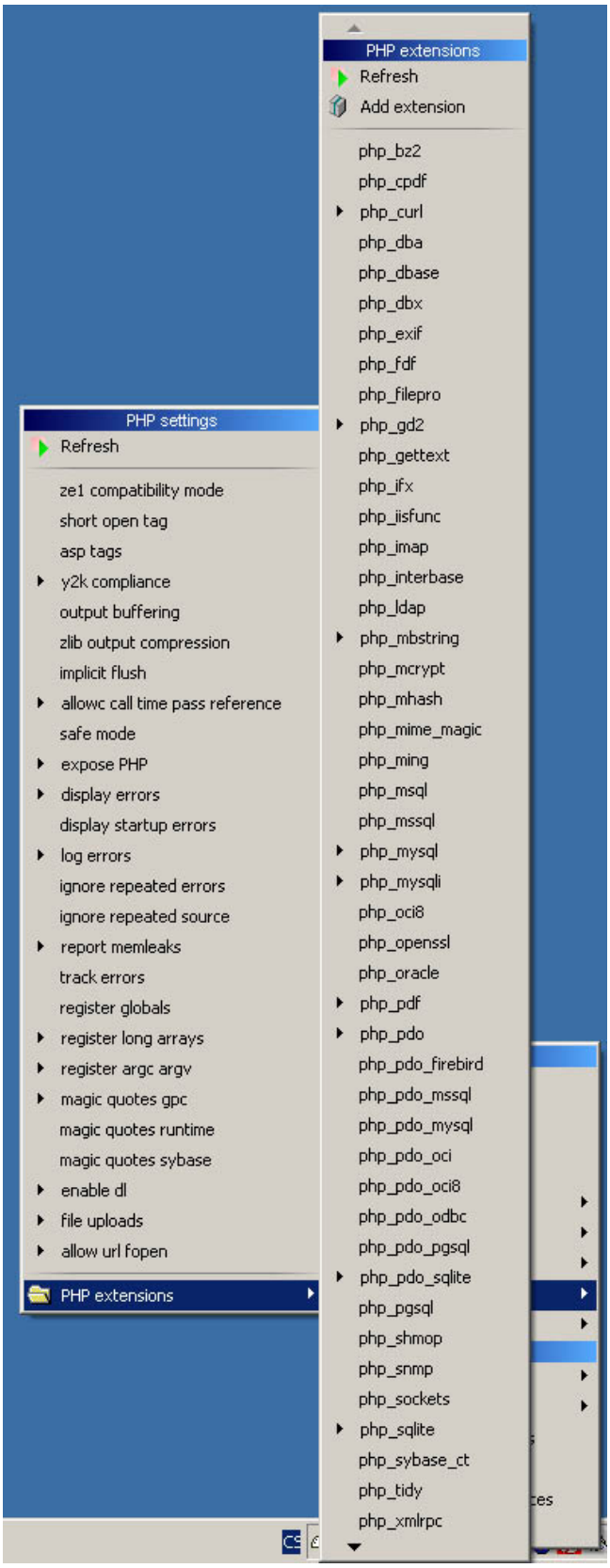
Skript index.php tyto údaje jen odesílá dalšímu skriptu pdf.php, který pak tyto vstupní údaje zpracuje a zároveň udělá patřičné pdf s grafem.

Vytvoření grafu je popsáno v Kapitole 4.4, čtení XML souboru v Kapitole 4.2, skript pdf.php, který toto vše provádí a vytvoří výsledné PDF s grafem a hodnotami je popsán v Příloze 5 spolu se skriptem settings.php, kde se vše uživatelsky nadefinuje. Skript pdftable.php, který je rozšířením knihovny FPDF pro vytváření tabulek v PDF je popsán v Příloze 4

5.1.4 Nastavení serveru

K praktické úloze se využívá volně dostupný WampServer verze 1.6.6, který obsahuje konfigurovatelný server s PHP 5.2.0, Apache 2.0.59, MySQL 5.0.27. Ostatní knihovny (jako GD) se zinicializují pouhým kliknutím myši v uživatelském rozhraní tohoto serveru (obr. 7)

Instalační soubor WampServeru stejně jako všechny skripty výše popsané se nachází na přiloženém CD, obsah CD je uveden v Příloze 7.

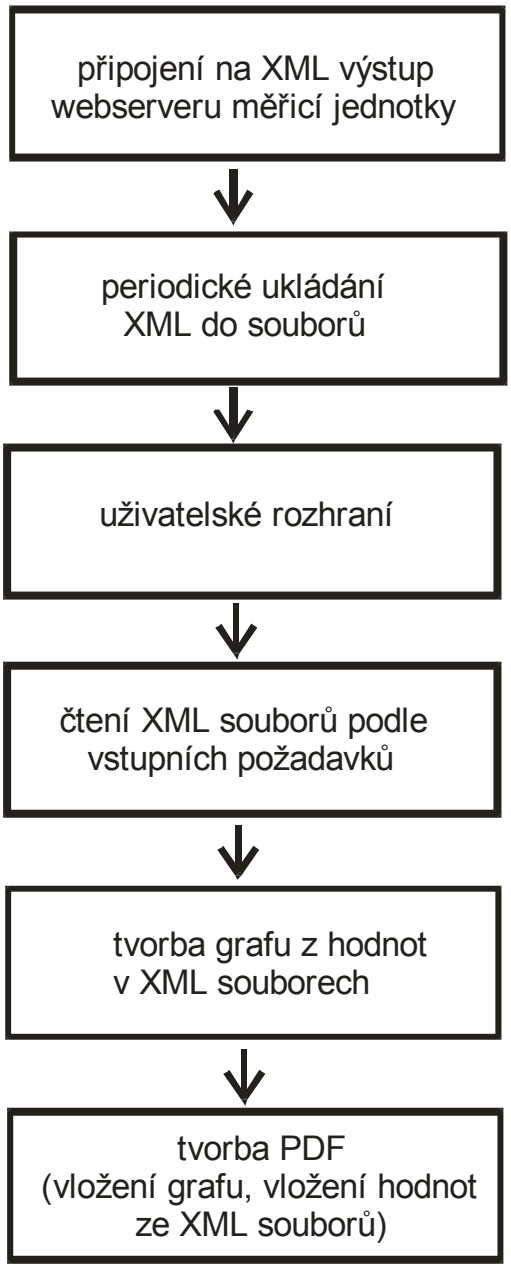


obr. 7 – konfigurování WampServeru

5.2 Dosažený výsledek

Výsledkem aplikace postupu automatického převodu XML na PDF, jehož návrh řešení a jednotlivé kroky jsou uvedeny v Kapitole 4, na této praktické úloze je dokument PDF, který naleznete v Příloze 1.

Na obrázku 8 je zobrazeno schéma postupu navržené metody v této praktické úloze.



obr. 8 - schéma postupu navržené metody praktické úlohy

6 Převod MathML do PDF

Během převodu je potřeba řešit i případ, kdy měřicí server uchovává z nějakého důvodu i matematické vzorce, např. pro ověření a přehled jakou operaci daná soustava provedla. K uchovávání vzorců slouží MathML, který je specifickým a striktně daným XML dokumentem.

Cílem této úlohy je převod modelem vyprodukovaného XML dokumentu se strukturou MathML do uživatelsky čitelného PDF pro další zpracování a tisk. V praxi to může vypadat tak, že stroj provede nějakou matematickou operaci, výstupem bude i její matematický vzorec, který se převede a uloží do PDF dokumentu a obsluha stroje si tuto operaci může kdykoliv později vyhledat, zkontrolovat, vytisknout, zveřejnit.

6.1 Historie MathML

Matematika vyjadřuje myšlenky pomocí vzorců, které se více podobají grafice než běžnému textu. S masivním rozšiřováním převážně textového webu však vyvstal problém jejich zápisu. Většinou se tento problém řeší bitmapovým obrázkem, což má ale mnoho nedostatků. Problémů, spojených se zápisem matematických vzorců prostřednictvím bitmapových obrázků, je celá řada:

- datová velikost není zrovna ideální
- složitá manipulace, tvorba grafickým editorem
- nemožnost interpretace dat, žádná aplikace z obrázku nevyčte jeho význam
- dynamická tvorba (závislá na datech uživatele) takových vzorců je nemožná - není možné měnit vzorec přímo v kódu

V době, kdy internet neexistoval, řešil se problém zápisu matematických vzorců. Nejen pro tento účel vznikl program TeX, který pomocí slov, čísel a operátorů vyjadřuje rovnice. Zápis se do značné míry podobá anglickému textovému zápisu. I přesto, že internet byl od počátku věnován vědcům, na matematické vzorce ve své roztržitosti nepamatovali. Proto se i nadále vzorce šířily prostřednictvím Texu.

Později se objevila možnost zapisovat vzorce do stránek pomocí Java-apletů WebEQ (Web EQuation). Vzorce jsou obsaženy ve zdrojovém kódu (jako hodnota atributu "value" elementu <param>) a jeho zápis vychází z TeXu. Dalším značkovacím jazykem je ISO 12083, vycházející z SGML. Jeho struktura je přísnější než struktura TeXu, nicméně obě technologie mají řadu společných vlastností.

Někdy kolem roku 1995 organizace W3C uznala, že nemožnost výměny matematických vzorců představuje vážný problém. Objevily se návrhy na rozšíření HTML (tehdy HTML 3.0) o nové elementy, umožňující formátování vzorců. Později se ujasnilo, že by měl vzniknout obecný mechanismus (XML) a zápis vzorců by měl být jednou z jeho podmnožin. Nakonec vznikl **MathML** - Mathematical Markup Language - matematický značkovací jazyk, podmnožina XML. Jde o značkovací jazyk podobný svou strukturou HTML.

MathML bylo navrženo s několika základními cíli:

1. možnost konvertovat existující dokumenty do MathML (nejčastěji TeX a ISO 12083)
2. možnost připojit MathML k HTML a interpretovat ho prohlížeči (toho bylo dosáhnuto díky odvození z XML)
3. možnost získání vzorce ze zdrojového zápisu, který může být určeným programem interpretován a vyhodnocen

Prezentace MathML je do značné míry založena na stylech. Díky nim je sice možné zobrazit MathML v prohlížeči, ale je třeba rozšířit schopnosti prohlížečů pomocí elementů pro zobrazení MathML. Sám jazyk MathML však prostřednictvím elementů a atributů popisuje druh obsahu (číslo, proměnná, operátor...) a, v podstatě, i požadovaný vzhled, protože většina matematických konstrukcí má určeno dané grafické spodobnění.

6.2 Základní popis MathML

MathML je založeno na XML. Struktura MathML je tvořena, stejně jako v případě XML, elementy, atributy a entitami pro zápis zvláštních znaků. MathML prvky lze rozdělit do několika kategorií:

- prezentační prvky popisují strukturu vzorce (horní index, dolní index)
- významové prvky přímo popisují matematický objekt (například "plus" nebo "vector")
- prvky rozhraní jsou ty, které slouží k zapojení MathML do dokumentu HTML, XML a dalších

Níže uvádím prezentační prvky MathML, které jsou použity v této praktické úloze:

<mi>.....uzavírá proměnné

<mn>.....uzavírá čísla

<mo>.....uzavírá operátory

<mfrac>.....element označující zlomek

<msqrt>.....druhá odmocnina

<msup> .....horní index

<msub> .....dolní index

<mrow>.....řádek tabulky (používá se k uzavření dolního, horního indexu)

6.2.1 Dokument MathML

Praktickým případem kódování v MathML je vzorec

$$\sqrt{\frac{2+\sqrt{x}}{5*g}} - \frac{\sqrt{2-x^{2-y}+5}}{\sqrt{3}} + x^2 + \frac{x}{2-x^3}$$

Jeho kód vypadá následujícím zpusobem:

```
<?xml version="1.0" encoding="UTF-8"?>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <msqrt>
    <mfrac>
      <mn>2<mo>+</mo>
      <msqrt>
        <mi>x</mi>
      </msqrt>
      </mn>
      <mn>5<mo>*</mo>
      <mi>g</mi>
      </mn>
    </mfrac>
  </msqrt>
  <mo>-</mo>
  <mfrac>
    <msqrt>
      <mn>2<mo>-</mo>
      <msup>
        <mi>x</mi>
        <mrow>
```

```
            <mn>2</mn>
            <mo>-</mo>
            <mi>y</mi>
          </mrow>
        </msup>
        <mo>+</mo>
        <mn>5</mn>
        </mn>
      </msqrt>
      <msqrt>
        <mn>3</mn>
      </msqrt>
    </mfrac>
    <mo>+</mo>
    <msup>
      <mn>x</mn>
      <mrow>
        2</mrow>
    </msup>
    <mo>+</mo>
    <mfrac>
      <mn>x</mn>
      <mn>2<mo>-</mo>
      <msup>
        <mi>x</mi>
        <mrow>
          <mn>3</mn>
        </mrow>
      </msup>
      </mn>
    </mfrac>
</math>
```

Na tomto vztahu bude později ilustrován převod MathML dokumentu do PDF.

6.2.2 Uložení MathML do databáze

Pro převod MathML syntaxe jsou uloženy všechny tágy, jejich hodnoty do databáze, aby se mohl zaručit korektní převod i vnořených tágu, musí se znát i jejich hloubka vnoření, s kterou se poté prioritně pracuje.

Takže databáze pojmenovaná XML má strukturu

sloupec	**typ**
id	int(5)
tag	text
value	text
level	int(5)

Sloupec „id“ slouží pro řazení a vkládá se se tam pořadí uložení tágu
Do sloupce „tag“ se zapisuje jméno tágu.
V sloupci „value“ se nachází hodnota daného tágu.
A „level“ je číslo vnoření daného elementu.
Zde příklad tabulky:

id	tag	value	level
1	msqrt		0
2	mfrac		1
3	mn	2	2
4	mo	+	3
5	msqrt		3
6	mi	x	4
7	mn	5	2
8	mo	*	3
9	mi	g	3
10	mo	-	0
11	mfrac		0
12	msqrt		1
13	mn	2	2
14	mo	-	3
15	msup		3
16	mi	x	4
17	mrow		4

18	mn	2	5
19	mo	-	5
20	mi	y	5
21	mo	+	3
22	mn	5	3
23	msqrt		1
24	mn	3	2
25	mo	+	0
26	msup		0
27	mn	x	1
28	mrow	2	1
29	mo	+	0
30	mfrac		0
31	mn	x	1
32	mn	2	1
33	mo	-	2
34	msup		2
35	mi	x	3
36	mrow		3
37	mn	3	4

Takto se uložil do tabulky XML dokument pomocí příkazů SimpleXML v PHP5, kde se musel použít rekurzní postup:

```
<?php
include "dbconnect.php";
mysql_query('TRUNCATE TABLE structure');

$sx = simplexml_load_file("m1g.xml");

$level = 0;

NLevel($sx,$level);

function NLevel($element,$level)
{
  static $line = 0;
```

```
  foreach ($element->children() as $tags => $child)
    {
       print ++$line.". ".$tags . "->$child ($level)<br>";

mysql_query('INSERT INTO structure SET id='.$line.',
tag="'.$tags.'", value="'.$child.'", level='.$level.'');
       NLevel($child,$level+1);
   }
}
?>
```

Tímto způsobem je možné ukládat libovolný XML dokument. Metoda je tudíž univerzální a hlavně uložená data se dále mohou libovolně zpracovávat a ukládat do PDF.

6.2.3 Analýza MathML značek a jejich převod na obrázek

V převodu MathML do PDF je hlavním problémem zobrazování matematických vzorců jako zlomky, mocniny, odmocniny, a ty se nedají zobrazit pouze jako textová forma.

Proto se musí převést do tvaru umožňující převod do obrázků a poté daný obrázek uložit do PDF dokumentu.

Na internetu existuje volně přístupný projekt PHPMathPublisher z roku 2005, jehož autorem je Francouz Pascal Brachet. Ten se zabýval převodem textových řetězců do grafických symbolů a vzorců. Má striktně zadáno jak se musí daný řetězec napsat, aby se změnil v obrázek.

Úkolem této úlohy tím pádem je převést uložené informace z XML dokumentu v databázi na strukturu řetězce, na který bude schopno použít jeho knihovnu.

Zdánlivě jednoduchý úkol se stal matematickým problémem (např. složitě vnořeny zlomek, do zlomku ve jmenovateli a čitateli, s odmocninou), ale nakonec skript, který převádí strukturu XML z databáze na čitelný formát je na světě. V úloze se uvažuje převod zlomků, odmocnin, mocnin, dolních indexů.

Analýza XML dokumentu je ve třech fázích

1. přečtení databázových hodnot odshora dolů, označkování počátečního a konečného tágu pomocí čísla vnoření. Označení nulových vnoření.

2. kontrolní analýza správné syntaxe, odstranění chyb první analýzy, převedení na konstrukci řetězce. Problém třeba nastal, když zlomek byl dříve než konec odmocniny, pak se musel přehodit, aby odmocnina byla uzavřena, poté se mohl nakreslit znak zlomkové čary. Dále nesmí chybět stejný počet počátečních a koncových značek. Kontrolují se úseky od jednoho nulového vnoření k druhému. Jen v těchto úsecích můžu zjistit stejný počet počátečních a koncových značek, pokud chybí, pak se v těchto úsecích doplní. Nulové úseky vnoření totiž vždy zajišťují, že na tomto místě začíná nový element.

3. kontrola, zda je dolní nebo horní index, jelikož tuto značku vždy uvozuje mrow, který je stejný pro horní i dolní index, ale je vnořen buď do msup nebo msub entity, což určuje správný tvar. Nechaly se při analýze znaky msub, msup a mrow v řetězci, nenahrazovaly se. Poté se muselo zjistit jaký řetězec je v řetězci před mrow, rozsekal jsem tedy kompletně sestavený řetězec na podřetězce oddělené právě znaky mrow a na nich jsem zjišťoval zda obsahuji „msup“ nebo „msub“, po zjištění správné entity v tomto podřetězci se mohlo mrow změnit na správný znak, který buď uvazuje, že se jedná o dolní nebo horní index. A msub a msup se mohl nahradit korektními znaky, aby z toho vznikl obrázek v PHPMathPublisheru

Skript mathmltopdf.php:

```
<?php
include "settings.php";
$hodnota=1;
$result = mysql_query('SELECT * FROM structure order by id');

// 1. analýza
while($links = mysql_fetch_array($result)) {

if ($links['tag']=='msup') {
      $levsup=$links['level']; $znaksup=0; $znsup='sup';}
```

```
if ($links['tag']=='msub') {
      $levsub=$links['level']; $znaksub=0; $znsub='sub';}

if ($links['tag']=='msubsup') {
      $znacka=''; $levsubsup=$links['level']; $znaksubsup=0;
$zn='subsup';}

if ($links['tag']=='mfrac') {
      $znacka=''; $levfrac=$links['level']; $znakfrac=0;
$znfr='frac';}

if ($links['tag']=='msqrt') {
      $znacka=''; $lev=$links['level']; $znak=0;
$znakzab=0;$znsqrt='sqrt';}

      if ($links['tag']=='mrow') {
      $levrow=$links['level']; $znakrow=0; $znrow='row';}


// označit tág na nulovém vnoření (nevnořený)
if ($links['level']==0) {$str=$str.'|';}

if ($links['level']<=$levsup) {$znaksup++;
if ($znaksup==1) {$str=$str.';sup;';}
if ($znaksup==2) {$str=$str.';/sup;';}}

if ($links['level']<=$levsub) {$znaksub++;
if ($znaksub==1) {$str=$str.';sub;';}
if ($znaksub==2) {$str=$str.';/sub;';}}

if ($links['level']<=$levsubsup) {$znaksubsup++;
if ($znaksubsup==1) {$str=$str.';subsup;';}
if ($znaksubsup==2) {$str=$str.';/subsup;';}}

if ($links['level']<=$levrow) {$znakrow++;
if ($znakrow==1) {$str=$str.';row;';}
```

```
}

if ($links['level']<=$levfrac) {$znakfrac++;
 if ($znakfrac==1) {$str=$str.';frac;'; $znfrac='fraczac';}
 if ($znakfrac==2) {$str=$str.';/frac;';$znfrac='frackon';}}

     if (($znfrac=='fraczac')and($links['level']==$levfrac+1))
{$znzlomek++; if ($znzlomek==2) {$str=$str.';zlomek;';
$znzlomek=0;$znfrac=='';}}


if ($links['level']<=$lev) {$znak++;
if ($znak==1) {$str=$str.';sqrt;';$znsqrt='sqrtzac';}
if ($znak==2) {$str=$str.';/sqrt;'; $znsqrt='sqrtkon';}}



$str=$str.''.$links['value'].'';

}
// 2. analýza

$str=$str.'|';
//odstranit na začátcích nevnořených tágů (nulových) končící
tágy
$str=str_replace('|;/frac;','|', $str);
$str=str_replace('|;/sqrt;','|', $str);
$str=str_replace('|;/sup;','|', $str);
$str=str_replace('|;/sub;','|', $str);

// rozdělit na úseky, který začíná a končí nulově vnořenými
elementy
$pocetvyskytu=substr_count($str,'|');

$zac=strpos($str,'|');
$delka=0;
$pozice=0;
```

```
while ($pozice<$pocetvyskytu) {
$retezec[$pozice]=substr($str,$delka,$zac);
$delka=$delka+strlen($retezec[$pozice])+1;
$ret=substr($str,$delka);
$zac=strpos($ret,'|');
$pozice++;
}

// analyzovat úseky
$a=0;
while ($a<$pocetvyskytu) {

//nesmyslné značky opravit
$upraven[$a]=str_replace(';zlomek;;/sqrt;',';/sqrt;;zlomek;',
$retezec[$a]);

$upraven[$a]=str_replace('zlomek;;sqrt;',';/sqrt;;zlomek;;sqrt;'
, $upraven[$a]);

//počítat výskyt daných počátečních a končících značek
$pocetvsqrt=1;
  $q = ';sqrt;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsqrt++;
          $o = $p + 1;
  }

$pocetvsqrtk=1;
    $q = ';/sqrt;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsqrtk++;
          $o = $p + 1;
  }

 $pocetvfrac=1;
```

```
    $q = ';frac;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvfrac++;
          $o = $p + 1;
  }

 $pocetvfrack=1;
    $q = ';/frac;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvfrack++;
          $o = $p + 1;
  }

 $pocetvsup=1;
    $q = ';sup;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsup++;
          $o = $p + 1;
  }

 $pocetvsub=1;
    $q = ';sub;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsub++;
          $o = $p + 1;
  }

 $pocetvsupk=1;
    $q = ';/sup;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsupk++;
          $o = $p + 1;
```

```
  }

 $pocetvsubk=1;
    $q = ';/sub;';
  $o = 0;
  while (($p = strpos($upraven[$a], $q, $o)) !==  FALSE) {
$pocetvsubk++;
          $o = $p + 1;
  }



//doplnit chybějící koncové značky

for ($i=0;$i<($pocetvsqrt-$pocetvsqrtk);++$i)
{ $pridat[$a]=$pridat[$a].'}';}


for ($i=0;$i<($pocetvfrac-$pocetvfrack);$i++)
{ $pridat[$a]=$pridat[$a].'}';}


for ($i=0;$i<($pocetvsup-$pocetvsupk);$i++)
{ $pridat[$a]=$pridat[$a].'}';}

for ($i=0;$i<($pocetvsub-$pocetvsubk);$i++)
{ $pridat[$a]=$pridat[$a].'}';}

//převést zbývající správné značky na korektní symboly do
řetezce

$upraven[$a]=str_replace(';sqrt;','sqrt{', $upraven[$a]);
$upraven[$a]=str_replace(';frac;','{', $upraven[$a]);
$upraven[$a]=str_replace(';zlomek;','}/{', $upraven[$a]);
$upraven[$a]=str_replace(';/sqrt;','}', $upraven[$a]);
$upraven[$a]=str_replace(';/frac;','}', $upraven[$a]);
$upraven[$a]=str_replace(';/sup;','}', $upraven[$a]);
```

```
$upraven[$a]=str_replace(';/sub;','}', $upraven[$a]);
$upraven[$a]=str_replace(';','', $upraven[$a]);
$upraven[$a]=str_replace(" ","", $upraven[$a]);
$upraven[$a]=str_replace("\t","", $upraven[$a]);
$upraven[$a]=str_replace("\n","", $upraven[$a]);
$upraven[$a]=str_replace("\r","", $upraven[$a]);
$upraven[$a]=str_replace("\0","", $upraven[$a]);
$upraven[$a]=str_replace("\x0B","", $upraven[$a]);


$teduprаven[$a]=$upraven[$a].$pridat[$a];

$a++;
}

$dan=0;
while ($dan<$pocetvyskytu) {
$message=$message.''.$teduprаven[$dan+1];
$dan++;
}
// zjištění horních a dolních indexu

$dily = explode("row", $message);
substr_count($message, "row");

for ($i=0;$i<substr_count($message, "row");$i++)
{
if (strpos($dily[$i], 'sup')==true) {
$umistit[$i]='^{';
$dily[$i]=str_replace("sup","", $dily[$i]);
} else {$umistit[$i]='_{';
$dily[$i]=str_replace("sub","", $dily[$i]);
}
$dily[$i]=str_replace("sup","", $dily[$i]);
}

$messageindexy=$message;
```

```
$message='';
for ($i=0;$i<(substr_count($messageindexy, "row")+1);$i++)
{
$message.=$dily[$i].''.$umistit[$i];
}
include("prevodgifpdf.php") ;

//převést do obrázku a následné do PDF
mathfilter("<m>".$message."</m>","14",TMPIMG);
?>
```

Výsledek zobrazení řetězce:

```
sqrt{{2+sqrt{x}}/{5*g}}-{sqrt{2-x^{2-
y}+5}}/{sqrt{3}}+x^{2}+{x}/{2-x^{3}}
```

Převedení řetězce do grafické podoby:

$$\sqrt{\frac{2+\sqrt{x}}{5*g}}-\frac{\sqrt{2-x^{2-y}+5}}{\sqrt{3}}+x^{2}+\frac{x}{2-x^{3}}$$

6.3 Rozhraní

Soubor pro vložení MathML dokumentu na převod se jmenuje index.php, je v něm formulář, který umožní uploadování souboru na server:

```
<h1>MathML to PDF</h1>
Vyberte soubor s MathML strukturou pro převod do PDF:
<form action="savexmlsoub.php" METHOD="post" ENCTYPE="multipart/form-
data">
<input type="hidden" name="send" value="ok">
<input type="file" name="soubor">
<input type="submit" value="Odeslat">
</form>
```

Poté se volá skript savexmlsoub.php, který vymaže data z databáze, uploaduje soubor na server do nastaveného adresáře, vloží strukturu souboru do databáze a nakonec se provede skript mathmltopdf.php popsaný v Kapitole 6.2.3, který vytvoří obrázek a vloží do PDF.

Savexmlsoub.php:

```
<?php
import_request_variables('GPC');
include "settings.php";
mysql_query('TRUNCATE TABLE structure');
if (is_uploaded_file($_FILES["soubor"]["tmp_name"])) {
  $name = $_FILES["soubor"]["name"];
  move_uploaded_file($_FILES["soubor"]["tmp_name"],
TMPUPLOAD."$name");
}
$sx = simplexml_load_file(TMPUPLOAD."$name");

$level = 0;

NLevel($sx,$level);

function NLevel($element,$level)
{
  static $line = 0;
```

```
  foreach ($element->children() as $tags => $child)
    {
   //    print ++$line.". ".$tags . "->$child ($level)<br/>";
mysql_query('INSERT INTO structure SET id='.$line.',
tag="'.$tags.'", value="'.$child.'", level='.$level.'');
       NLevel($child,$level+1);
   }
}
include 'mathmltopdf.php';
?>
```

Definice cest adresářů na serveru a přístupu k databázi se nachází v souboru settings.php, adresář pro nahrávání MathML souboru a pro tvorbu obrázků musí mít na serveru nastaveno právo čtení i zápisu, tzn. chmod 777. Uploadované soubory a vytvořené obrázky na serveru zůstávají, po provedení převodu se můžou smazat ručně.

Settings.php:

```
<?php
//nastaveni databaze
$db = mysql_connect("localhost", "test", "test");
mysql_select_db("data",$db);

// nastaveni adresaru
define('TMPUPLOAD','data/');  //prava i pro zapis, CHMOD 777
define('FONTY','fonts/');
define('TMPIMG','img/'); //prava i pro zapis, CHMOD 777
?>
```

Pro převod obrázku vytvořeného přes PHPMathPublisher do PDF se modifikovala drobně funkce „MathImage“ v knihovně tak, aby se přímo vkládal obrázek do PDF, modifikovaný skript prevodgifpdf.php najdete na přiloženém CD.

7 Závěr

Cílem práce bylo navrhnout novou metodu automatického převodu obecného XML do uživatelsky čitelného PDF dokumentu.

Tato metoda využívá stávající funkce a knihovny skriptovacího jazyka PHP 5, které jsou zde vysvětleny a dále navržen postup jejich použití.

Aplikaci navržené metody jsem názorně uvedl v praktické úloze měření teploty a doplňkové úloze převodu MathML do PDF.

Výběr už je na čtenářích, kteří se s touto diplomovou prací seznámí, čemu dají přednost, zda využijí pro svůj převod z XML do PDF, buď mou variantu se SimpleXML a FPDF nebo jinou, např. DOM a PDFlib? Tak či tak, nová metoda je vytvořena s využitím možností webového serveru se skriptovacím jazykem PHP 5.

V praktické úloze převodu MathML do PDF jsem také zjistil, že důležité je ukládat nejen jméno tágu, jeho hodnotu, případně jejich atributy, ale i hodnotu vnoření daného elementu do druhého, protože bez této informace bych nemohl korektně převézt obecné XML dokumenty a věřím, že rekurzivní postup na čtení vnořených tágů pomocí SimpleXML uveden v této práci bude přínosem.

Také jsem si vědom, že jazyk MathML má daleko více prezentačních značek než používám zde. Mým cílem nebylo vytvořit kompletní převod MathML do PDF (to by byla nová diplomová práce), ale jen další ukázku aplikace postupu převodu XML do PDF pro důkaz širšího uplatnění.

Veškeré skripty, ať už popsány v této práci nebo jen nutné pro funkčnost praktických úloh s vysvětlením instalace a diplomovou práci ve formátu PDF najdete na přiloženém CD.

8 Seznam použitých zdrojů

[1] YOUNG, Michael J., XML krok za krokem, Mobil Media a.s., Praha, 2002

[2] CASTAGNETTO, J., RAWAT, H., SCHUMANN, S., SCOLLO, Ch., VELIATH, D., Programujeme PHP profesionálně, Computer Press, Praha, 2002

[3] SIEVER, E. a kol., Linux v kostce, Computer Press, Praha, 1999

[4] STOCKER Ch., XML in PHP5 - What's new? (http://www.zend.com/php5/articles/php5-xmlphp.php - 3. 12. 2006)

[5] PHP Manuál - http://www.php.net/ (3. 12. 2006)

[6] XML pro každého - http://www.kosek.cz/xml/ (3. 12. 2006)

[7] W3C - http://www.w3.org/XML/ (3. 12. 2006)

[8] Adobe PDF Technology Center http://partners.adobe.com/public/developer/pdf/topic.html (3. 12. 2006)

[9] FPDF – http://www.fpdf.org (3. 12. 2006)

[10] WebGURU.cz - Generování dokumentů PDF pomocí PHP a fpdf.php http://www.webguru.cz/clanky/view.php?id=93 (10. 9. 2006)

[11] WampServer – http://www.wampserver.com/en/index.php (3. 12. 2006)

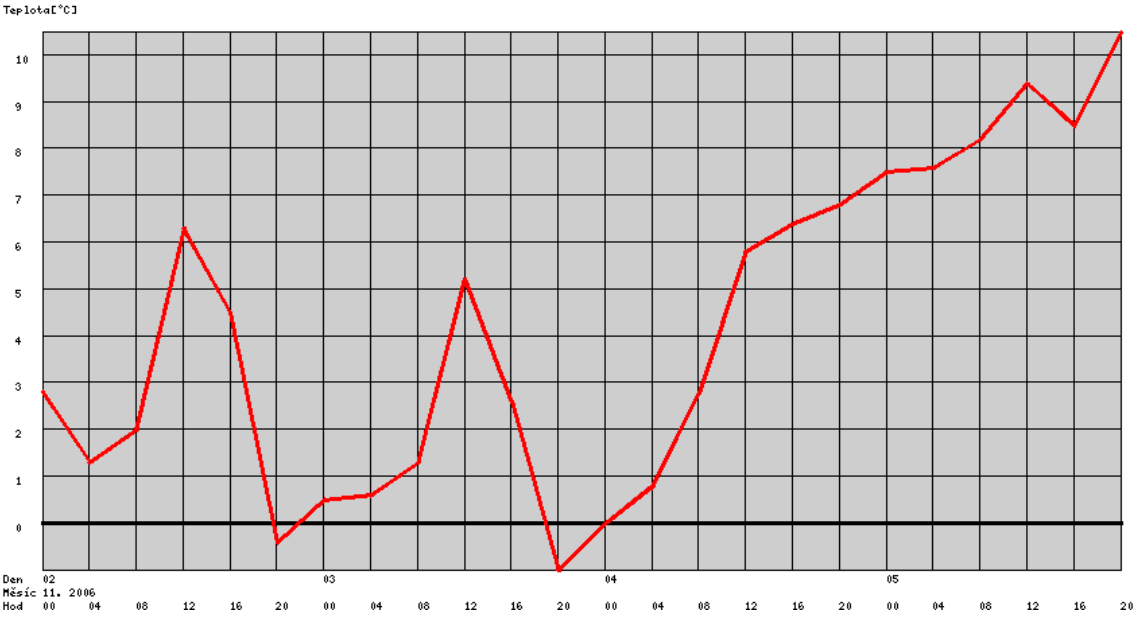
[12] Interval.cz - K čemu je nám MathML http://interval.cz/clanky/k-cemu-je-nam-mathml (3.12.2006)

[13] PHPMathPublisher - http://www.xm1math.net/phpmathpublisher (3. 12. 2006)

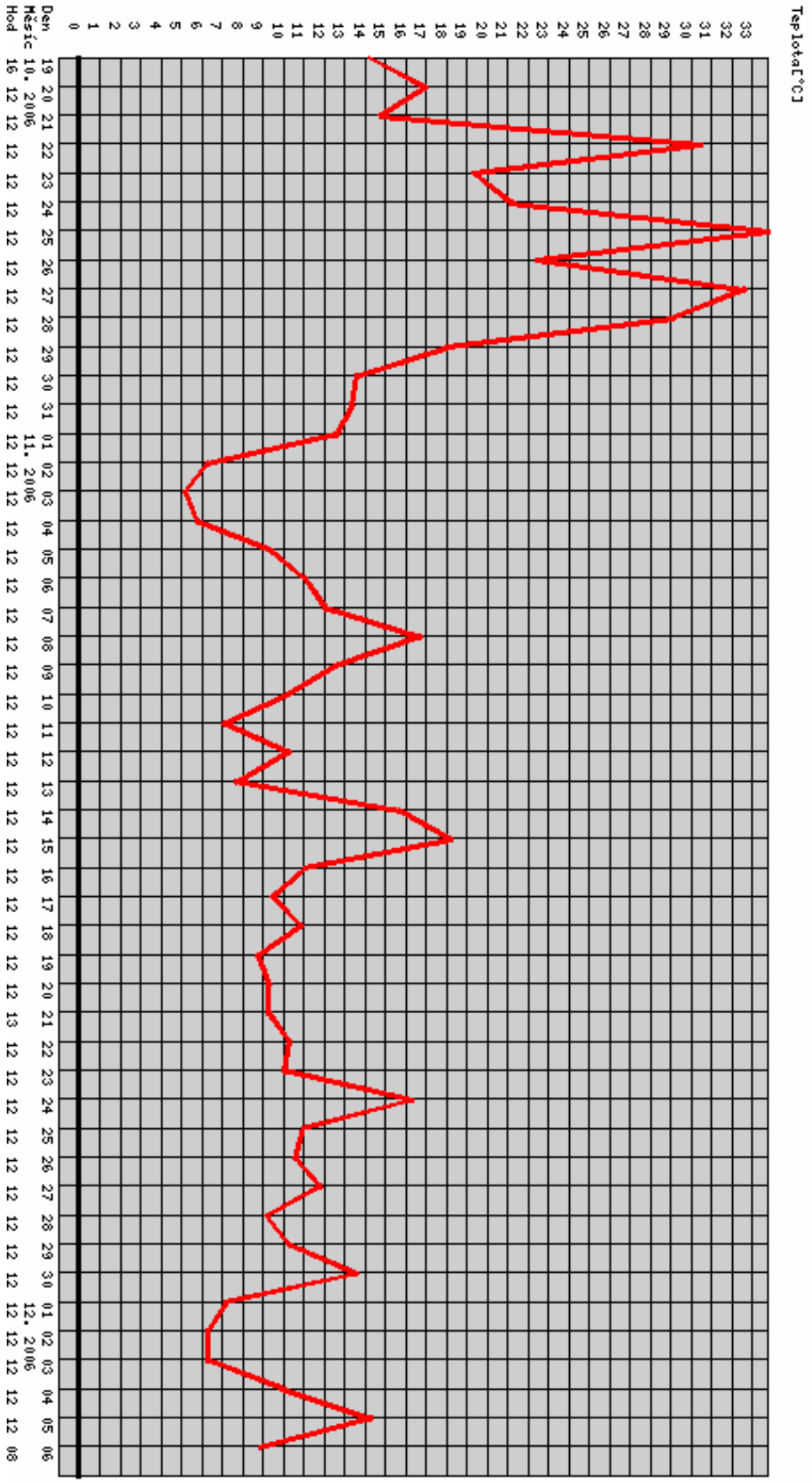
9 Přílohy

Příloha 1: Příklad výstupu PDF – měření teploty

Den	Měsíc	Rok	Čas	Teplota [°C]
02	11	2006	00:00	2.8
02	11	2006	04:00	1.3
02	11	2006	08:00	2.0
02	11	2006	12:00	6.3
02	11	2006	16:00	4.5
02	11	2006	20:00	-0.4
03	11	2006	00:00	0.5
03	11	2006	04:00	0.6
03	11	2006	08:00	1.3
03	11	2006	12:00	5.2
03	11	2006	16:00	2.6
03	11	2006	20:00	-1.0
04	11	2006	00:00	0.0
04	11	2006	04:00	0.8
04	11	2006	08:00	2.8
04	11	2006	12:00	5.8
04	11	2006	16:00	6.4
04	11	2006	20:00	6.8
05	11	2006	00:00	7.5
05	11	2006	04:00	7.6
05	11	2006	08:00	8.2
05	11	2006	12:00	9.4
05	11	2006	16:00	8.5
05	11	2006	20:00	10.5



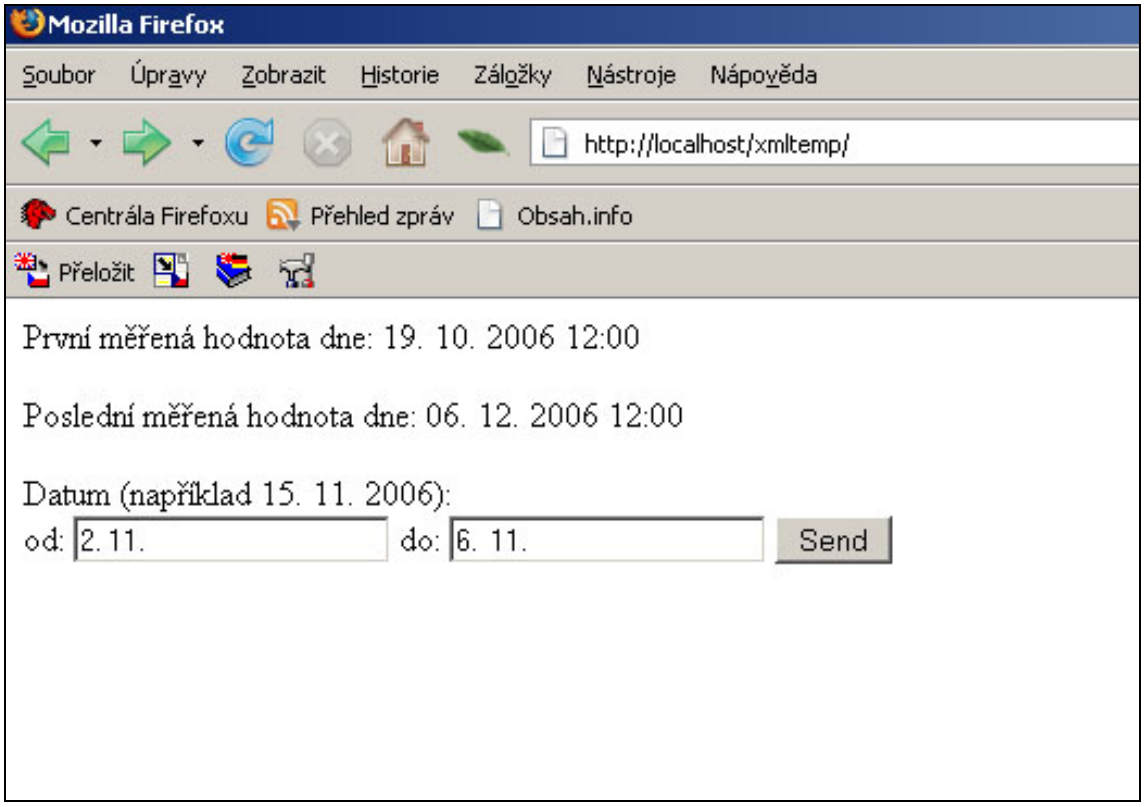
Vzorkovaný graf - zobrazuje se jen jedna denní naměřená hodnota



Příloha 2: Skript skriptteplota.php – čtení XML a ukládání do souboru

```
<?php

//vytvorit spojeni
      $ff = FSockopen("ctrlv4.cs.cas.cz",80);

// nastaveni timeoutu pro spojeni
      stream_set_timeout($ff,0,500000);

//nacist XML
      FPutS($ff,"GET /temper.xml HTTP/1.0\r\n\r\n");

// precist vse pred tagem <?xml
&agrave;    while (($t=FGetS($ff)) && (substr($t,0,5) !=
"<?xml")) {

// overit zda spojeni nevytimeoutovalo
            $info = stream_get_meta_data($ff);
            if ($info['timed_out']) {
                  $continue = 1;
                  break;
            }
      }
// pokud spojeni timeoutovalo - cteme znova
      if ($continue) continue;


// precist zbytek dokumentu

      $text = $t;
      while ($t=FGetS($ff)) {

            $text .= $t;

            $info = stream_get_meta_data($ff);
            if ($info['timed_out']) {
                  $continue = 1;
                  break;
            }
      }
// pokud spojeni timeoutovalo, dokument neni cely - cteme znova
     if ($continue) continue;


// v promenne „text" cely xml dokument a ulozime jej do souboru
$wfip=fopen('/home/people/mrazekk/public_html/'.date("YmdHi").'.
xml',"w");
FputS($wfip, $text);
Fclose($wfip);
?>
```

Příloha 3: Skript index.php – rozhraní



obr. - ukázka uživatelského rohraní

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
      <title></title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1250">
</HEAD>
<body>
<?php
error_reporting(0);
import_request_variables('GPC');
$pocetzjisteni=0;
//zjisteni souboru
$handle=opendir("teplota/");
for ($i=0; $file = readdir($handle); $i++)
{
      if (strlen($file)>2)
      {
            $xml = simplexml_load_file("teplota/".$file);
            $hodnota[$pocetzjisteni]=$xml->te0[0]->val;

            $rok[$pocetzjisteni]=substr($file,0,4);
            $mesic[$pocetzjisteni]=substr($file,4,2);
            $den[$pocetzjisteni]=substr($file,6,2);
            $hodina[$pocetzjisteni]=substr($file,8,2);
            $minuta[$pocetzjisteni]=substr($file,10,2);
            $pocetzjisteni++;
```

```
     }
}
echo '<p></p>';
closedir($handle);
//prvni merena hodnota
echo 'První měřená hodnota dne: ';
echo $den[0].'. '.$mesic[0].'. '.$rok[0].'
'.$hodina[0].':'.$minuta[0];

//posledni merena hodnota
echo '<p></p>Poslední měřená hodnota dne: ';
echo $den[$pocetzjisteni-1].'. '.$mesic[$pocetzjisteni-1].'.
'.$rok[$pocetzjisteni-1].'  '.$hodina[$pocetzjisteni-
1].':'.$minuta[$pocetzjisteni-1];

//vstupni formular
?>
<p></p>
Datum (například 15. 11. 2006):
<form action="pdf.php">
<input type="hidden" name="send" value="ok">
od:
<input type="text" name="datum">
do:
<input type="text" name="datumdo">
<input type="hidden" name="datumnejmensi" value="<?php echo
$datum=$rok[0].''.$mesic[0].''.$den[0].''.$hodina[0].''.$minuta[
0];?>">
<input type="hidden" name="datumnejvetsi" value="<?php echo
$rok[$pocetzjisteni-1].''.$mesic[$pocetzjisteni-
1].''.$den[$pocetzjisteni-1].''.$hodina[$pocetzjisteni-
1].''.($minuta[$pocetzjisteni-1]+10);?>">
<input type="submit" value="Send">
</form>
</body>
</html>
```

Příloha 4: Skript pdftable.php - rozšíření FPDF pro tabulky

```
<?php
require('fpdf.php');

class PDF_table extends FPDF
{
var $widths;
var $aligns;

function SetWidths($w)
{
    //nastavit pole sirky sloupcu
    $this->widths=$w;
}

function SetAligns($a)
```

```
{
    //nastavit pole zarovnani sloupcu
    $this->aligns=$a;
}

function Row($data)
{
    //vypocitat sirku radku
    $nb=0;
    for($i=0;$i<count($data);$i++)
        $nb=max($nb,$this->NbLines($this-
>widths[$i],$data[$i]));
    $h=5*$nb;
    //overeni
    $this->CheckPageBreak($h);
    //nakreslit bunky na radku
    for($i=0;$i<count($data);$i++)
    {
        $w=$this->widths[$i];
        $a=isset($this->aligns[$i]) ? $this->aligns[$i] : "R";
        //ulozit aktualni pozici
        $x=$this->GetX();
        $y=$this->GetY();
        //nakreslit oramovani
        $this->Rect($x,$y,$w,$h);
        //text
        $this->MultiCell($w,5,$data[$i],0,$a);
        $this->SetXY($x+$w,$y);
    }
    //jit na dalsi
    $this->Ln($h);
}

function CheckPageBreak($h)
{
    //jestlize vyska h muze pretect, pak vytvorit novou stranku
    if($this->GetY()+$h>$this->PageBreakTrigger)
        $this->AddPage($this->CurOrientation);
}

function NbLines($w,$txt)
{
    //pocitat pocet radek v bunce sirky
    $cw=&$this->CurrentFont['cw'];
    if($w==0)
        $w=$this->w-$this->rMargin-$this->x;
    $wmax=($w-2*$this->cMargin)*1000/$this->FontSize;
    $s=str_replace("\r",'',$txt);
    $nb=strlen($s);
    if($nb>0 and $s[$nb-1]=="\n")
        $nb--;
    $sep=-1;
    $i=0;
    $j=0;
    $l=0;
    $nl=1;
```

```
    while($i<$nb)
    {
        $c=$s[$i];
        if($c=="\n")
        {
            $i++;
            $sep=-1;
            $j=$i;
            $l=0;
            $nl++;
            continue;
        }
        if($c==' ')
            $sep=$i;
        $l+=$cw[$c];
        if($l>$wmax)
        {
            if($sep==-1)
            {
                if($i==$j)
                    $i++;
            }
            else
                $i=$sep+1;
            $sep=-1;
            $j=$i;
            $l=0;
            $nl++;
        }
        else
            $i++;
    }
    return $nl;
}
}
?>
```

Příloha 5: Skript settings.php, pdf.php – tvorba výsledku v pdf a grafu

settings.php:

```
<?php
define('ADRESARSOUBORU','teplota/');
define('VYSTUPGRAFU','graf/graf.png'); //prava zapisu, chmod 777
define('FPDF_FONTPATH',dirname(__FILE__).'/fpdf_font/');

//DEFINE("popisx",5);
DEFINE("mezera", 30);
DEFINE("grafmax_y",400);
DEFINE("grafmax_x",800);
DEFINE("max_x", mezera + grafmax_x + mezera);
DEFINE("max_y", mezera + grafmax_y + mezera);

DEFINE("max_mezera_x", 9);
DEFINE("min_mezera_xpopis", 12);
```

```
DEFINE("pocetmereni_pro_zmenu_vzorku", 720);   // 720/6 = 120
dni
?>
```

pdf.php:

```
<?php
include "settings.php";
//vypnuti chybovych hlaseni, pro pripad neulozeneho xml
error_reporting(0);
//nastaveni nezavislosti serveru na register_globals, muze byt
OFF nebo ON, je to jedno
import_request_variables('GPC');

// funkce pro tloustku cary
function imagelinethick($image, $x1, $y1, $x2, $y2, $color,
$thick = 1)
{
   if ($thick == 1) {
       return imageline($image, $x1, $y1, $x2, $y2, $color);
   }
   $t = $thick / 2 - 0.5;
   if ($x1 == $x2 || $y1 == $y2) {
       return imagefilledrectangle($image, round(min($x1, $x2) -
$t), round(min($y1, $y2) - $t), round(max($x1, $x2) + $t),
round(max($y1, $y2) + $t), $color);
   }
   $k = ($y2 - $y1) / ($x2 - $x1); //y = kx + q
   $a = $t / sqrt(1 + pow($k, 2));
   $points = array(
       round($x1 - (1+$k)*$a), round($y1 + (1-$k)*$a),
       round($x1 - (1-$k)*$a), round($y1 - (1+$k)*$a),
       round($x2 + (1+$k)*$a), round($y2 - (1-$k)*$a),
       round($x2 + (1-$k)*$a), round($y2 + (1+$k)*$a),
   );
   imagefilledpolygon($image, $points, 4, $color);
   return imagepolygon($image, $points, 4, $color);
}

//pokud se odeslou vstupni data
if ($send=='ok')
{
     //nadefinovani PDF, vlozeni fontu
     require('pdftable.php');
     $pdf=new PDF_table('P','mm','A4');
     $pdf->AddFont('Arial','','arial.php');
     $pdf->AddFont('Arial','B','arialbold.php');
     $pdf->AddPage();
     $pdf->SetFont('Arial','B',8);

     //zjisteni, jak byl zadan datum a prevedeni
     if ($datum=='') { $datum=$datumnejmensi;}
     else
     {
     $datum=str_replace(" ","",$datum);
          $ro=explode(".",$datum);
```

```
            if ($ro[2]=='') { $ro[2]=date("Y");}
            if ($ro[1]=='') { $ro[1]=Date("n");}
            if (strlen($ro[0])<2) $ro[0]='0'.$ro[0];
            if (strlen($ro[1])<2) $ro[1]='0'.$ro[1];
            $datum=$ro[2].''.$ro[1].''.$ro[0].'0000';
      }

      if ($datumdo=='') { $datumdo=$datumnejvetsi;}
      else
      {
      $datumdo=str_replace(" ","",$datumdo);
            $rodo=explode(".",$datumdo);
            if ($rodo[2]=='') { $rodo[2]=date("Y");}
            if ($rodo[1]=='') { $rodo[1]=Date("n");}
            if (strlen($rodo[0])<2) $rodo[0]='0'.$rodo[0];
            if (strlen($rodo[1])<2) $rodo[1]='0'.$rodo[1];
            $datumdo=$rodo[2].''.$rodo[1].''.$rodo[0].'2300';
      }


//zacatek grafu

$pocet=0;
$handle=opendir(ADRESARSOUBORU);
for ($i=0; $file = readdir($handle); $i++)
{
      if (strlen($file)>2)
      {
            //podminka pro datum
            if
(($datum<=substr($file,0,12))and($datumdo>substr($file,0,12)))
            {
                  $xml =
simplexml_load_file(ADRESARSOUBORU.$file);
                  $hodnota[$pocet]=floatval($xml->te0->val);
                  $rok[$pocet]=substr($file,0,4);
                  $mesic[$pocet]=substr($file,4,2);
                  $den[$pocet]=substr($file,6,2);
                  $hodina[$pocet]=substr($file,8,2);
                  $minuta[$pocet]=substr($file,10,2);
                  $pocet++;
            }
      }
}
$pocetsouboru=$pocet;
closedir($handle);

//nastaveni grafu
header("content_type: image/png");

$img=ImageCreate(max_x,max_y);

// nejsou udaje k vytvoreni grafu
if ($pocetsouboru<2)
{
      $bgcolor=ImageColorAllocate($img,0xFF,0xFF,0xFF);
```

```
      ImageFill($img,60,60,$bgcolor);
      $lncolor=ImageColorAllocate($img,0,0,0);
      ImageString($img,4,100,100,'Nejsou udaje k vytvoreni
grafu',$lncolor); //pise mezeru
      ImagePNG($img, VYSTUPGRAFU);
}
else
{
      //barva pozadi
      $bgcolor=ImageColorAllocate($img,0xFF,0xFF,0xFF);
      //barva textu
      $lncolor=ImageColorAllocate($img,0,0,0);
      //popisky os
      ImageString($img,1,1,10,'Teplota[°C]',$lncolor);
      ImageString($img,1,1,max_y - mezera+3,'Den',$lncolor);
      ImageString($img,1,1,max_y - mezera+13,'Měsíc',$lncolor);
      ImageString($img,1,1,max_y - mezera+23,'Hod',$lncolor);

      //vytvorime ohraniceny obdelnik 50px od okraje obrazku
      ImageLine($img,mezera,mezera,max_x-mezera,mezera,$lncolor);
//horni cara
      ImageLine($img,mezera,max_y - mezera,max_x-mezera,max_y-
mezera,$lncolor); //dolni cara

      ImageLine($img,mezera,mezera,mezera,max_y -
mezera,$lncolor); //leva bocni cara
      ImageLine($img,max_x - mezera,mezera,max_x - mezera,max_y -
mezera,$lncolor); //prava bocni cara

      //barva pozadi grafu
      $color = ImageColorAllocate($img,0xCE,0xCE,0xCE);

      //vyplneni obrazce barvou - pokud je neuzavreny obrazec,
pak program vymaluje celou plochu.
      ImageFill($img,60,60,$color);

      $grcolor = ImageColorAllocate($img,0xFF,0x00,0x00);

      $mezerax=(grafmax_x/($pocetsouboru-1));
//ImageString($img,4,100,100,$mezerax,$lncolor); //pise mezeru
if ($mezerax<max_mezera_x)
{
      // vzorkovani grafu, maximum az 50*365 souboru, 50let
      for ($pocetkrat=0;$pocetkrat<50;++$pocetkrat)
      {
            if
(($pocetsouboru>($pocetkrat*pocetmereni_pro_zmenu_vzorku))and($p
ocetsouboru<(($pocetkrat+1)*pocetmereni_pro_zmenu_vzorku))) {
$pocetpreskoku=($pocetkrat+1);};
      }
      //vlozeni x_popisku
      $a=0;
      $ap=0;
      while ($a < $pocetsouboru)
      {
            if ($den[$a]<>$den[$a-1])
```

```
            {
                  $prumer[$ap]=$a;
                  $ap++;
            }
            $a++;
      }
      $prumer[$ap]=$pocetsouboru;

      $ap=1;
      while ($ap < count($prumer))
      {
            $prumerindex=floor(($prumer[$ap]+$prumer[$ap-1])/2);
            $prumeri[$ap]=$prumerindex;
            $ap++;
      }

      $mezerax=(grafmax_x/floor((count($prumer)/$pocetpreskoku)-
1));


      $pocetd=1;
      while ($pocetd < (count($prumer)/$pocetpreskoku))
      {
      $hodnotavz[$pocetd]=$hodnota[$prumeri[$pocetd*$pocetpreskok
u]];
            ImageString($img,1,mezera + ($mezerax * ($pocetd-1))
,max_y -
mezera+3,$den[$prumeri[$pocetd*$pocetpreskoku]],$lncolor);

            if
($mesic[$prumeri[$pocetd*$pocetpreskoku]]<>$mesic[$prumeri[$poce
td*$pocetpreskoku-(1*$pocetpreskoku)]])
            {
                  ImageString($img,1,mezera + ($mezerax *
($pocetd-1)) ,max_y -
mezera+13,$mesic[$prumeri[$pocetd*$pocetpreskoku]].'.
'.$rok[$prumeri[$pocetd*$pocetpreskoku]],$lncolor);
            }

            if ($mezerax>min_mezera_xpopis)
            {
                  ImageString($img,1, mezera + ($mezerax *
($pocetd-1)) ,max_y - mezera
+23,$hodina[$prumeri[$pocetd*$pocetpreskoku]],$lncolor);
            }

            ImageLine($img,mezera+($mezerax * ($pocetd-
1)),mezera,mezera+($mezerax * ($pocetd-1)),max_y -
mezera,$lncolor);
            $pocetd++;
      }

$pr=0;
      $max=floatval(MAX($hodnota));
      $min=floatval(MIN($hodnota));
      $y_value=($max - $min) / grafmax_y ;
```

```
      $stupen=($max - $min);
      $mezeray=(grafmax_y/($max - $min));

// vodorovne cary
      $l=1;
while($l<=$stupen)
      {
            ImageLine($img,mezera,max_y - mezera - $mezeray *
$l,max_x-mezera,max_y -mezera - $mezeray * $l,$lncolor);
            $l++;
      }
// nulova cara
      if (($min<0)and($max>0)) {
ImageLinethick($img,mezera,max_y - mezera - $mezeray *
abs($min),max_x-mezera,max_y -mezera - $mezeray *
abs($min),$lncolor,3);
}
      $d = 1;

if ($mezeray<11) {$kroky=2;}
if ($mezeray>11) {$kroky=1;}
      while ($d<=($stupen/$kroky))
      {
$osay[$d]=round (($y_value * $kroky*$mezeray * $d) + $min);
ImageString($img,1,mezera - 20,max_y - mezera - $kroky*$mezeray
* $d,$osay[$d],$lncolor);
            $d++;
      }

      $pocetd=1;
      while ($pocetd < ((count($prumer))/$pocetpreskoku)-1)
      {
ImageLinethick($img,mezera + ($mezerax * ($pocetd-1)),max_y -
mezera - (($hodnota[$prumeri[$pocetd*$pocetpreskoku]]-
$min)/$y_value),mezera + ($mezerax * (($pocetd-1) + 1)),max_y -
mezera -
(($hodnota[$prumeri[($pocetd*$pocetpreskoku)+$pocetpreskoku]]-
$min)/$y_value),$grcolor,3);
            $pocetd++;
      }

      ImagePNG($img, VYSTUPGRAFU);

// konec vzorkovani
}
else
//zacatek bez vzorkovani
{
      $a=0;
      while ($a < $pocetsouboru)
      {
            if ($den[$a]<>$den[$a-1])
            {
                  ImageString($img,1,mezera + ($mezerax * $a)
,max_y - mezera + 3 ,$den[$a],$lncolor);
            }
```

```
            if ($mesic[$a]<>$mesic[$a-1])
            {
                  ImageString($img,1,mezera + ($mezerax * $a)
,max_y - mezera + 13 ,$mesic[$a].'. '.$rok[$a],$lncolor);
            }
            if ($mezerax>min_mezera_xpopis)
            {
                  ImageString($img,1, mezera + ($mezerax * $a)
,max_y - mezera + 23,$hodina[$a],$lncolor);
            }
            ImageLine($img,mezera+($mezerax *
$a),mezera,mezera+($mezerax * $a),max_y - mezera,$lncolor);
            $a++;
      }


      $pr=0;
      $max=floatval(MAX($hodnota));
      $min=floatval(MIN($hodnota));
      $y_value=($max - $min) / grafmax_y ;
      $stupen=($max - $min);
      $mezeray=(grafmax_y/($max - $min));

      $l=1;
//vodorovne cary
      while($l<=$stupen)
      {
            ImageLine($img,mezera,max_y - mezera - $mezeray *
$l,max_x-mezera,max_y -mezera - $mezeray * $l,$lncolor);
            $l++;
      }
      if (($min<0)and($max>0)) {
ImageLinethick($img,mezera,max_y - mezera - $mezeray *
abs($min),max_x-mezera,max_y -mezera - $mezeray *
abs($min),$lncolor,3);
}
      $d = 1;

      while ($d<=$stupen)
      {
$osay[$d]=round (($y_value * $mezeray * $d) + $min);
ImageString($img,1,mezera - 20,max_y - mezera - $mezeray *
$d,$osay[$d],$lncolor);
            $d++;
      }

      while ($pr<$pocetsouboru-1)
      {
      ImageLinethick($img,mezera + ($mezerax * $pr),max_y -
mezera - (($hodnota[$pr]- $min)/$y_value),mezera + ($mezerax *
($pr + 1)),max_y - mezera - (($hodnota[($pr+1)]-
$min)/$y_value),$grcolor,3);
            $pr++;
      }

      ImagePNG($img, VYSTUPGRAFU);
```

```
//konec bez vzorkovani
}

//konec obrazku
}

$pdf->SetWidths(array(20,20,20,20,20));
$pdf->Row(array('Den','Měsíc','Rok','Čas', 'Teplota [°C]'));
$pdf->SetFont('Arial','',8);
$pdf->SetWidths(array(20,20,20,20,20));

$pocetpreskoku=1;
$pocet=0;
$handle=opendir(ADRESARSOUBORU);
for ($i=0; $file = readdir($handle); $i++)
{
      if (strlen($file)>2)
      {
      //podminka pro datum
            if
(($datum<=substr($file,0,12))and($datumdo>substr($file,0,12)))
            {
                  $xml =
simplexml_load_file(ADRESARSOUBORU.$file);
                  $hodnota[$pocet]=$xml->te0->val;
                  $rok[$pocet]=substr($file,0,4);
                  $mesic[$pocet]=substr($file,4,2);
                  $den[$pocet]=substr($file,6,2);
                  $hodina[$pocet]=substr($file,8,2);
                  $minuta[$pocet]=substr($file,10,2);
                  $pdf->Row(array($den[$pocet],
$mesic[$pocet],$rok[$pocet],$hodina[$pocet].':'.$minuta[$pocet],
$hodnota[$pocet]));
                  $pocet++;
            }
      //konec podminky pro datum
      }
}
$pocetsouboru=$pocet;  //pocet souboru
closedir($handle);
$pdf->Cell(0,10," ","0",1,"C");
$pdf->SetFont('Arial','',8);
$y=$pdf->GetY();
if ($y>180)
{
      $pdf->Cell(80,15,"Graf je na další stránce...","0",1,"C");
      $pdf->AddPage();
      $y=5;
}
$imga=VYSTUPGRAFU;
$info=getimagesize($imga);
$x=5;
$pdf->Image($imga,$x,$y,200,100);
$pdf->Output();
}
?>
```

Příloha 6: Použitelné znakové entity v XML, HTML

&zápis;	vzhled	Český název entity	Anglický název entity
Nbsp		nedělitelná mezera	non-breaking space
iexcl	¡	obrácený vykřičník	inverted exclamation mark
cent	¢	cent	cent sign
pound	£	libra	pound sign
curren	¤	znak měny	currency sign
yen	¥	yen	yen sign (yuan sign)
brvbar	¦	přerušený proužek	broken bar (broken vertical bar)
sect	§	paragraf	section sign
uml	¨		diaeresis (spacing diaeresis)
copy	©	copyright	copyright sign
ordf	ª		feminine ordinal indicator
laquo	«	francouzské uvozovky	left-pointing double angle quotation mark
not	¬	logické not	not sign
shy		rozdělovník = spojovník	soft hyphen (discretionary hyphen)
reg	®	registrovaná značka	registered sign (registered trade mark sign)
macr	¯		macron (spacing macron, overline APL overbar)
deg	°	stupeň	degree sign
plusmn	±	plus mínus	plus-minus sign (plus-or-minus sign)
sup2	²	na druhou	superscript two (superscript digit two, squared)
sup3	³	na třetí	superscript three (superscript digit three, cubed)
acute	´	dlouhé á	acute accent (spacing acute)
micro	µ	mikro	micro sign
para	¶	znak odstavce	pilcrow sign (paragraph sign)
middot	·	středová tečka	middle dot (Georgian comma, Greek middle dot)
cedil	¸		cedilla (spacing cedilla)
sup1	¹		superscript one (superscript digit one)

ordm	º		masculine ordinal indicator
raquo	»	francouzské uvozovky	right-pointing double angle quotation mark (guillemet)
frac14	¼	čtvrtina	vulgar fraction one quarter
frac12	½	polovina	vulgar fraction one half
frac34	¾	tři čtvrtiny	vulgar fraction three quarters
iquest	¿	obrácený otazník	inverted question mark (turned question mark)
Agrave	À		Latin capital letter A with grave
Aacute	Á		Latin capital letter A with acute
Acirc	Â		Latin capital letter A with circumflex
Atilde	Ã		Latin capital letter A with tilde
Auml	Ä		Latin capital letter A with diaeresis
Aring	Å		Latin capital letter A with ring above
AElig	Æ		Latin capital letter AE (Latin capital ligature AE)
Ccedil	Ç		Latin capital letter C with cedilla
Egrave	È		Latin capital letter E with grave
Eacute	É		Latin capital letter E with acute
Ecirc	Ê		Latin capital letter E with circumflex
Euml	Ë		Latin capital letter E with diaeresis
Igrave	Ì		Latin capital letter I with grave
Iacute	Í		Latin capital letter I with acute
Icirc	Î		Latin capital letter I with circumflex
Iuml	Ï		Latin capital letter I with diaeresis
ETH	Ð		Latin capital letter ETH
Ntilde	Ñ		Latin capital letter N with tilde
Ograve	Ò		Latin capital letter O with grave
Oacute	Ó		Latin capital letter O with acute

Ocirc	Ô		Latin capital letter O with circumflex
Otilde	Õ		Latin capital letter O with tilde
Ouml	Ö		Latin capital letter O with diaeresis
times	×	krát	multiplication sign
Oslash	Ø		Latin capital letter O with stroke
Ugrave	Ù		Latin capital letter U with grave
Uacute	Ú		Latin capital letter U with acute
Ucirc	Û		Latin capital letter U with circumflex
Uuml	Ü		Latin capital letter U with diaeresis
Yacute	Ý		Latin capital letter Y with acute
THORN	Þ		Latin capital letter THORN
szlig	ß		Latin small letter sharp s (ess-zed)
agrave	à		Latin small letter a with grave
aacute	á		Latin small letter a with acute
acirc	â		Latin small letter a with circumflex
atilde	ã		Latin small letter a with tilde
auml	ä		Latin small letter a with diaeresis
aring	å		Latin small letter a with ring above
aelig	æ		Latin small letter ae (Latin small ligature ae)
ccedil	ç		Latin small letter c with cedilla
egrave	è		Latin small letter e with grave
eacute	é		Latin small letter e with acute
ecirc	ê		Latin small letter e with circumflex
euml	ë		Latin small letter e with diaeresis
igrave	ì		Latin small letter i with grave
iacute	í		Latin small letter i with

			acute
icirc	î		Latin small letter i with circumflex
iuml	ï		Latin small letter i with diaeresis
eth	ð		Latin small letter eth
ntilde	ñ		Latin small letter n with tilde
ograve	ò		Latin small letter o with grave
oacute	ó		Latin small letter o with acute
ocirc	ô		Latin small letter o with circumflex
otilde	õ		Latin small letter o with tilde
ouml	ö		Latin small letter o with diaeresis
divide	÷	děleno	division sign
oslash	ø		Latin small letter o with stroke
ugrave	ù		Latin small letter u with grave
uacute	ú		Latin small letter u with acute
ucirc	û		Latin small letter u with circumflex
uuml	ü		Latin small letter u with diaeresis
yacute	ý		Latin small letter y with acute
thorn	þ		Latin small letter thorn with
yuml	ÿ		Latin small letter y with diaeresis
fnof	*ƒ*	italské ef	Latin small f with hook (function, florin)
Alpha	A		Greek capital letter alpha
Beta	B		Greek capital letter beta
Gamma	Γ		Greek capital letter gamma
Delta	Δ		Greek capital letter delta
Epsilon	E		Greek capital letter epsilon
Zeta	Z		Greek capital letter zeta
Eta	H		Greek capital letter eta
Theta	Θ		Greek capital letter theta
Iota	I		Greek capital letter iota

Kappa	Κ		Greek capital letter kappa
Lambda	Λ		Greek capital letter lambda
Mu	Μ		Greek capital letter mu
Nu	Ν		Greek capital letter nu
Xi	Ξ		Greek capital letter xi
Omicron	Ο		Greek capital letter omicron
Pi	Π		Greek capital letter pi
Rho	Ρ		Greek capital letter rho
Sigma	Σ		Greek capital letter sigma
Tau	Τ		Greek capital letter tau
Upsilon	Υ		Greek capital letter upsilon
Phi	Φ		Greek capital letter phi
Chi	Χ		Greek capital letter chi
Psi	Ψ		Greek capital letter psi
Omega	Ω		Greek capital letter omega
alpha	α	alfa	Greek small letter alpha
beta	β	beta	Greek small letter beta
gamma	γ	gama	Greek small letter gamma
delta	δ	delta	Greek small letter delta
epsilon	ε		Greek small letter epsilon
zeta	ζ		Greek small letter zeta
eta	η		Greek small letter eta
theta	θ		Greek small letter theta
iota	ι		Greek small letter iota
kappa	κ		Greek small letter kappa
lambda	λ		Greek small letter lambda
mu	μ		Greek small letter mu
nu	ν		Greek small letter nu
xi	ξ		Greek small letter xi
omicron	ο		Greek small letter omicron
pi	π	pí	Greek small letter pi
rho	ρ		Greek small letter rho
sigmaf	ς		Greek small letter final sigma
sigma	σ		Greek small letter sigma
tau	τ		Greek small letter tau
upsilon	υ		Greek small letter upsilon
phi	φ		Greek small letter phi
chi	χ		Greek small letter chi

psi	ψ		Greek small letter psi
omega	ω		Greek small letter omega
bull	•	puntík	bullet (black small circle)
hellip	…	tři tečky	horizontal ellipsis (three dot leader)
prime	′	minuta, stopa	prime (minutes, feet)
Prime	″	vteřina, palec	double prime (seconds, inches)
oline	‾	nadtržítko	overline (spacing overscore)
frasl	⁄	šikmé lomítko	fraction slash
trade	™	obchodní značka	trade mark sign
larr	←	šipka vlevo	leftwards arrow
uarr	↑	šipka nahoru	upwards arrow
rarr	→	šipka doleva	rightwards arrow
darr	↓	šipka dolů	downwards arrow
harr	↔	pravo-levá šipka	left right arrow
part	∂	parciální derivace	partial differential
prod	∏	celkový součin	n-ary product (product sign)
sum	∑	celkový součet	n-ary summation
minus	−	mínus	minus sign
radic	√	odmocnina	square root (radical sign)
infin	∞	nekonečno	infinity
cap	∩	průnik	intersection (cap)
int	∫	integrál	integral
asymp	≈	anglické téměř rovno	almost equal to (asymptotic to)
ne	≠	nerovnost	not equal to
equiv	≡	identita	identical to
le	≤	menší nebo rovno	less-than or equal to
ge	≥	větší nebo rovno	greater-than or equal to
loz	◊	kosočtverec	lozenge
spades	♠	piky	black spade suit
clubs	♣	kříže	black club suit (shamrock)
hearts	♥	srdce	black heart suit (valentine)
diams	♦	kára	black diamond suit
quot	"	rovné uvozovky	quotation mark (APL quote)
amp	&	et	ampersand
lt	<	menší	less-than sign

gt	>	větší	greater-than sign
OElig	Œ		Latin capital ligature OE
oelig	œ		Latin small ligature oe
Scaron	Š		Latin capital letter S with caron
scaron	š		Latin small letter s with caron
Yuml	Ÿ		Latin capital letter Y with diaeresis
circ	ˆ	programátorská mocnina	modifier letter circumflex accent
tilde	˜	vlnka, tilda	small tilde
zwnj		bezrozměrné svislice	zero width non-joiner
zwj		bezrozměrné dělítko	zero width joiner
lrm		zleva do prava (typ.)	left-to-right mark
rlm		zprava doleva (typ.)	right-to-left mark
ndash	–	menší pomlčka	en dash
mdash	—	pomlčka	em dash
lsquo	‘		left single quotation mark
rsquo	’		right single quotation mark
sbquo	‚		single low-9 quotation mark
ldquo	“	levé horní uvozovky	left double quotation mark
rdquo	”	horní pravé uvozovky	right double quotation mark
bdquo	„	levé dolní uvozovky	double low-9 quotation mark
dagger	†	křížek	dagger
Dagger	‡	dvojkřížek	double dagger
permil	‰	promile	per mille sign
lsaquo	‹		single left-pointing angle quotation mark
rsaquo	›		single right-pointing angle quotation mark
euro	€	euro	euro sign

Příloha 7: Obsah CD a internetové stránky

Hlavní adresáře na CD:

/wampserver/ - instalační program WampServeru 1.6.6 pro Windows

/xmltemp/ - skripty pro převod XML souborů do PDF – 1. praktická úloha

/xmltemp/teplota/ - XML soubory, zdroj dat

/skriptteplota/ - skript pro čtení XML dokumentu a ukládání spouštěný např. cronem

/Makefont/ - tvorba písma s českými znaky (Kapitola 4.7)

/mathml/ - skripty pro převod MathML do PDF – 2. praktická úloha

/mathml/examples/ - příklady XML souborů v MathML

/diplomka/ - diplomová práce v PDF

Internetové adresy:

Úloha měření teploty:

http://www.obsah.info/xmltemp

Převod MathML do PDF:

http://www.obsah.info/mathml

http://www.obsah.info/mathml/examples