Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

DIPLOMOVÁ PRÁCE

Plzeň, 2006 Jan Kupka

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

PVS - Portál veřejné správy a jeho využití

Plzeň, 2006 Jan Kupka

Abstract

Target of this diploma thesis is to research Portal of the Public Administration of the Czech republic and develop an application, which allows users to send transmissions to the Portal of the Public Administration. These transmissions contains forms for the public administration. The diploma thesis is divided into theoretical and practical section. Theoretical section describes technologies used for commnunication with Portal of the Public Administration. Practical part describes Portal of the Public Administration, its usage in an accounting system and implementation of an application.

Obsah

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 22.5.2006, Jan Kupka

Poděkování

Rád bych touto cestou poděkoval panu Josefu Weinrebovi za vedení diplomové práce. Dále chci poděkovat Martině Balíkové z ČSSZ za poskytnutí údajů pro přístup k testovací části PVS. V neposlední řadě chci poděkovat všem administrátorům PVS, kteří mi poskytli cenné informace týkající se funkcí PVS.

Kapitola 1

Úvod

Tato diplomová práce pojednává o Portálu veřejné správy. PVS je projektem Ministerstva financí, který by měl zastřešovat veškerou komunikaci občanů, firem a institucí se státní správou. Prezentační část portálu se nachází na webové adrese http://portal.gov.cz. Úkolem PVS je poskytovat informace lidem a firmám v České republice a umožnit jim jednodušší komunikaci se státní správou než tomu bylo možné doposud. Občanům by měl poskytovat veškeré informace pro řešení běžných životních situací. Firmám by měl naopak zpřístupnit informace o zákonech a možnost komunikovat s orgány státní správy elektronickou cestou, tedy například podávat elektronická daňová přiznání.

A právě firemní komunikace s Portálem veřejné správy je pilířem této diplomové práce. V terminologii PVS se tato komunikace označuje jako elektronická podání a je vystavena na třech pilířích, kterými jsou moderní technologie XML, komunikace pomocí webovských technologií a počítačová bezpečnost.

XML je skutečný fenomén posledních let a stále více služeb je na tomto formátu postaveno. Komunikace s PVS je založena na moderních technologiích zaručujících bezpečný přenos dat po internetu. Samozřejmě nesmí chybět ani počítačová bezpečnost, disciplína, která je v neustálem vývoji a stává se pro současný počítačový svět stále potřebnější. Tyto moderní technologie budou detailně popsány v následujících kapitolách a budeme z nich vycházet pro popis komunikace s Portálem veřejné správy za účelem elektronických podání.

Stěžejní částí této diplomové práce je program pro komunikaci s Portálem veřejné správy. Tento program je vytvářen jako můstek pro přenos dat mezi účetním systémem (WinStrom či libovolná jiná aplikace) a Portálem veřejné správy.

1.1 Cíl diplomové práce

Hlavním cílem této diplomové práce je vytvoření aplikace, která bude sloužit pro přenos dat z účetního systému (WinStrom či libovolný jiný účetní systém) na Portál veřejné správy.

Pro realizaci tohoto cíle je nutné seznámit se s technologiemi potřebnými pro komunikaci s PVS. Popis těchto technologií bude součástí teoretické části diplomové práce.

Praktickou částí diplomové práce bude detailní popis informační a transakční částí PVS a možností využití PVS v ekonomickém systému. Součástí praktické části bude i porovnání konkurenčních aplikací a samozřejmě návrh a implementace vlastní aplikace.

Kapitola 2

XML

Tato kapitola popisuje jazyk XML a možnosti jeho využití. Jde o klíčovou technologii pro práci s Portálem veřejné správy, proto jí bude věnován dostatek prostoru. Na druhou stranu tato kapitola nebude suplovat referenční příručku jazyka XML, ale půjde pouze o úvod do technologií, které se používají v souvislosti s jazykem XML a je nezbytně nutné se o nich zmínit. Podrobný přehled této problematiky se nachází například na stránkách Jiřího Koska [5].

2.1 Úvod do XML

Jazyk XML definovala roku 1996 pracovní skupina W3C jako podmnožinu jazyka SGML, který je základem všech značkovacích jazyků.

Pojem značkovací jazyk určuje, že se daný formát zabývá logickou strukturou dokumentu. Jazyk XML je „příbuzný“ s jazykem HTML, který je taktéž značkovacím jazykem, ale narozdíl od XML klade důraz na fyzické vyznačování, tedy jak budou data zobrazena. XML nám způsob svého zobrazení nedefinuje, určuje pouze logické uspořádání a na fyzické vyznačování slouží jiné mechanismy, které budou popsány dále.

Jazyk XML byl od počátku vytvářen tak, aby mohl být užíván různými národy s ohledem na jejich jazykové odlišnosti. Vždyť samotným požadavkem, na základě kterého byl jazyk XML vytvořen, byla konstrukce „jazyka SGML pro web“. A web, to je ryze multilinguální prostředí.

Jazyk XML je považován za otevřený a přizpůsobivý jazyk. Otevřený hlavně z toho důvodu, že byl vytvořen mezinárodním konsorciem W3C, které je nezávislé na jakýchkoliv

komerčních aktivitách. Z tohoto důvodu je XML přístupné každému, kdo o něj projeví zájem, zcela zdarma. Jeho přizpůsobivost tkví v tom, že je možné jazyk XML libovolně transformovat do vlastního značkovacího jazyka.

Jazyk XML je vytvořen tak, aby byl snadno čitelný pro člověka i pro počítač. Z tohoto důvodu je formát XML souboru čistě textový a uživatel většinou nemá problém se v XML souboru orientovat. Stejně tak si mohou XML soubory mezi sebou vyměňovat počítačové programy, neboť XML má přesně definovaná pravidla, která je nutno dodržovat. Využití takových pravidel bude popsáno v následujících kapitolách.

Každá mince má svůj rub a líc. Podívejme se tedy na nevýhody XML. Jazyk XML je díky tomu, aby byl snadno čitelný pro lidi i pro stroje, redundantní, což se může projevovat například pomalejším přenosem XML souborů po síti. Dále jazyk XML potřebuje vytvoření složitějšího parseru (programu pro čtení dat z XML dokumentu), což je dosti nepříjemné například na přenosných zařízeních s malou kapacitou paměti. Za nevýhody může být považováno i hierarchické uspořádání dat a také to, že data nejsou definována datovými typy.

2.2 Specifikace

Jazyk XML byl vytvořen tak, aby splňoval 10 základních požadavků, které si stanovili členové skupiny W3C.

Požadavky skupiny W3C na jazyk XML

1. Dokumenty XML by mělý být přímo použitelné na internetu
2. XML by mělo podporovat širokou paletu aplikací
3. Jazyk XML by měl být kompatibilní s jazykem SGML
4. Tvorba programů zpracovávajících XML by měla být snadná
5. Počet volitelných částí XML by měl být naprosto minimalizován, v ideálním případě by měl být nulový
6. Dokumenty XML by měly být pro člověka snadno čitelné, dostatečně jasné
7. Návrh XML by měl být rychle hotov

8. Návrh XML by měl být metodický a stručný
9. Vytváření dokumentů XML by mělo být snadné
10. Obsažnost značkování XML není příliš důležitá

2.3 Použití XML

Před několika lety byla jazyku XML předpovídána veliká budoucnost. Dnes už víme, že se proroctví vyplnila.

Příklady použití jazyka XML

- **SMIL (Synchronized Multimedia Integration Language)**
  - popis multimediálních prezentací
- **SOAP (Simple Object Access Protocol)**
  - webové služby
- **MathML (Mathematical Markup Language)**
  - formátování matematických vzorců a vědeckých výpočtů na webu
- **AML (Astronomical Markup Language)**
  - výměna astronomických dat
- **XHTML (Extensible Hypertext Markup Language)**
  - tvorba webových stránek
- **SVG (Scalable Vector Graphics)**
  - vektorová grafika
- **WML (Wireless Markup Language)**
  - „webové stránky“ pro mobilní zařízení
- **RSS (Really Simple Syndication)**
  - publikování seznamu odkazů na články

2.4 XML dokument

XML dokument je každý takový dokument, který je správně strukturovaný (well-formed) podle specifikace W3C [25]. XML soubory mohou být vytvářeny buď programově nebo ručně například v programu Poznámkový blok. V praxi je samozřejmě častější první způsob. Podívejme se na příklad takto vytvořeného souboru:

```
<?xml version="1.0" encoding="UTF-8"?>
<zakaznik id="2000">
  <jmeno>
    <krestni>Pavel</krestni>
    <prijmeni>Novák</prijmeni>
    <adresa />
  </jmeno>
</zakaznik>
```

Zásady well-formed (správně strukturovaného) XML dokumentu

- Dokument musí obsahovat pouze jeden element nejvyšší úrovně (kořenový element).
- Všechny ostatní elementy jsou zanořeny v něm.
- Všechny elementy musí být správně zanořeny (nesmějí se křížit). Pokud element začíná uvnitř jiného elementu, musí v něm také končit.
- Každý element musí obsahovat jak počáteční, tak ukončovací značku. Toto nemusí platit, pokud použijeme tzv. prázdnou značku.
- Název elementu v počáteční značce musí být totožný s názvem v ukončovací značce.
- V názvech elementů jsou rozlišována malá a velká písmena.
- Názvy elementů nesmějí obsahovat mezeru.
- Mezi počáteční a koncovou značkou se nesmí používat znaky < a &, ale nedoporučuje se používat ani znak >.

- Hodnota atributů musí být vždy uvedena v uvozovkách (je možné používat i jednoduché uvozovky, ale je lepší vše značit dvojitými, pokud nemáme nějaký speciální důvod pro použití jednoduchých).
- XML dokument se skládá z prologu, kořenového elementu a zanořených elementů.

2.4.1 Prolog

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

Na prvním řádku XML dokumentu se nachází tzv. prolog. Jeho první částí je deklarace dokumentu označující, že se jedná o dokument ve formátu XML. Jako číslo verze se v době psaní této práce používalo označení 1.0. Následně je uvedeno kódování dokumentu, které je ovšem nepovinné. V případě neuvedení typu kódování počítá parser s XML souborem ve formátu UTF-8. Pokud je potřeba nastavit jiné kódování, musí to být implicitně deklarováno atributem encoding v prologu dokumentu. Je možné zde například nastavit kódování windows-1250. V případě použití jiného kódování je potřeba otestovat parser, zda dokument v tomto formátu umí zpracovat, neboť dle specifikace je po parseru vyžadováno jen zpracování dokumentů ve formátu UTF-8. Poslední atribut standalone obsahuje hodnoty yes nebo no, přičemž `standalone="yes"` znamená, že soubor je nezávislý na případném DTD, zatímco `standalone="no"` deklaruje, že obsah XML souboru závisí na DTD (popis DTD bude uveden dále).

2.4.2 Kořenový element a zanořené elementy

Druhou částí XML dokumentu je kořenový element, který se skládá z dalších zanořených elementů. Kořenový element je základním elementem XML dokumentu a musí být vždy přítomen. Jde o standardní element, který zakládá hierarchickou strukturu XML dokumentu. Elementy zapisujeme pomocí XML značek (neboli tagů). Většina elementů obsahuje počáteční a koncovou značku. Existují i elementy, jejichž počáteční značka je zároveň koncovou značkou. Tyto elementy označujeme jako prázdné. Názvy značek (tagů) zapisujeme mezi znaky < a >, které vymezují značku. Názvy značek jsou narozdíl od jazyka HTML volitelné.

Počáteční značka je tvořena názvem elementu, který je uzavřen mezi znaky < a >. Koncová značka je uzavřena také mezi znaky < a >, navíc je před názvem elementu znak /.

Mezi obě značky je vložen obsah elementu.

2.4.3 Prázdné elementy

Obsah elementu mezi značkami (tagy) může být také prázdný, např. `<adresa></adresa>`. Přednostně se ale doporučuje prázdný element zapisovat tímto způsobem: `<adresa />`.

2.4.4 Atributy

Do počáteční značky elementu je možné vložit jeden nebo více atributů, které jsou tvořeny dvojicí název-hodnota a logicky souvisí s elementem. U atributů nezáleží na pořadí v jakém jsou uvedeny. Hodnota atributu musí být uvedena v jednoduchých nebo dvojitých uvozovkách a je přiřazena jménu atributu znakem =.
Příklad elementu s atributy: `<zakaznik id="2000"></zakaznik>`.

2.4.5 Jmenné prostory

Jazyk XML přináší téměř absolutní svobodu při tvorbě elementů XML dokumentu. Tato výhoda však zároveň představuje problém, pokud existuje několik tvůrců XML dokumentů a ti chtějí své dokumenty spojit dohromady. V takovém případě může dojít k situaci, že pojmenovali elementy stejnými názvy a ty zatím slouží k jinému účelu. Za tohoto stavu můžeme buď dokumenty přepsat, aby nedocházelo ke kolizím názvů, nebo použít již připravené řešení označované jako jmenné prostory. Nejde o žádnou novinku, neboť jmenné prostory se běžně používají v programovacích jazycích. Autoři specifikace XML se rozhodli použít jako jedinečný identifikátor jmenného prostoru název domény autora dokumentu. Dle této specifikace jeden z autorů použije například jmenný prostor http://honza.cz a druhý jmenný prostor http://petr.cz. Každý autor může používat více jmenných prostorů či jiné verze jmenných prostorů. Z tohoto důvodu se specifikovalo následující použití jmenného prostoru - jmenný prostor začíná doménou autora, následují volitelné informace, za nimi se nachází popis daného jmenného prostoru a poslední je volitelné číslo verze. Příklad adresy jmenného prostoru, který odpovídá výše uvedené specifikaci: http://honza.cz/namespaces/basic/1.0. Přestože má označení formu URI, nejedná se o klasické URI, ale pouze o řetězec odlišující jmenné prostory.

Implicitní jmenný prostor

Jmenný prostor je možné deklarovat v kterémkoliv elementu XML a následně se bude vztahovat na všechny potomky daného elementu. Implicitní jmenný prostor deklarujeme v počáteční značce zvoleného elementu (jde vlastně o atribut). Tento atribut má název xmlns a jeho hodnotou je právě název jmenného prostoru. Často se používá deklarace implicitního jmenného prostoru v kořenovém elementu, z čehož vyplývá, že všechny elementy dokumentu pochází ze jmenného prostoru uvedeného v kořenovém elementu.

Jmenné prostory jednotlivých elementů

Jmenné prostory mohou být deklarovány i tak, aby platily jen v jednotlivých vyjmenovaných elementech. V takovém případě pojmenujeme atribut jako xmlns:prefix (část prefix si libovolně zvolíme). Umístěním prefixu před název elementu (a může se jednat i o element, ve kterém je prefix deklarován) začleníme tento element do uvedeného jmenného prostoru.

Příklad použití jmenného prostoru:

```
<hns:zakaznik id="2000" xmlns:hns="http://honza.cz/namespaces/basic/1.0">
  <hns:jmeno>
    <hns:krestni>Pavel</hns:krestni>
    <hns:prijmeni>Novák</hns:prijmeni>
    <hns:adresa />
  </hns:jmeno>
</hns:zakaznik>
```

2.4.6 Komentáře

Komentáře v XML zapisujeme ve tvaru <!--komentář--!>. Nesmějí být do sebe zanořovány a také nesmějí být součástí značky.

2.4.7 CDATA

Do sekce CDATA vkládáme jakékololiv znaky kromě řetězce]] >. Můžeme zde tedy používat jinak zakázané znaky jako < či &. Sekci CDATA zapisujeme následujícím způsobem: <![CDATA[obsah]] >. Parser nebude obsah interpretovat, ale nechá ho tak, jak je. Často

se toho využívá například při výpisu zdrojových kódů či kódů HTML/XML. Sekce CDATA může být použita v obsahu elementu a nemůže být zanořována.

Příklad neinterpretovaného HTML kódu:

```
<![CDATA[
<html>
  <head>
  </head>
  <body>
    <p>Vítejte</p>
  </body>
</html>
]]>
```

2.5 Definice typu dokumentu

V předcházejících kapitolách byl naznačen způsob tvorby správně strukturovaných XML dokumentů. Z důvodu dalších možností využití XML dokumentů budeme muset navíc přesně definovat strukturu dokumentů, aby aplikace mohly ověřit, zda je dokument validní.

Zde bych chtěl upozornit na to, že jsou často zaměňovány pojmy well-formed (dobře strukturovaný) a valid (validní) XML dokument. Pokud mluvíme o dobře strukturovaném dokumentu, máme na mysli, že dodržuje základní pravidla pro tvorbu XML dokumentu, tedy např. zanořování elementů (viz dříve). Naproti tomu validní dokument je definován pomocí šablony a určuje, jaké elementy se budou na jakém místě v XML dokumentu nacházet a další vlastnosti.

K čemu tedy používat šablonu dokumentu? XML nám díky svému otevřenému formátu dává možnost definice vlastních značek. To je samozřejmě velká výhoda, ale zároveň to může být i nevýhoda při výměně dokumentů. Pokud například chceme odeslat fakturu jiné firmě, měli bychom dodržet určitou předepsanou strukturu pro tvorbu faktury. A tuto strukturu právě definuje šablona (schéma) dokumentu. Díky tomu, že například na svých webových stránkách umístíme šablonu námi přijímané faktury, si mohou zákazníci jednoduše ověřit, zda mají XML dokument validní a následně nám ho odeslat. Příjemce faktury naopak může použít schéma pro validaci XML dokumentu a už se nemusí zabývat

kontrolou chyb v dokumentu jako tomu bylo například u čistě textových souborů (např. CSV).

K ověřování správnosti XML dokumentů se používají validátory, které mohou být součástí programů nebo může jít o samostatné parsery s možností validace jako je například Xerces (DTD, XSD) či Jing (RELAX NG). Použití šablon přináší ještě jednu výhodu. Při tvorbě XML dokumentu s pomocí některého pokročilého editoru (například Emacs či XXE) může editor načíst schéma dokumentu a nabízet uživateli elementy XML, které vyhovují danému schématu.

V současnosti se můžeme setkat nejčastěji s těmito schémovými jazyky (jazyky pro definici dokumentů):

- **DTD**
  Jde o základní formát definice XML dokumentu vycházející ze SGML. Neobsahuje podporu pro definici jmenných prostorů a nemá ani podporu definice datových typů.
- **XSD**
  V současnosti jde o nejrozšířenější schémový jazyk, který je podporován velkými počítačovými firmami. Obsahuje podporu pro definici jmenných prostorů a datových typů. Jeho nevýhodou je složitá specifikace a značná „ukecanost“.
- **RELAX NG**
  Je velmi podobný jazyku XSD, odstraňuje jeho složitost a „ukecanost“. Bohužel ale není podporován velkými společnostmi, neboť nepřináší příliš mnoho novinek vyjma jednoduššího zápisu.
- **SCHEMATRON**
  Jedná se o jiný přístup než výše uvedené. Jazyk se skládá ze sady XPATH příkazů, které umožňují definovat relace mezi jednotlivými elementy. Často se používá jako doplněk výše uvedených jazyků.

2.6 DTD

DTD (Document Type Definition) je nejstarším schémovým jazykem. Bylo uvedeno společně s jazykem XML. DTD můžeme umístit jak uvnitř XML dokumentu samotného (interní definice DTD), tak mimo něj do zvláštního souboru (externí DTD). Pokud DTD

umístíme uvnitř XML dokumentu, může být použito právě jen s tímto dokumentem, což je hlavní nevýhodou tohoto způsobu. Externí DTD můžeme naopak připojit k jakémukoliv XML dokumentu.

Příklad externího DTD:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Osoba [
<!ELEMENT Osoba (Jmeno,Adresa,Datum_narozeni) >
<!ATTLIST Osoba id CDATA #REQUIRED >
<!ATTLIST Osoba stav CDATA #REQUIRED >
<!ELEMENT Jmeno (#PCDATA) >
<!ELEMENT Adresa (#PCDATA) >
<!ELEMENT Datum_narozeni (#PCDATA) >
]>
```

Příklad XML validního dle výše uvedeného DTD:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Osoba id="2006" stav="svobodny">
  <Jmeno>Pavel Novák</Jmeno>
  <Adresa />
  <Datum_narozeni>17.01.1982</Datum_narozeni>
</Osoba>
```

DTD obsahuje čtyři typy deklarací, z čehož poslední dvě se vyskytují velmi zřídka:

- deklarace elementů - $<!ELEMENT>$
- deklarace atributů - $<!ATTLIST>$
- deklarace entit - $<!ENTITY>$
- deklarace notací - $<!NOTATION>$

2.6.1 Deklarace elementů

Pořadí deklarace elementů nemusí odpovídat jejich uspořádání v XML dokumentu, je však dobré ho dodržovat. Element XML dokumentu deklarujeme pomocí DTD následujícím způsobem:

```
<!ELEMENT nazev_elementu obsah_elementu>
```

- **Prázdný element - EMPTY**

  ```
  DTD: <!ELEMENT Adresa EMPTY>
  XML: <Adresa />
  ```

- **Libovolný typ elementu - ANY**

  ```
  DTD: <!ELEMENT Adresa ANY>
  XML: <Adresa>Univerzitní 22</Adresa> nebo <Adresa />
  ```

- **Modelová skupina**
  - deklaruje strukturu XML dokumentů (nadřazený element a jeho vztah k zanořeným elementům)

  Pořadí elementů

  ```
  DTD: <!ELEMENT Osoba (Jmeno,Adresa,Datum_narozeni)>
  XML:
  <Osoba>
    <Jmeno/>
    <Adresa />
  </Osoba>
  ```

  Výběr jednoho z předdefinovaných elementů

  ```
  DTD: <!ELEMENT Osoba (Jmeno|Prezdivka) >
  XML:
  <Osoba>
    <Jmeno/>
  </Osoba>
  ```

- **Elementy obsahující text**

  ```
  DTD: <!ELEMENT Jmeno (#PCDATA) >
  XML: <Jmeno>Jan Kupka</Jmeno>
  ```

- **Elementy se smíšeným obsahem**
  - elementy můžeme libovolně kombinovat, abychom dosáhli požadované struktury XML dokumentu

  ```
  <!ELEMENT smiseny (#PCDATA | pozdrav | rozlouceni)* >
  ```

2.6.2 Deklarace atributů

Deklaraci atributů zapisujeme následujícím způsobem:

```
<!ATTLIST název_elementu deklarace_atributů>
```

- **Atribut CDATA**
  - hodnotou atributu je jednoduchý řetězec znaků

  ```
  DTD: <!ATTLIST Osoba poznamka CDATA>
  XML: <Osoba poznamka="velmi dulezita osoba"></Osoba>
  ```

- **Atribut NMTOKEN**
  - hodnotou atributu je jedno slovo

  ```
  DTD: <!ATTLIST Osoba poznamka NMTOKEN>
  XML: <Osoba poznamka="VIP"></Osoba>
  ```

- **Atribut NMTOKENS**
  - hodnotou je několik slov oddělených mezerami

  ```
  DTD: <!ATTLIST Osoba poznamka NMTOKENS>
  XML: <Osoba poznamka="VIP pristup_na_raut"></Osoba>
  ```

- **Atribut ID**
  - atribut typu ID se používá pro identifikátory
  - hodnota atributu typu ID musí být jedinečná mezi všemi ostatními hodnotami atributů tohoto typu v rámci dokumentu (i v případech, kdy se jedná o atributy s různým názvem u různých elementů)
  - hodnota ID musí začínat písmenem

```
DTD: <!ATTLIST Osoba identifikator ID>
XML: <Osoba identifikator="A01165"></Osoba>
```

- **Výčet hodnot**
  - výběr z několika možných hodnot atributu

```
DTD: <!ATTLIST Osoba stav (svobodny | zenaty | rozvedeny ) >
XML: <Osoba stav="svobodny"></Osoba>
```

Povinnost výskytu atributu

Za typ atributu můžeme ještě uvést, zda se jedná o povinný či nepovinný atribut.

- **Povinný atribut**
  - hodnota nastavená při neuvedení povinnosti atributu

```
<!ATTLIST Osoba identifikator ID #REQUIRED>
```

- **Volitelný atribut**

```
<!ATTLIST Osoba poznamka NMTOKENS #IMPLIED>
```

- **Implicitní hodnota atributu**
  - používá se u výčtů, kdy jednu hodnotu předvolíme pro případ, že by atribut nebyl vyplněn

```
<!ATTLIST Osoba stav (svobodny | zenaty | rozvedeny ) "svobodny">
```

Jestliže v DTD za typem atributu uvedeme klíčové slovo `#FIXED` následované hodnotou v uvozovkách, atribut bude mít právě tuto pevně stanovenou hodnotu.

2.6.3 Entity

Entity umožňují rozdělení jednoho dokumentu do více souborů a také mohou ušetřit práci při častém opakování textu v dokumentu. Entity můžeme rozlišovat podle jejich vlastností na obecné a parametrické, externí či interní a textové či binární. Podrobnější popis entit se nachází v knize [2].

- **Příklad entity pro připojení externího XML dokumentu**

  ```
  <!ENTITY narodnost SYSTEM "narodnost.xml">
  ```

  Volání entity: &narodnost

- **Příklad entity pro časté opakování textu**

  ```
  <!ENTITY asap "As soon as possible">
  ```

  Volání entity: &asap

2.6.4 Notace

Notace představuje způsob, jak identifikovat formát či způsob zpracování dat, která nejsou určena XML procesoru.

- **Příklad notace**

  ```
  <!NOTATION gif SYSTEM "image/gif" >
  ```

2.7 XSD

DTD sice bylo prvním stylovým jazykem pro XML, ale zdaleka není dokonalé. Často je mu vytýkáno, že není napsáno v jazyce XML. DTD také nepopisuje datové typy, rozsahy či omezení, ale pouze strukturu XML dokumentu. V neposlední řadě DTD nepodporuje jmenné prostory. Z těchto a mnoha dalších důvodů bylo nutné přijít s novým prostředkem pro definici XML dokumentů. A tak se roku 2001 zrodilo XSD (eXtensible Stylesheet Definition). Tento stylový jazyk brzy podpořily významné firmy jako Sun či Microsoft, a tak se stal nejpoužívanějším jazykem pro definici XML dokumentů.

V případě XSD mluvíme o tzv. schématech, což je ovšem pouze jiný název pro definici dokumentu u DTD. Schéma a dokument XML, který popisuje, jsou uloženy v samostatných souborech. Schéma je XML dokument, který je napsán v souladu se specifikací W3C. Jde tedy o zvláštní aplikaci XML (proto používáme písmena xsd jako prefix pro jmenný prostor schématu).

Příklad XSD dokumentu:

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Osoba">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Jmeno" type="xsd:string" />
        <xsd:element name="Adresa" type="xsd:string" />
        <xsd:element name="Datum_narozeni" type="xsd:date" />
     </xsd:sequence>
     <xsd:attribute name="id" type="xsd:positiveInteger" use="required"/>
     <xsd:attribute name="stav" type="xsd:string" use="required"/>
   </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Příklad XML validního dle výše uvedeného XSD:

```
<?xml version="1.0" encoding="UTF-8" ?>
<Osoba id="2006" stav="svobodny">
  <Jmeno>Pavel Novák</Jmeno>
  <Adresa />
  <Datum_narozeni>17.01.1982</Datum_narozeni>
</Osoba>
```

2.7.1 Deklarace elementů

Element deklarujeme pomocí schématu:
`<xsd:element name="nazev" type="xsd:typ"></xsd:element>`.

`XSD: <xsd:element name="Osoba"></xsd:element>`
`XML: <Osoba></Osoba>`

Deklarace elementu pomocí předdefinovaného jednoduchého typu

Pro deklaraci elementu se používá jeden z předdefinových datových typů jmenného prostoru xsd. Počet výskytů elementu je možné nastavit pomocí atributů minOccurs (minimální

počet výskytů elementu) a maxOccurs (maximální počet výskytů elementu). Implicitně jsou nastaveny na hodnotu jedna. Libovolný počet výskytů je možné nastavit atributem `maxOccurs="unbounded"`.

```
XSD: <xsd:element name="vybrano" type="xsd:boolean">
XML: <vybrano>true</vybrano>
```

```
XSD: <xsd:element name="datum" type="xsd:date">
XML: <datum>2006-01-01</datum>
```

```
XSD: <xsd:element name="zaporne_cislo" type="negativeInteger">
XML: <zaporne_cislo>-1024</zaporne_cislo>
```

Deklarace elementu pomocí definice jednoduchého typu

Z jednoduchých datových typů můžeme vytvořit odvozený datový typ soustavou omezujících pravidel.

```
<xsd:element name="vek">
  <xsd:simpleType>
    <xsd:restriction base="xsd:positiveInteger">
      <xsd:minExclusive value="0" />
      <xsd:maxExclusive value="150" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd>
```

Datový typ může být omezen i výčtem řetězců.

```
<xsd:element name="pozice">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="student" />
      <xsd:enumeration value="ucitel" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd>
```

Datový typ string může omezit regulární výraz.

```
<xsd:element name="telefon">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="+420\d{9}" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd>
```

Odvozené datové typy mohou být anonymní či jmenné. Anonymní datové typy se vztahují k jednomu konkrétnímu elementu. Jsou to všechny výše definované datové typy. Jmenný datový typ pro telefonní číslo bude vypadat takto:

```
<xsd:simpleType name="telefonni_cislo">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="+420\d{9}" />
  </xsd:restriction>
</xsd:simpleType>
```

Tento typ bude uveden přímo v kořenovém elementu xsd:scheme. A pokud ho budeme chtít použít, provedeme to následujícím schématem:

```
<xsd:element name="telefon" type="telefonni_cislo" />
```

Deklarace elementu komplexního typu

Pro deklaraci elementu komplexního typu lze použít předdefinovaný typ `xsd:anyType`. Jedná se o základní typ elementu, který je implicitně nastaven, pokud není uveden atribut type. Tento element může obsahovat libovolná data. Může také obsahovat libovolný element nebo atribut, který je deklarován v kořenovém elementu `xsd:schema`.

Komplexní typ definujeme pomocí elementu schématu `xsd:complexType`. Tento typ můžeme použít k deklaraci elementu s obsahem element (pouze elementy potomků), se smíšeným obsahem (elementy potomka a znaková data) nebo s prázdným obsahem (žádný element potomka ani znaková data).

Použití schématu sequence určuje přesné pořadí jednotlivých elementů.

```
<xsd:element name="adresa">
  <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="ulice" type="xsd:string" />
     <xsd:element name="orientacni_cislo" type="xsd:positiveInteger" />
     <xsd:element name="psc" type="xsd:positiveInteger" />
     <xsd:element name="mesto" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Použití schématu choice definuje možnost výběru pouze jednoho elementu.

```
<xsd:element name="jmeno">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="prezdivka" type="xsd:string" />
      <xsd:element name="jmeno" type="xsd:string" />
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Použití schématu all definuje možnost libovolného uspořádání elementů. Počet výskytů jednotlivých elementů nastavíme pomocí atributů minOccurs a maxOccurs.

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="jmeno" type="xsd:string" />
      <xsd:element name="adresa" type="xsd:string" maxOccurs="2" />
      <xsd:element name="datumNarozeni" type="xsd:date" minOccurs="0" />
     </xsd:all>
  </xsd:complexType>
</xsd:element>
```

Pomocí těchto základních schémat je možno vytvářet komplexnější celky.

2.7.2 Deklarace elementu se smíšeným obsahem

Element se smíšeným obsahem může obsahovat vnořené elementy i znaková data.

```
<xsd:element name="jmeno">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="prezdivka" type="xsd:string" />
     </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Deklarace elementu s prázdným obsahem

Element, který neobsahuje žádného potomka ani žádná data deklarujeme takto:

```
<xsd:element name="prazdny">
  <xsd:complexType>
  </xsd:complexType>
</xsd:element>
```

2.7.3 Deklarace atributů

Atribut deklarujeme pomocí elementu schématu `xsd:attribute`. Atribut má vždy jednoduchý datový typ. Stejně jako u deklarace elementu můžeme použít buď předdefinovaný typ či definovat nový jednoduchý typ.

Předdefinovaný typ atributu

```
<xsd:attribute name="identifikator" type="xsd:positiveInteger"/>
```

Odvození z předdefinovaného typu atributu

```
<xsd:attribute name="stav">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="svobodny/a" />
```

```
      <xsd:enumeration value="zenaty/vdana" />
     <xsd:enumeration value="rozvedeny/a" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

Výskyt atributu určuje atribut schématu use. Pokud atribut use nepoužijeme či jako jeho hodnotu uvedeme výraz optional, nemusí být atribut použit. Pokud nastavíme hodnotu na required, musí být atribut vložen. Pokud přiřadíme hodnotu prohibited, atribut nemůže být vložen.

```
<xsd:attribute name="identifikator" type="xsd:positiveInteger"
use="required" />
```

Atributu můžeme přednastavit výchozí hodnotu, která se použije v případě, že atribut nebude nastaven.

```
<xsd:attribute name="stav" type="xsd:string" default="svobodny"/>
```

Alternativou k atributu default je atribut fixed. V případě vynechání argumentu s nastavením fixed, je doplněna hodnota atributu přiřazená jako fixed. Pokud je atribut vložen, musí mít nastavenu hodnotu přiřazenou jako fixed.

Vkládání atributů do elementu s obsahem element, smíšeným nebo prázdným obsahem

Do elementu s obsahem element, smíšeným nebo prázdným obsahem vložíme atribut přidáním deklarace do elementu `xsd:complexType`. Deklaraci atributu musíme vložit až po elementech `xsd:sequence`, `xsd:choice` nebo `xsd:all`.

```
<xsd:element name="osoba">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="jmeno" type="xsd:string" />
      <xsd:element name="adresa" type="xsd:string" maxOccurs="2" />
      <xsd:element name="datumNarozeni" type="xsd:date" minOccurs="0" />
     </xsd:all>
```

```
    <xsd:attribute name="identifikator" type="xsd:positiveInteger"/>
  </xsd:complexType>
</xsd:element>
```

Vkládání atributů do elementu obsahujícího pouze znaková data

```
<xsd:element name="datum">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:sdate" />
        <xsd:attribute name="poznamka" type="xsd:string" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

2.8 XPATH

Před stylováním XML dokumentu pomocí technologie XSLT je zapotřebí prozkoumat jazyk, který je pro popis stylovacího jazyka XSL často využíván. To ale neznamená, že jazyk XPATH je určen pouze pro podporu stylovacího jazyka XSL. Jazyk XPATH se stal standardem pro procházení XML dokumentů. Tento jazyk dokáže přesně definovat polohu prvku v XML dokumentu. Výrazy jazyka XPATH připomínají zadávání cesty k souboru v souborovém systému. Jde tedy o způsob, jak nalézt prvek v XML dokumentu. Tímto prvkem nemusí být jen element či atribut, ale například i komentář či jmenný prostor.

Lomítko (/)

Jednotlivé kroky v cestě oddělujeme stejně jako v případě adresářové struktury lomítky. Pokud je prvním znakem vyhledávacího výrazu lomítko, začíná prohledávání XML dokumentu kořenovým elementem. Jinak se jedná o relativní cestu a vyhledávání začíná v aktuálním místě stromu.

```
/zamestnanci/zamestnanec
```

- vrátí všechny elementy zamestnanec jako přímé potomky kořenového elementu zamestnanci

```
jmeno/krestni
```

- vrátí všechny elementy krestni, které jsou potomky elementu jmeno, v případě, že se nacházíme v elementu nadřazeném elementu jmeno

Dvě lomítka (//)

Slouží k překonání libovolného počtu úrovní nacházejících se mezi elementy.

```
zamestnanci//jmeno
```

- vrátí všechny elementy jmeno, které jsou přímými i nepřímými potomky elementu jmeno v případě, že se nacházíme v elementu nadřazeném elementu zamestnanci

```
//jmeno
```

- vrátí všechny elementy nazvané jmeno

```
//*
```

- vrátí všechny elementy

Identifikátory osy

Pomocí identifikátorů osy můžeme určovat osu procházení XML dokumentu.

Rozlišujeme následující identifikátory osy:

- child:: - přímí potomci aktuálního uzlu
- descendand:: - všichni potomci aktuálního uzlu
- descendand-or-self:: - aktuální uzel a všichni jeho potomci
- self:: - aktuální uzel
- ancestor-or-self:: - aktuální uzel a všichni jeho předci
- ancestor:: - všichni předci aktuálního uzlu

- parent:: - rodič aktuálního uzlu
- following:: - všechny uzly, které se v toku XML dokumentu nacházejí za aktuálním uzlem
- preceding:: - všechny uzly, které se v toku XML dokumentu nacházejí před aktuálním uzlem
- following-sibling:: - všichni následující sourozenci aktuálního uzlu
- preceding-sibling:: - všichni předcházející sourozenci aktuálního uzlu
- attribute:: - atributy aktuálního uzlu
- namespace:: - deklarované jmenné prostory

Testy uzlu

Slouží k dalšímu vymezení uzlů.

Uzel určený názvem

Uzel můžeme definovat přímo jeho názvem stejně jako tomu bylo ve všech předchozích případech.

Uzel určený typem

Pomocí výrazů `comment()`, `text()`, `processing-instruction()` a `node()` je možné definovat typ elementu.

```
//text()
```

- vybere všechny textové uzly v dokumentu

Podmínky

Do hranatých závorek umísťujeme další podmínky kladené na uzel. Pokud do hranatých závorek uvedeme číslo, vybereme tím uzel vyhovující podmínce s daným pořadovým číslem.

```
//zakaznik[@id="2001"]
```

- vybere všechny elementy typu zakaznik s atributem id, který má hodnotu 2001

Zkracování syntaxe

- Hvězdička (*) - vybere všechny elementy
- Tečka (.) - je odkaz sám na sebe
- Dvě tečky (..) - stejně jako u souborového systému jde o posun o úroveň výš
- @ - slouží jako náhrada identifikátoru osy attribute::

2.9 Stylování dokumentu

Základním principem technologie XML je důsledné oddělení informačního obsahu od zobrazení. Pro zobrazování obsahu XML dokumentů se v současné době využívají dvě základní metody. Starším způsobem je návrh vzhledu pomocí kaskádových stylů (CSS). Novějším způsobem je použití stylovacího jazyka XSL (eXtensible Stylesheet Language). CSS soubory nejsou primárně určeny pro zobrazování XML dokumentů, ale používají se pro návrh HTML stránek. Díky podobnosti značkovacích jazyků XML a HTML je můžeme použít i pro stylování XML souborů. XSL je novější způsob stylování značkovacích jazyků, který byl navržen přímo pro stylování XML dokumentů.

2.10 CSS

Stylování pomocí CSS není primárně určeno pro XML, a proto nebude v této práci popisováno. Detailní popis CSS lze nalézt například v knize Petra Staníčka [4].

2.11 XSL

Stylovací jazyk XSL je mnohem silnějším nástrojem stylování XML dokumentů než CSS. CSS dovoluje pouze formátovat každý element XML podle určitých pravidel. Naproti tomu XSL poskytuje širší možnosti. Umožňuje přístup ke všem komponentám XML dokumentu (jako jsou elementy, atributy, komentáře a instrukce zpracování), jednoduché třídění a filtrování dat, vkládání smyček a podmínkových struktur a používání proměnných jako v programovacím jazyce. Zároveň poskytuje sadu vestavěných funkcí, které je možné použít pro práci s daty v XML dokumentu.

XSL transformaci lze rozdělit do dvou fází. První je transformace XML dokumentu XSLT (eXtensible Stylesheet Transformation) a druhou definice vzhledu formátování XSL-FO (eXtensible Stylesheet Formatting Objects).

Pokud potřebujeme transformovat XML dokument například do jiného XML dokumentu, do HTML či do textového souboru, bude nám dostačovat XSLT.

Pokud budeme chtít XML dokument transformovat například do formátu RTF, PS či PDF, budeme muset po XSLT použít ještě formátovací objekty, které data připraví do správného formátu.

2.12 XSLT

Stylový formát XSLT je stejně jako definiční formát XSD aplikací XML. XSL je uloženo v samostatném souboru s příponou .xsl a k dokumentu XML je připojeno syntaxí `<?xml-stylesheet type="text/xsl" href="zakaznik.xsl"?>`. V případě, že není potřeba XML soubor napevno spojit s XSL souborem, může být propojení provedeno tak, že oba soubory budou předány jako parametry XSL procesoru (mezi nejznámější procesory patří Saxon, MSXSL či Xalan). Jako obsah atributu href může být použita relativní adresa vzhledem k umístění XML souboru nebo URL. Instrukci zpracování xml-stylesheet je potřeba umístit hned na začátek XML souboru za deklaraci XML a případnou deklaraci typu dokumentu. K jednomu XML dokumentu je možné připojit pouze jeden stylový soubor, případné další jsou standardně ignorovány.

Příklad převedení XML dokumentu zakaznici.xml do podoby HTML pomocí šablony zakaznici.xsl:

zakaznici.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="zakaznik.xsl"?>
<zakaznici>
  <zakaznik id="2000">
    <jmeno>
      <krestni>Pavel</krestni>
      <prijmeni>Novák</prijmeni>
    </jmeno>
```

```
  </zakaznik>
  <zakaznik id="2001">
    <jmeno>
      <krestni>Josef</krestni>
      <prijmeni>Novák</prijmeni>
    </jmeno>
  </zakaznik>
</zakaznici>
```

zakaznici.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
  <head>
    <title>Zákaznící</title>
  </head>
  <body>
  <h1>Zákazníci</h1>
  <xsl:for-each select="/zakaznici/zakaznik/jmeno">
    <p><xsl:value-of select="krestni" /> 
    <xsl:value-of select="prijmeni" /></p>
  </xsl:for-each>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Výsledek transformace po zpracování webovým prohlížečem

```
Zákazníci
Pavel Novák
```

```
Josef Novák
```

Kořenovým elementem XSL souboru je element `<xsl:stylesheet>` s atributy určujícími verzi XSL souboru a jmenný prostor. V době psaní této práce existovalo XSL ve verzích 1.0 a 2.0, přičemž většinou se stále ještě používala verze 1.0. Adresa jmenného prostoru je velmi důležitá, neboť definuje, že se jedná o XSL soubor.

XSL soubor může obsahovat jednu nebo více šablon, z nichž každá je definována pomocí elementu xsl:template. Více šablon využijeme například v tom případě, když jeden ze stylů bude sloužit pro zobrazení na obrazovce a jiný pro tisk.

Atribut match označuje konkrétní komponentu nebo sadu, pro kterou obsahuje šablona transformaci. Konkrétní komponentu či sadu přiřadíme pomocí hodnoty atributu match nazývané lokace, která určuje pozici uzlu nebo uzlů s využitím jazyka XPATH. Lokace, v tomto případě znak /, odpovídá kořenovému uzlu XSLT, který reprezentuje celý dokument XML.

2.12.1 Elementy

Transformační soubory se skládají z elementů, které jsou součástí samotného výstupu XHTML (`<head>`,`<title>`,`<body>`,...) a z prvků vkládajících do kódu data (xsl:value-of). Vzhledem k tomu, že šablona XSL je také XML aplikací, musí být výsledný kód v jazyce XHTML.

Elementy s písmenným (literal) výsledkem

Jde o transformační elementy, které reprezentují elementy jazyka XHTML. Patří sem tedy například tagy (`<head>`,`<title>`,`<body>`,...). Tyto elementy se ve výsledném transformovaném dokumentu zobrazí v takové podobě, jakou mají v transformační šabloně.

Elementy XSL

Elementy XSL jsou zpravidla označeny jmenným prostorem xsl. Tyto elementy obsahují další informace definující úpravu dat z XML.

Element xsl:value-of

Jeden ze základních stylovacích elementů, který vkládá text v konkrétním XML elementu či uzlu jiného typu na výstup. Text obsažený v elementu se skládá ze všech znakových dat obsažených přímo v elementu a všech znakových dat jeho potomků. Jako hodnotu atributu select použijeme stejně jako v jiných XSL elementech výraz jazyka XPATH s určitými omezeními (například nelze použít identifikátory osy). Uspořádáním elementů xsl:value-of v šabloně je možné určit výsledné pořadí výpisu jednotlivých elementů.

```
<xsl:value-of select="jmeno">
```

- v případě vzorového XML dokumentu z počátku kapitoly vypíše: "PavelNovák" (první nalezený element)

Element xsl:for-each

Tento element se používá pro výpis všech elementů nebo dalších typů uzlů XML dokumentu. Stejně tak ho lze použít pro pouhé procházení uzlů a jejich výpis jinými elementy (viz úvodní příklad).

```
<xsl:for-each select="jmeno">
```

- v případě vzorového dokumentu vypíše "PavelNovákJosefNovák" (všechny vyhovující elementy)

Elementy xsl:template a xsl:apply-template

V ukázkovém příkladu je soubor XSL vytvořen pomocí jedné šablony. Jinou možností je přístup pomocí více dílčích šablon (vrátí naprosto shodný výsledek jako u ukázkového XSL souboru). Tento přístup využije tagů `<xsl:template>` a `<xsl:apply-template>`. Základní šablona (která se nachází i v ukázkovém souboru) `<xsl:template match="/">` označuje, že se transformace bude aplikovat na kořenový element XML dokumentu. Jako parametr match se používá u šablon adresa uzlu zapsaná pomocí jazyka XPATH. Pomocí tagu `<xsl:apply-templates>` se definuje, že mají být aplikovány šablony všech uzlů, které jsou potomky (i nepřímými) aktuálně zvoleného uzlu. XSL procesor vyhledá šablonu pro uzel jmeno a na něj aplikuje pravidla, která jsou v něm uvedená. Výstup transformace bude naprosto shodný jako v dříve uvedeném ukázkovém příkladu. Výhodou nově uvedeného zápisu je modularita kódu podobně jako v programovacích jazycích.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/"> <!-- sablona korenoveho uzlu -->
<html>
  <head>
    <title>Zákaznící</title>
  </head>
  <body>
  <h1>Zákazníci</h1>
  <p><xsl:apply-templates /></p>
  </body>
</html>
</xsl:template>

<xsl:template match="jmeno"> <!-- sablona elementu jmeno -->
  <p><xsl:value-of select="krestni" /> 
  <xsl:value-of select="prijmeni" /></p>
</xsl:template>

</xsl:stylesheet>
```

Element pro řazení dat xsl-sort

Jednotlivé uzly v transformační šabloně mohou být libovolně uspořádány. Pokud je však potřeba uzly seřadit, existuje jednodušší možnost. Tou je použití tagu xsl:sort, který se uvádí před řazenými uzly. Element xsl:sort má tři atributy - atribut select udává atribut XML, podle které se budou data řadit, atribut data-type udává typ řazení dat a může být buď text (abecední řazení) nebo number (číselné řazení), poslední atribut order udává, zda budeme data řadit vzestupně (ascending) nebo sestupně (descending).

```
<xsl:for-each select="/zakaznici/zakaznik">
<xsl:sort select="@id" data-type="text" order="descending" />
```

```
  <p><xsl:value-of select="jmeno/krestni" /> 
  <xsl:value-of select="jmeno/prijmeni" /></p>
</xsl:for-each>
```

Uvedený příklad opět vypíše křestní jméno a příjmení každého zákazníka, nyní však budou seřazeni sestupně dle číselných hodnot jejich identifikačních čísel.

2.13 XSL-FO

XSL-FO je navazující částí XSL transformace. Tato technologie se zabývá transformací XML dokumentu pomocí formátovacích objektů. Nejdříve se provede standardní XSL transformace s tím rozdílem, že do výsledného souboru jsou přidány tzv. formátovací objekty. Výsledný dokument tedy obsahuje XML data spolu s formátovacími objekty. Takový dokument je následně předán FO procesoru, který ho převede na námi požadovaný formát, například PDF, PS, RTF či TEX. Mezi programy umožňující FO transformaci patří Passive TEX, FOP či XEP.

Kapitola 3

XML prakticky

Tato kapitola pojednává o praktickém využití jazyka XML. V této souvislosti je nutné vyřešit také otázku výběru programovacího jazyka pro zpracování dokumentů XML.

Volba programovacího jazyka je jednou ze základních otázek každé firmy vyvíjející software. Většina firem v dnešní době vybírá mezi dvěma mainstream technologiemi. Jsou to technologie JSE+JEE (programovací jazyk JAVA) a .NET (programovací jazyky C#, Visual Basic, C++ a J#).

Nebudu zde hodnotit, která platforma je lepší, neboť by taková studie vydala na celou diplomovou práci. Vzhledem k tomu, že firma WinStrom vyvíjí nový účetní systém v programovacím jazyce Java, je i praktická část této diplomové práce vytvořena v tomto programovacím jazyce.

Pokud bychom měli porovnat jazyky C# a Java ve vztahu ke zpracování XML souborů, je vítěz jednoznačný. I příznivci Javy uznávají, že na tomto poli firma Sun (autor programovacího jazyka Java) zaspala a do dnešní doby se jí náskok jazyka C# dohnat nepodařil. Hlavním problémem Javy je její značná „roztříštěnost“ v oblasti zpracování XML dokumentů. Naproti tomu jazyk C# má všechny základní třídy pro práci s XML uložené ve jmenném prostoru `System.Xml`.

Pokud potřebujeme XML dokument zpracovávat, musíme ho nejdříve nějakým způsobem dostat do naší aplikace. K této činnosti slouží XML parsery. Základní činností, kterou parser provádí, je kontrola syntaxe dokumentu (testování, zda je dokument dobře strukturovaný). Dále může provádět validaci pomocí šablony (DTD, XSD, Relax NG). Nakonec parser převede XML dokument na interní datovou reprezentaci, kterou je možné dále zpracovávat. Při parsování dokumentu existují dva základní přístupy - přístup založený na

událostech a přístup založený na stromové struktuře.

3.1 Rozhraní založená na událostech

Nejpoužívanějším API (Application Program Interface) založeném na událostech je SAX (Simple API for XML). Principem tohoto přístupu je postupné čtení XML dokumentu a vyvolávání událostí. Tyto události jsou postupně programově zpracovávány. Mezi výhody tohoto přístupu patří velká rychlost a malá paměťová náročnost. Mezi nevýhody patří sekvenční průchod dokumentem, z čehož vyplývá, že se nemůžeme při průchodu dokumentem vracet. Další nevýhodou je nízkoúrovňové zpracování, které není příliš programátorsky přívětivé. Z těchto důvodů se přístup založený na událostech používá prakticky pouze pro čtení. Parser SAX2 je součástí Java Core API 1.5.

3.2 Rozhraní založená na stromové struktuře

Základním rozhraním založeném na stromové struktuře je DOM (Document Object Model). Principem tohoto přístupu je načítání celého XML dokumentu do stromu uloženého v paměti. Díky tomu je kterýkoliv uzel přístupný jako objekt. Mezi výhody této metody patří jednoduchý přístup k jakémukoliv elementu v paměti. Model DOM je vhodný pro čtení i pro zápis, po úpravě stromové struktury je možné celý strom uložit zpátky do XML souboru. Mezi nevýhody patří malá rychlost a velká paměťová náročnost. Parser DOM Level 3 je součástí Java Core API 1.5.

3.3 JAXP

JAXP (Java API for XML Processing) je rozhraním Java Core API 1.5 pro práci s XML dokumenty. Pomocí tohoto rozhraní můžeme zpracovávat dokumenty metodami SAX i DOM nebo transformovat XML dokumenty pomocí jazyka XSL. Toto rozhraní umožňuje používat libovolný parser (implicitně používá parser Xerces). Rozhraní se nachází v balíčcích `java.xml.parsers`, `java.xml.transform`, `org.xml.sax` a `org.w3c.dom`.

Základní třídy rozhraní JAXP pro parsování XML dokumentů:

SAXParserFactory

- nalezne konkrétní implementaci třídy SAXParserFactory podle nastavení JAXP a vytvoří její instanci

SAXParser

- rozhraní pro práci se SAX parserem

DocumentBuilderFactory

- nalezne konkrétní implementaci třídy DocumentBuilderFactory podle nastavení JAXP a vytvoří její instanci

DocumentBuilder

- rozhraní pro práci s DOM parserem

3.4 SAX

SAX (Simple API for XML) je rozhraní pro proudové zpracování XML dokumentů. Jak už bylo uvedeno, tato metoda zpracovává XML dokumenty pomocí obsluhy událostí.

Jednoduchý příklad použití SAX rozhraní:

zpracovávaný XML dokument - zakaznici.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<zakaznici>
  <zakaznik id="2000">
    <jmeno>
      <krestni>Pavel</krestni>
      <prijmeni>Novák</prijmeni>
    </jmeno>
  </zakaznik>
  <zakaznik id="2001">
    <jmeno>
```

```
      <krestni>Josef</krestni>
      <prijmeni>Novotný</prijmeni>
    </jmeno>
  </zakaznik>
</zakaznici>
```

Inicializace SAX parseru - Hlavni.java

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.XMLReader;
import com.sun.org.apache.xml.internal.utils.DefaultErrorHandler;

public class Hlavni {
  public static void main(String[] args)
  {
    try
    {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      spf.setValidating(false);
      SAXParser sp = spf.newSAXParser();
      XMLReader parser = sp.getXMLReader();
      parser.setErrorHandler(new DefaultErrorHandler());
      parser.setContentHandler(new SAXHandler());
      parser.parse("zakaznici.xml");
    }
    catch (Exception e)
    {
      e.printStackTrace();
    }
  }
}
```

Parsování XML pomocí metody SAX - SAXHandler.java

```
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;

public class SAXHandler extends DefaultHandler {
  private Boolean uvnitrElementu = false;
  private static final int VELIKOST_BUFFERU = 100;
  private StringBuffer hodnota = new StringBuffer(VELIKOST_BUFFERU);

  public void startElement(String uri, String localName,
                           String qName, Attributes atts)
  {
    if ((qName.equals("krestni"))||(qName.equals("prijmeni")))
    {
      hodnota.setLength(0);
      uvnitrElementu = true;
    }
  }

  public void endElement(String uri, String localName, String qName)
  {
    if (qName.equals("krestni"))
    {
      System.out.println("Křestní jméno: "+hodnota);
      uvnitrElementu = false;
    }

    if (qName.equals("prijmeni"))
    {
      System.out.println("Příjmení: "+hodnota);
      uvnitrElementu = false;
    }
```

```
  }

  public void characters(char[] ch, int start, int length)
  {
    if (uvnitrElementu == true)
    {
      hodnota.append(ch,start,length);
    }
  }
}
```

3.5 DOM

Tato metoda zpracování XML dokumentů je z programátorského hlediska mnohem jednodušší než metoda SAX. Celý XML dokument je uložen v podobě objektů do paměti. Z toho pramení možnost dokument v paměti libovolně procházet či upravovat. Nevýhodou této metody je její vysoká paměťová náročnost.

Následuje příklad na parsování XML dokumentu metodou DOM, který používá stejný XML soubor a poskytuje stejné výsledky jako tomu bylo v případě metody SAX.

Parsování XML pomocí metody DOM - Hlavni.java

```
import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import com.sun.org.apache.xml.internal.utils.DefaultErrorHandler;

public class Hlavni {
  public static void main(String[] args) {
    try
    {
```

```
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    dbf.setValidating(false);
    DocumentBuilder builder = dbf.newDocumentBuilder();
    builder.setErrorHandler(new DefaultErrorHandler());
    Document doc = builder.parse(new File("zakaznici.xml"));
    ZpracujDocument(doc);
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}

public static void ZpracujDocument(Document doc)
{
  NodeList nl = doc.getElementsByTagName("jmeno");

  for (int i = 0 ; i < nl.getLength(); i++)
  {
    for (int j = 0 ; j < nl.item(i).getChildNodes().getLength() ; j++)
    {
      Node child = nl.item(i).getChildNodes().item(j);
      if (child.getNodeName().equals("krestni"))
      {
        System.out.println("Křestní jméno: "+GetInnerText(child));
      }
      if (child.getNodeName().equals("prijmeni"))
      {
        System.out.println("Příjmení: "+GetInnerText(child));
      }
    }
  }
}
```

```
  public static String GetInnerText(Node node)
  {
    NodeList childs = node.getChildNodes();
    for (int j = 0 ; j < childs.getLength() ; j++)
    {
      if (childs.item(j).getNodeType() == Node.TEXT_NODE)
      {
        return (childs.item(j).getNodeValue().trim());
      }
    }
    return null;
  }
}
```

3.6 JDOM

Protože aplikace rozhraní DOM v Javě není příliš jednoduchá, vzniklo speciální rozhraní JDOM, které má práci se stromovou strukturou XML dokumentu v Javě usnadnit. Toto rozhraní však není součástí Java Core API 1.5.

3.7 XSLT

JAXP obsahuje také podporu pro XSLT transformace. Jednoduchým způsobem tak můžeme XML dokument transformovat například do jiného XML dokumentu či XHTML dokumentu. Příklad XSLT transformace (na vstupu máme dokumenty XML (inputstream) a XSL (xslfile), na výstupu dostaneme řetězec obsahující transformovaná data (result)):

```
TransformerFactory tFactory = TransformerFactory.newInstance();
Transformer transf = tFactory.newTransformer
  (new javax.xml.transform.stream.StreamSource(xslfile));
transf.transform(new javax.xml.transform.stream.StreamSource(inputStream),
  new javax.xml.transform.stream.StreamResult(result));
```

Kapitola 4

Počítačová bezpečnost

Tato kapitola se zabývá fenoménem posledních let, počítačovou bezpečností. Protože vše v této diplomové práci směřuje ke komunikaci s Portálem veřejné správy, budou po nezbytném teoretickém úvodu do počítačové bezpečnosti následovat témata týkající se elektronického podpisu a zabezpečení přenosu dat.

Počítačová bezpečnost je obor, který se vyvíjí neuvěřitelnou rychlostí. I pár měsíců staré informace v tomto oboru už mohou být zastaralé. Zároveň je tato disciplína postavena na mohutných matematických základech. Tato kapitola přináší informace o počítačové bezpečnosti aktuální v době napsání této práce a nezabývá se matematickým pozadím dále uvedených algoritmů, ale pouze jejich základním funkčním popisem a vlastnostmi.

4.1 Úvod do počítačové bezpečnosti

Celá vědecká disciplína zabývající se počítačovou bezpečností se nazývá kryptologie a je rozdělena na dvě dílčí disciplíny - kryptografie se zabývá kódováním a šifrováním dat zatímco kryptoanalýza se zabývá analýzou algoritmů a zašifrovaných dat.

Data v čitelné podobě jsou označována jako otevřený text a jejich šifrováním dostaneme šifrovaný text. Opačný postup nazýváme dešifrování. Algoritmus provádějící šifrování a dešifrování označujeme jako kryptografický algoritmus. Kryptografický algoritmus můžeme dále dělit na šifrovací algoritmus a kódovací algoritmus. Šifrovací algoritmus potřebuje narozdíl od kódovacího algoritmu ke své úspěšné funkci ještě tzv. šifrovací klíč. Prolomení algoritmu nastává v okamžiku, kdy je možné číst chráněná data bez znalosti šifrovacího klíče či kódovacího algoritmu.

4.2 Kryptologie

4.2.1 Symetrické a asymetrické šifrovací algoritmy

V současné době se používájí dva základní typy šifrovacích algoritmů - symetrické a asymetrické. Symetrické algoritmy jsou založeny na principu sdíleného klíče. Obě komunikující strany mají shodný klíč pomocí kterého zprávu zašifrují i dešifrují. Tento způsob šifrování je vhodný například pro bezpečnou komunikaci dvou osob, ale pro komunikaci více osob je z bezpečnostních důvodů nevhodný (každá strana musí vlastnit klíče všech stran, se kterými chce komunikovat). Kvůli tomu vznikly asymetrické algoritmy. Každá z komunikujících stran si v tomto případě vytvoří dvojici klíčů. Jeden z nich nechá ukrytý v tajnosti (soukromý klíč) a druhý zveřejní (veřejný klíč) například na svých webových stránkách. Pokud chce někdo tomuto člověku zaslat zašifrovanou zprávu, stáhne si veřejný klíč a pomocí něj zašifruje zprávu. Zpráva může být dešifrována jedině soukromým klíčem, který vlastní příjemce. Každá strana tedy musí střežit v tajnosti pouze svůj soukromý klíč. Dalo by se říci, že asymetrické algoritmy jsou mnohem lepší než symetrické, tak proč tedy symetrické vůbec používat? Důvodem je matematická složitost asymetrických algoritmů, která je mnohonásobně vyšší než je tomu u algoritmů symetrických.

Jednou z možností, jak zkombinovat výhody obou zmíněných algoritmů, jsou hybridní algoritmy. Před šifrováním je vygenerováno náhodné číslo, relační symetrický šifrovací klíč, určený pouze pro tuto operaci. Otevřený text je pomocí symetrického algoritmu s tímto klíčem zašifrován velmi rychle. Klíč poté zašifrujeme pomocí asymetrické kryptografie. Vzhledem k malé délce klíčů (ve srovnání s délkou chráněných dat je délka klíče zanedbatelná), trvá jeho šifrování velmi krátkou dobu. Zašifrovaný symetrický klíč je přiložen ke zprávě a odeslán. Druhá strana tedy musí nejprve použít asymetrický algoritmus (použije veřejný klíč), aby získala symetrický klíč pro tuto relaci a mohla zprávu dešifrovat.

4.2.2 Šifrování a podepisování

Šifrování není jedinou možností ochrany dat. Můžeme použít i způsob nazývaný podepisování. Podepisování plní dvě základní funkce. První funkcí je zajištění integrity dat, což znamená, že data od doby svého podepsání do doby kontroly platnosti podpisu nesmí být nijak změněna. Druhou funkcí je nepopiratelnost, která musí zajistit jedinečnou identifikaci osoby, která danou zprávu podepsala. Podpis tedy vytvoří jediný člověk a ostatní si mohou

jeho zprávu přečíst a ověřit si identitu autora a zda zprávu někdo nezměnil. Tento postup nelze z hlediska nepopiratelnosti uplatnit u symetrické kryptografie, neboť obě strany vlastní shodný klíč. U asymetrické kryptografie díky soukromým a veřejným klíčům tento postup použít lze a bude popsán v kapitole o digitálním podpisu.

4.2.3 Proudové a blokové šifry

Šifrovací algoritmy můžeme dělit podle množství šifrovaných dat na proudové a blokové. Jak už názvy napovídají, proudové šifrování spočívá ve zpracování dat znak po znaku a v blokovém zpracování je v jednu chvíli zpracováván celý blok dat. Blokové šifry můžeme dále dělit podle způsobu, jakým zpracovávají data. Popis šifer vychází z knihy [14].

ECB (Electronic code book)

Jedná se o základní mód blokových šifer. Na vstup přichází otevřený text, na výstupu se objeví šifrovaný text. Pokud přijde na vstup podruhé stejný blok, je zašifrován stejně jako blok první. Každému bloku textu je tedy přiřazen jiný blok textu, což umožní útočníkovi získat po určité době seznam dvojic otevřený text-šifrovaný text. Jednotlivé bloky v módu ECB jsou navzájem nezávislé, což umožňuje útočníkovi tato data nahradit jinými.

CBC (Cipher block chaining)

Zavádí do blokové šifry určitou závislost mezi šifrovanými bloky. Využívá k tomu logickou funkci nonekvivalence (XOR). Po zašifrování prvního bloku otevřeného textu je získán blok zašifrovaného textu. Tento blok se xoruje s druhým blokem otevřeného textu a teprve výsledek této operace vstupuje do šifrovacího algoritmu. Problémem tohoto způsobu je hlavně první blok, který není xorován, což ho degraduje na způsob ECB. Řešením může být použití inicializačního vektoru, se kterým se první blok xoruje. Tento inicializační vektor může být zcela náhodný a je přiložen k zašifrované zprávě.

CFB (Cipher feedback mode) a OFB (Output feedback mode)

Tyto dva módy přepínají blokovou šifru do proudového režimu. Aby nedocházelo k problému se stejným šifrováním stejného znaku, je do systému zaveden pseudonáhodný prvek. Vlastní data jsou šifrována posloupností pseudonáhodných čísel, jejich generování je řízeno zpětnou vazbou.

V případě CFB je generátor pseudonáhodných čísel řízen symboly šifrovaného textu. Zpětná vazba je tedy zapojena až z úplného výstupu šifrovacího zařízení.

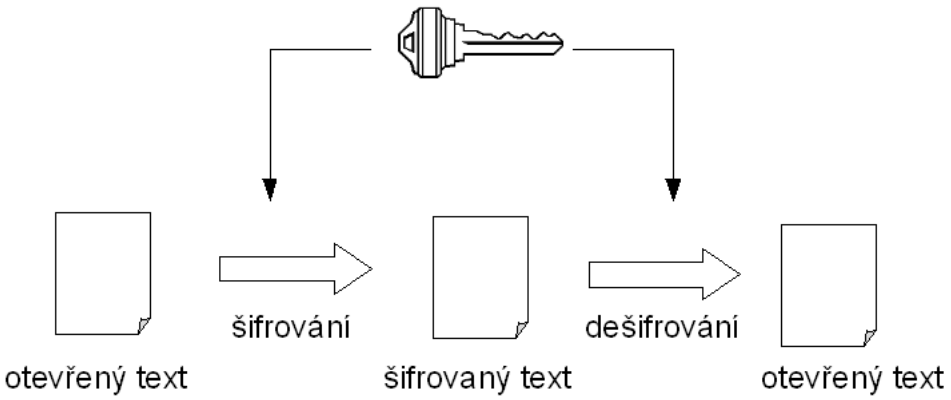
V případě OFB je generátor pseudonáhodných čísel řízen výstupem samotného generátoru. Zpětná vazba je v tomto případě zapojena již z výstupu generátoru pseudonáhodných čísel. Toto je zásadní rozdíl oproti módu CFB.

EDE (Encryption-Decryption-Encryption)

Jedná se o zapojení tří šifrovacích bloků do série. První a poslední blok data šifruje klíčem K1. Prostřední blok zašifrovaná data dešifruje klíčem K2. Délka klíče se tedy zdvojnásobí, existují ale i varianty, které využívají tří různých klíčů K1, K2 a K3, což samozřejmě délku klíče ztrojnásobuje.

4.3 Symetrické šifrování

Zásadní výhodou symetrického šifrování je jeho rychlost. Mezi nevýhody naopak patří problematická počáteční výměna klíčů. Nevýhodou je také to, že uživatel nemá plnou kontrolu nad bezpečností, protože stejný klíč vlastní i druhá strana. S tím souvisí i to, že počet klíčů v systému roste s druhou mocninou komunikujících stran. Z těchto důvodů vyplývá, že symetrické šifrování není použitelné pro digitální podpisy.



Obrázek 4.1: Princip symetrického šifrování

4.3.1 DES

DES (Digital Encryption Standard) je stále nejpopulárnějším symetrickým algoritmem. Byl vytvořen roku 1975 firmou IBM ve spolupráci s NSA (National Security Agency) a stal se americkou vládní normou pro šifrování.

Jedná se o symetrickou blokovou šifru s délkou bloku 64 bitů. Klíče jsou na dnešní dobu velmi krátké (56 bitů). Algoritmus se obvykle používá v módech CBC či CFB. Dlouhou dobu byl tento algoritmus považován za neprolomitelný, avšak v roce 1999 byl na speciálně sestaveném počítači prolomen za méně než 24 hodin.

4.3.2 3DES

3DES vychází z algoritmu DES, ale je bezpečnější. Využívá klíč o délce 112 bitů. V principu jde o několikanásobné použití algoritmu DES. Algoritmus DES je použit třikrát, v prvním a třetím kroku šifruje (pomocí první části klíče) a ve druhém kroku dešifruje (pomocí druhé části klíče). Někdy je 3DES imlementován tak, že v druhém kroku používá rovněž odlišný klíč. Délka klíče je tedy v takovém případě 168 bitů. 3DES je tedy pomalejší než DES, ale bezpečnější a není složité algoritmus DES na 3DES upravit.

4.3.3 AES

V devadesátých létech minulého století se stal algoritmus DES pro šifrovací účely nevyhovující. Organizace NIST (National Institute of Standards and Technology) vyhlásila soutěž, ve které hledala nový bezpečnostní standard. Soutěž nakonec vyhrál algoritmus Rijndael (autoři: Vincent Rijmen a Joan Daemen, Belgie), který byl přejmenován na AES (Advanced Encryption Standard). Z čehož vyplývá, že se s algoritmem do budoucna počítá jako se standardem. Algoritmus má klíče délky 128, 192 a 256 bitů.

4.3.4 RCx

Do skupiny RCx algoritmů patří RC2, RC4, RC5 a RC6. Autorem těchto algoritmů je Ronald L. Rivest. Tyto algoritmy jsou patentovány firmou RSA Security.

RC2

- bloková šifra
- vytvořena roku 1987
- délka bloků 64 bitů
- délka klíčů proměnlivá 8-128 bitů
- zdrojové kódy zcizeny roku 1996

RC4

- proudová šifra
- vytvořena roku 1987
- zdrojové kódy zcizeny roku 1994

RC5

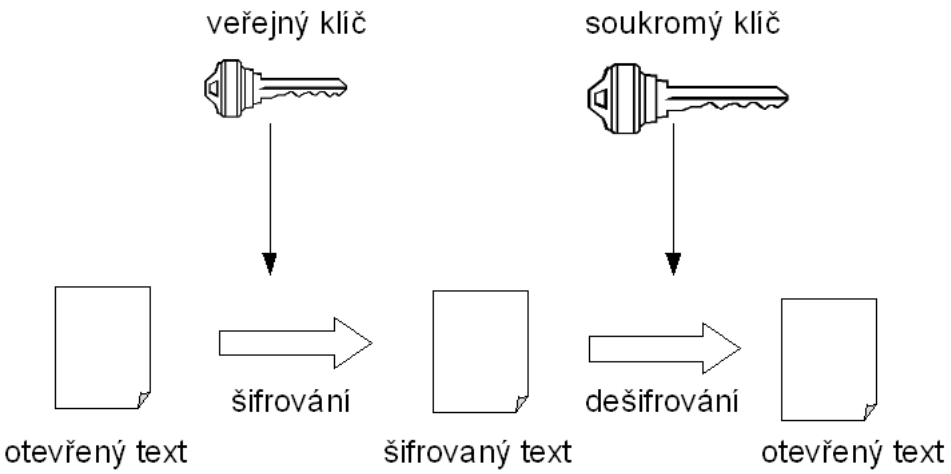
- bloková šifra
- vytvořena roku 1994
- délka bloků 32, 64 nebo 128 bitů
- délka klíče 0-2040 bitů
- organizace distributed.net prolomila tuto šifru s délkou klíčů 56 a 64 bitů, v současnosti pracuje na prolomení šifry o délce 72 bitů

RC6

- bloková šifra na principu RC5
- vytvořena roku 1998
- byla mezi nejlepšími pěti šiframi při hledání AES (viz AES)
- délka bloků 128 bitů
- délka klíčů 128, 192, 256 bitů

4.4 Asymetrické šifrování

Mezi výhody asymetrického šifrování patří to, že počet klíčů v systému roste lineárně s počtem komunikujících dvojic. Tento způsob šifrování je vhodný pro tvorbu digitálního



Obrázek 4.2: Princip asymetrického šifrování

podpisu. Mezi nevýhody asymetrického šifrování patří jeho značná matematická složitost a z toho důvodu výpočetní náročnost.

4.4.1 RSA

Algoritmus RSA (Ron Rivest, Adi Shamir, Len Adleman - 1977) pro výměnu klíčů a tvorbu elektronického podpisu patří mezi nepsané standardy. Ve srovnání se symetrickými algoritmy je samozřejmě mnohonásobně pomalejší, proto se často používá v hybridním šifrování (viz dále). Algoritmus je založen na složitosti faktorizace velkých prvočísel. Byl podroben mnoha testovacím útokům, ale žádný z nich nebyl úspěšný. Stal se standardem v mnoha zemích, v USA byl dokonce patentován. Patent vypršel v roce 2000 a algoritmus je nyní volně šiřitelný.

4.4.2 DSA (DSS)

National Institute of Standards and Technology navrhl algoritmus DSA (Digital Signature Algorithm) jako standard pro digitální podpisy DSS (Digital Signature standard). Tento standard byl přijat v roce 1994. DSA je postaven na problému diskrétních logaritmů. Zatímco RSA může být používáno pro šifrování i digitální podpisy, DSA může být použito pouze jako poskytovatel digitálních podpisů. Ačkoliv je DSA používáno v mnoha programech, má mnoho kritiků, kterým se nelíbí jeho výkonnost ani způsob, jakým se stal

algoritmus DSA standardem.

4.5 Hybridní šifrování

Hybridní princip šifrování spojuje výhody symetrického a asymetrického způsobu šifrování. Díky tomu umožňuje tvorbu digitálního podpisu s přijatelnou časovou náročností. Jednotlivé algoritmy jsou syntézou libovolného symetrického a asymetrického algoritmu.

4.6 Hashovací funkce

Hashovací funkce nebo také hashovací algoritmy jsou takové postupy, které přijímají libovolná vstupní data a vyprodukují výstup fixní délky, který označujeme jako hash nebo digitální otisk. Pokud použijeme dvě různé sady vstupních dat, měly by se jejich hash hodnoty lišit. Hashovací funkce jsou používany například v kryptografii či při kontrole obsahu souborů. Předpoklady správné hash funkce jsou následující: funkce je jednosměrná (nelze získat zpět hodnotu z hashe), funkce dokáže vytvořit hash z libovolných vstupních dat a je bezkolizní (nelze nalézt jiný text, který bude mít stejný hash). Pokud se nad tím trochu zamyslíme, musí nám být jasné, že žádná hashovací funkce nemůže být bezkolizní, neboť vstupních textů může být nekonečné množství a hashů jen omezené (například u funkce MD5 2^128). To je sice pravda, ale nalezení dvou vstupních textů se stejným hashem je složité a u aktuálně používaných hashovacích algoritmů by mělo být neřešitelné.

4.6.1 MD4

- délka otisku: 128 bitů
- autor algoritmu: Ronald Rivest (1990)
- nalezeny kolize (1995)
- již se nepoužívá

4.6.2 MD5

- délka otisku: 128 bitů
- autor algoritmu: Ronald Rivest (1991)
- nalezeny kolize (2004)

- přestává se používat (hlavně v oblasti digitálních podpisů není použití bezpečné)
- problémem je „dlouhý život“ funkce (1991-2004) a její nasazení v mnoha systémech

4.6.3 SHA-0

- funkce byla standardem do roku 1994
- nalezeny kolize (2004)
- dnes se již prakticky nepoužívá

4.6.4 SHA-1

- délka otisku: 160 bitů
- stále ještě standard NIST
- v současné době je funkce považována za slabou (zčásti prolomena)

4.6.5 SHA-2

- více různých algoritmů dle délky otisku (SHA-256, SHA-384, SHA-512)
- délka otisku: 256, 384, 512 bitů
- funkce je považována za bezpečnou
- doporučována pro nové aplikace
- do roku 2010 by se měla stát standardem NIST

4.7 Elektronický a digitální podpis

Elektronický podpis musí splňovat dvě základní vlastnosti. Jsou jimi nepopiratelnost a integrita. Nepopiratelnost zajišťuje, že osoba podepsaná pod zprávou nemůže nijak popřít to, že zprávu odeslala. Integrita naopak deklaruje, že odeslaná zpráva nebyla v průběhu přenosu od odesílatele k adresátovi změněna.

Jak to tedy celé funguje? Odesílatel podepíše svůj text a vznikne digitálně podepsaný text. Tento text je odeslán příjemci. Příjemce ověří, zda podpis patří opravdu odesílateli. Následně ověří, zda nedošlo k modifikaci zprávy během přenosu. Pokud některé z ověření selže, nemůže příjemce podpisu pod textem věřit. Pokud projdou testy v pořádku, můžeme si být jisti identitou odesílatele a integritou zprávy.

Z výše uvedených principů vyplývá, že pro digitální podpis jsou vhodné asymetrické šifrovací algoritmy. U těchto algoritmů existuje jeden soukromý klíč, který udržuje vlastník v bezpečí. K tomuto klíči existuje párový veřejný klíč, který je volně k dispozici a kdokoliv si může ověřit, že danou zprávu podepsal vlastník soukromého klíče. Tento postup je tedy přesně opačný než ten u šifrování pomocí asymetrických algoritmů. Je to z toho důvodu, že u šifrování potřebujeme, aby zprávu mohl zašifrovat kdokoliv a dešifrovat jí mohla pouze osoba, které je zpráva určena. Naproti tomu u podpisu může zprávu podepsat pouze jediná osoba a kdokoliv si může ověřit její identitu.

Jak už bylo uvedeno výše, asymetrická kryptografie je výpočetně velmi náročná. Naštěstí existuje způsob, jak tento problém odstranit. Pomůžou nám hashovací funkce, které jsme si uvedli v předcházejícím odstavci. Vytvoříme si tedy digitální otisk naší zprávy (hash). Tento otisk zašifrujeme pomocí našeho soukromého klíče, přiložíme k textu zprávy a odešleme příjemci. Příjemce rozdělí přijatou zprávu na text a hash. Pomocí stejné hashovací funkce, jakou jsme použili my, vytvoří adresát hash textu. Poté rozšifruje pomocí veřejného klíče hash přijatý jako součást zprávy. Obě hash hodnoty se porovnají a v případě shody může být podpis označen za platný. V případě neshody není možné podpis považovat za důvěryhodný.

4.7.1 Certifikovaný klíč

Je vidět, že není problém si vygenerovat soukromý a veřejný klíč (prakticky to opravdu není nic složitého například s pomocí utility OpenSSL). Jak ale takovému klíči důvěřovat, když si ho tímto způsobem může vygenerovat každý a vystupovat pod cizí identitou? V takovém případě musí do hry vstoupit třetí strana, která se nazývá certifikační autorita. Pokud nám tedy někdo zašle podepsaný e-mail, obrátíme se na certifikační autoritu a ta nám řekne, zda mail skutečně podepsal daný člověk. Samozřejmě se může stát, že dojde ke zcizení soukromého klíče jeho vlastníkovi. V takovém případě je postup podobný jako při ztrátě kreditní karty. Vlastník nahlásí certifikační autoritě ztrátu svého klíče a ta ho zneplatní. Slovní spojení ztráta klíče je poněkud zavádějící, protože nejde o fyzickou ztrátu klíče, ale stačí získání podezření, že se ke klíči dostala nepovolaná osoba. Certifikační autorita v takovém případě umístí certifikační klíč na Certificate Revocation List (seznam odvolaných certifikátů). Tento seznam vydávají certifikační autority každých 24-48 hodin. Z právnického hlediska je příjemce povinnen si před vlastním ověřením podpisu zkontro-

lovat, zda není příslušný certifikát v seznamu CRL dané autority. Problémem zůstává, že mnohé programy mají stahování seznamu CRL implicitně vypnuto, případně po určitém časovém intervalu stahování CRL zruší. Toto je velký problém. Pokud bychom použili přirovnání ke kreditní kartě, tak by naše karta byla zablokována, ale bankomat by vydával peníze dál. Existuje ještě jeden způsob ověřování certifikátů, který nazýváme OCSP (Online Certificate Status Protocol), který ověřuje platnost každého certifikátu přímo dotazem na příslušné certifikační autoritě. Tento postup je mnohem lepší, ale vyžaduje neustálé připojení daného počítače k internetu.

4.7.2 Certifikát dle normy X.509

Nejpoužívanější normou pro formát certifikátu je X.509. Skládá se z osobních údajů vlastníka (jméno, příjmení, rodné číslo, atd.), služebních údajů (ID certifikátu, platnost, použité certifikáty), veřejného klíče, který certifikuje, a digitálního podpisu všech uvedených položek.

Položky certifikátu

ID certifikátu - jednoznačné sériové číslo certifikátu

Datum vydání - kdy byl certifikát vydán

Platnost do - časové omezení platnosti

Omezení certifikátu - účel certifikátu

ID certifikační autority - vydavatel certifikátu

Algoritmy CA - algoritmy, které používá CA pro podepisování

Veřejný klíč CA - nemusí být součástí

ID vlastníka - rodné číslo či IČO

Veřejný klíč vlastníka - vlastní certifikovaný klíč

Podpis - předcházející data jsou podepsána soukromým klíčem CA

4.7.3 Třídy certifikátů

Certifikáty podle jejich účelu dělíme do čtyř základních tříd.

class 1

- CA pouze ověřuje, zda je jméno, které chceme do certifikátu zapsat, volné
- hodí se pro různá testování

class 2

- CA se spoléhá na ověření údajů v certifikátu třetí stranou (nejčastěji notářem)
- příliš se nepoužívá

class 3

- standardní certifikát vhodný pro většinu účelů
- identita žadatele o certifikát je ověřována dle politiky CA (nejčastěji osobní návštěva s donesením dvou osobních dokladů)

class 4

- certifikát třídy 3 navíc doplněný o speciální autorizaci k nějaké činnosti

4.7.4 Certifikační autorita

Certifikační autorita je organizace, u které si můžeme ověřovat platnost digitálních podpisů. Celá tato organizace je postavena na důvěře, kterou do ní uživatelé vkládají. Základním majetkem každé certifikační autority je její soukromý a veřejný podpisový klíč. Soukromý klíč je nejstřeženějším majetkem certifikační autority. Je používán pro podepisování certifikátů uživatelů, ale nikdy se nesmí dostat na veřejnost. Pokud by se tak stalo, ztratila by daná certifikační autorita svou důvěru a stejně tak by se již nedalo věřit ani certifikátům podepsaným touto autoritou. Z tohoto důvodu jsou soukromé klíče certifikačních autorit uchovávány v hardwarových zařízeních, které umožňují podepisovat certifikáty, ale nikdy se nedá klíč z těchto zařízení získat. Naproti tomu veřejný klíč je volně dostupný na internetových stránkách CA a uživatelé si ho mohou nainstalovat na svůj počítač a automaticky tak budou důvěřovat všem certifikátům vydaným danou certifikační autoritou.

4.7.5 Základní funkce certifikační autority

- Vydávání certifikátů různých druhů

- Ověřování pravdivosti informací v žádosti o certifikát (nejčastěji pomocí dvou dokladů)
- Zneplatňování certifikátů
- Zveřejňování seznamu zneplatněných certifikátů (Certificate Revocation List)

4.7.6 PKI normy

Celou problematiku týkající se elektronického podpisu, certifikátů a certifikačních autorit označujeme jako PKI (Public Key Infrastructure). Tato problematika má samozřejmě mnoho standardů a norem. Za mezinárodní standard jsou považovány normy firmy RSA Security, které se označují jako PKCS (Public Key Cryptography Standards).

4.7.7 Standardy PKCS

PKCS #1: RSA Cryptography Standard

PKCS #3: Diffie-Hellman Key Agreement Standard

PKCS #5: Password-Based Cryptography Standard

PKCS #6: Extended-Certificate Syntax Standard

PKCS #7: Cryptographic Message Syntax Standard

PKCS #8: Private-Key Information Syntax Standard

PKCS #9: Selected Attribute Types

PKCS #10: Certification Request Syntax Standard

PKCS #11: Cryptographic Token Interface Standard

PKCS #12: Personal Information Exchange Syntax Standard

PKCS #13: Elliptic Curve Cryptography Standard

PKCS #15: Cryptographic Token Information Format Standard

4.7.8 Elektronický podpis v České republice

Dle zákona o elektronickém podpisu jsou definovány tři druhy certifikačních autorit, v terminologii zákona se pro ně používá název „poskytovatelé certifikačních služeb“.

1. poskytovatel certifikačních služeb
2. poskytovatel certifikačních služeb vydávající kvalifikované certifikáty
3. akreditovaný poskytovatel certifikačních služeb

Akreditovaní poskytovatelé certifikačních služeb:

- První certifikační autorita a.s. (http://www.ica.cz/)
- Česká pošta s.p. (http://qca.postsignum.cz/)
- eIdentity a.s. (http://www.ie.cz/)

4.7.9 Praktické zkušenosti s elektronickým podpisem

Pro komunikaci s Portálem veřejné správy je potřeba elektronický podpis jedné z akreditovaných certifikačních autorit. Pro svůj elektronický podpis jsem si zvolit certifikační autoritu České pošty - PostSignum. Tato certifikační autorita se stala druhou akreditovanou certifikační autoritou na českém trhu a byla akreditována dne 3. 8. 2005 Ministerstvem Informatiky. Své služby začala poskytovat dne 1. 9. 2005.

Pro vydání elektronického podpisu je zapotřebí dodat na certifikační autoritu (v tomto případě na některou z vybraných pošt) dva exempláře vytištěné objednávky certifikačních služeb. Na této objednávce vyberete požadovanou verzi certifikátu a vyplníte osobní údaje, které budou součástí vašeho certifikátu. Jako součást certifikátu je možné si zažádat o tzv. identifikátor MPSV, který je vyžadován některými institucemi státní správy. Dále budete samozřejmě potřebovat vámi vygenerovaný veřejný klíč. Certifikační autority mají pro generování klíčů vytvořeny vlastní programy, které běžného uživatele odstíní od poměrně složitých mechanismů PKI. U České pošty je možné vygenerování žádosti o certifikát pomocí webové aplikace či pomocí programu PostSignum Tools, který se nachází ke stažení na stránkách certifikační autority PostSignum. Výsledkem bude vytvoření žádosti s vašimi osobními údaji, ke kterým bude přiložen veřejný klíč a vše bude podepsáno vaším

soukromým klíčem, který bude taktéž vygenerován. Podpisem pomocí soukromého klíče stvrzujeme, že jsme skutečně vlastníkem soukromého klíče. Vše doručíme na disketě na pobočku certifikační autority.

Na pobočce certifikační autority dojde k ověření naší totožnosti (obvykle pomocí dvou průkazů totožnosti) a je nám na disketu nahrán certifikát.

Digitální certifikát na svém domácím počítači spojíme se soukromým klíčem. Výsledek exportujeme do systémového úložiště nebo do souboru standardu PKCS#7 (p7, pfx).

Kapitola 5

Počítačová bezpečnost prakticky

5.1 Kryptografie v jazyce JAVA

Pro šifrování v jazyce Java slouží balíček JCE (Java Cryptography Extension), což je pracovní prostředí nebo aplikační rámec (framework) pro šifrování a dešifrování, pro tvorbu klíčů, pro tvorbu dohod o výměně klíčů a pro autentizační kódy zpráv. Kryptografické algoritmy obsažené v balíku JCE jsou určeny pro tvorbu symetrických, asymetrických, blokových i proudových šifer s dodatečnou podporou bezpečných proudů a zapečetěných objektů.

Aplikační rozhraní balíčku JCE umožňuje vývojáři pracovat s rozsáhlou sadou šifrovacích algoritmů, aniž by musel znát jejich konkrétní implementaci (podobně jako to dělá rozhraní JAXP u práce s XML). Firma Sun poskytla v základní verzi balíku JCE základní šifrovací algoritmy (Sun JCE), zároveň ovšem umožnila dalším firmám vytvářet vlastní knihovny šifrovacích algoritmů.

Díky vývozním omezením Spojených států amerických nemohou být některé kryptografické algoritmy součástí distribučních balíků Javy (JDK, JRE). Pokud potřebujeme využívat kompletní sadu šifrovacích algoritmů, musíme si nainstalovat balíček Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files 5.0, který je ke stažení na stejných stránkách jako balíky JDK a JRE.

Pravděpodobně nejpoužívanějším poskytovatelem kryptografických algoritmů v jazyce Java je knihovna Bouncy Castle JCE [21]. Je to bezplatná, avšak velice obsáhlá implementace JCE, která pracuje s mnoha algoritmy, které nejsou obsaženy v základní implementaci Sun JCE dodávaného jako součást JDK.

5.2 Šifrování

Základní třídou pro šifrování v balíku Bouncy Castle je CMSEnvelopedDataGenerator z balíčku `org.bouncycastle.cms`, zde je základní princip jejího použití:

```
CMSEnvelopedDataGenerator  fact = new CMSEnvelopedDataGenerator();
fact.addKeyTransRecipient(x509);
String algorithm = CMSEnvelopedDataGenerator.DES_EDE3_CBC;
CMSEnvelopedData data = fact.generate(new CMSProcessableByteArray(buffer),
  algorithm, "BC");
```

Na první řádce vytvoříme instanci třídy CMSEnvelopedDataGenerator. Dále připojíme šifrovací klíč, který jsme již dříve načetli ze souboru či z úložiště. Na třetí řádce zvolíme jeden z algoritmů pro šifrování. Na výběr máme následující algoritmy: DES_EDE3_CBC, RC2_CBC, IDEA_CBC, CAST5_CBC, AES128_CBC, AES192_CBC, AES256_CBC. A konečně na čtvrté řádce dojde k vygenerování zašifrovaných dat.

5.3 Podepisování

Klíčovou třídou pro podepisování v balíku Bouncy Castle je CMSSignedDataGenerator z balíčku `org.bouncycastle.cms`. Zde je základní princip jejího použití:

```
CMSSignedDataGenerator signGen = new CMSSignedDataGenerator();
signGen.addSigner(privatekey, cert, CMSSignedDataGenerator.DIGEST_SHA1);
signGen.addCertificatesAndCRLs(certs);
CMSProcessable content = new CMSProcessableByteArray(buffer);
CMSSignedData signedData = signGen.generate(content, false, "BC");
```

Na první řádce vytvoříme instanci třídy CMSSignedGenerator. Dále připojíme privátní klíč, veřejný certifikát a zvolíme podpisový algoritmus. Na výběr máme následující algoritmy: DATA, DIGEST_SHA1, DIGEST_SHA224, DIGEST_SHA256, DIGEST_SHA384, DIGEST_SHA512, DIGEST_MD5, DIGEST_GOST3411, ENCRYPTION_RSA, ENCRYPTION_DSA, ENCRYPTION_GOST3410, ENCRYPTION_ECGOST3410. Na třetí řádce se přidají příslušné kořenové certifikáty a CRL. Nakonec se vstupní data převedou na požadovaný typ a zpracují se metodou generate.

5.4 BASE64

Výstupem šifrování i podepisování je pole bytů. Vzhledem k tomu, že tyto byty zapisujeme do textového souboru (XML), musí nás zajímat jejich výsledný tvar. Výsledné byty naší šifry by totiž mohly odpovídat znakům ASCII používaným například jako znaky konce řádků nebo konce souboru. Tím bychom poškodili obsah datového souboru, jehož další část by se mohla stát pro aplikaci nepoužitelnou. Aby k tomu nedošlo, využijeme techniku označovanou jako BASE64. Tato technika umožní převést skupinu tří bytů na skupinu čtyř tisknutelných znaků. V podstatě lze říci, že tímto způsobem mapujeme každých šest bitů na osmibitový znak ASCII (metoda se označuje BASE64, protože šest bitů reprezentuje čísla v rozsahu 0 až 63, čili 64 možných hodnot). Výsledná data můžeme bez problémů zapsat do textového souboru.

Firma Sun nabízí v sadě JDK třídy, které proces transformace zajišťují. Jde o třídy BASE64Encoder a BASE64Decoder z balíčku `sun.misc`. Vzhledem k tomu, že všechny třídy uložené v balíčku `sun.misc` mají tu vlastnost, že jejich existence v dalších verzích JDK není firmou Sun zaručena, je lepší při použití poskytovatele Bouncy Castle použít třídu Base64 z balíčku `org.bouncycastle.util.encoders.Base64`, která poskytuje metody encode() a decode() plnící stejnou funkci jako třídy v JDK.

Kapitola 6

Webovské technologie

6.1 Úvod

V této kapitole budou popsány vybrané webovské technologie. O webovských technologiích byly napsány tisíce knih, proto na tomto místě nebudu zacházet do přílišných detailů a zaměřím se pouze na technologie, které se bezprostředně dotýkají komunikace s Portálem veřejné správy. A tímto nemám na mysli technologie pro generování webových stránek, ale spíše technologie pro vzájemnou komunikaci aplikací a technologie pro zabezpečení přenosu dat.

6.2 World Wide Web

Vznik celosvětové internetové sítě (World Wide Web) je datován do let 1989-1990 a za autora je považován Tim Berners Lee. Pod tento název jsou zastřešeny základní webovské technologie mezi které patří hypertext, značkovací jazyky a architektura webu. Pod pojmem hypertext rozumíme nesekvenční propojení jednotlivých dokumentů, obrázků, animací, zvuků a dalších elementů. Značkovacím jazykem pro zobrazování dokumentů na WWW se stalo HTML. Základním znakem internetu je to, že je postaven na mnoha architekturách a přenos dat může probíhat pomocí různých technologií. Na aplikační vrstvě se však většinou setkáme s protokolem pro přenos hypertextových či hypermediálních dokumentů HTTP.

6.3 Protokol HTTP

HTTP (HyperText Transfer Protocol) je internetový protokol, který byl původně určen pro výměnu hypertextových dokumentů ve formátu HTML. V současné době je tento protokol používán i pro přenos dalších informací. Pomocí rozšíření MIME je schopen podobně jako e-mail přenášet jakýkoliv soubor. Společně s technologií XML se používá pro webové služby (vzdálené volání procedur). Pro adresování objektů v internetu používá HTTP tzv. jednotný identifikátor prostředků URL (Uniform Resource Locator), který jednoznačně identifikuje jakýkoliv zdroj nacházející se na internetu. HTTP je textový a bezestavový protokol. Prakticky žádným způsobem neřeší zabezpečení přenášených dat.

6.4 Protokol HTTPS

HTTPS je nadstavbou protokolu HTTP, která poskytuje zvýšenou bezpečnost před odposloucháváním či podvržením dat. HTTPS není samostatný protokol, data jsou přenášena pomocí HTTP, ale jsou narozdíl od HTTP šifrována pomocí SSL nebo TSL, což zaručuje ochranu proti různým typům útoků. Implicitním portem pro komunikaci pomocí HTTPS protokolu je port 443. V prohlížeči rozeznáme protokol HTTPS pomocí URL začínající zkratkou https a většinou také ikonou zámku ve stavové liště.

Pro komunikaci pomocí HTTPS musí být na serveru nejdříve nainstalován certifikát. Takovýto certifikát by měl být podepsán certifikační autoritou, abychom si mohli být jisti identitou serveru. V některých případech si ale organizace mohou vytvořit vlastní certifikační autoritu a vydávat si své vlastní certifikáty. Problémem takových certifikátů je, že webový prohlížeč v takovém případě varuje uživatele o nedůvěryhodnosti podpisu. Tento problém je možné obejít instalací kořenového certifikátu vlastní certifikační autority na každém počítači, ze kterého budeme přistupovat na zabezpečené stránky. Nutno podotknout, že protokol HTTPS zabezpečuje pouze komunikaci na cestě od klienta k serveru a naopak. Další zabezpečení je otázkou jiných postupů uvedených v kapitole o počítačové bezpečnosti.

6.5 SSL

SSL je zkratka pro Secure Sockets Layer. Jedná se o protokol vložený mezi transportní (např. TCP/IP) a aplikační (např. HTTP) vrstvu. Poskytuje zabezpečení komunikace šif-

rováním a autentifikací komunikujících stran. Protokol SSL se nejčastěji využívá pro běžnou komunikaci s webovými servery pomocí HTTPS, který byl popsán v předcházejícím textu. Po vytvoření SSL spojení je komunikace mezi serverem a klientem šifrována.

Ustavení spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý. Veřejný klíč je možné zveřejnit a pokud tímto klíčem kdokoliv zašifruje nějakou zprávu, je zajištěno, že ji bude moci dešifrovat pouze majitel párového soukromého klíče (viz asymetrické šifrování v kapitole o počítačové bezpečnosti).

Ustavení spojení (SSL handshake)

1. Klient pošle serveru požadavek na SSL spojení spolu s doplňujícími informacemi (verze SSL, nastavení šifrování, atd.)

2. Server odešle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.

3. Podle přijatého certifikátu si klient ověří autenticitu serveru. Certifikát také obsahuje veřejný klíč serveru.

4. Na základě přijatých informací vygeneruje klient základ šifrovacího klíče, kterým se bude kódovat následující komunikace. Ten zašifruje veřejným klíčem serveru a pošle mu ho zpět.

5. Server použije svůj soukromý klíč k dešifrování základu šifrovacího klíče. Z tohoto základu vygenerují klient i server šifrovací klíč.

6. Klient a server si vzájemně potvrdí, že od této chvíle bude komunikace šifrována tímto klíčem. Fáze SSL handshake je u konce a klient se serverem mohou bezpečně komunikovat pomocí vygenerovaného šifrovacího klíče.

TLS (Transport Layer Security) je nástupcem protokolu SSL. TLS 1.0 přímo vychází z verze 3.0 protokolu SSL.

Kapitola 7

Portál veřejné správy

7.1 Úvod

Dne 1. 1. 2003 bylo zřízeno Ministerstvo informatiky České republiky jako ústřední orgán státní správy pro informační a komunikační technologie, telekomunikace a poštovní služby. Na Ministerstvo informatiky přešly v plném rozsahu kompetence Úřadu pro veřejné informační systémy, úseku spojů Ministerstva dopravy a pravomoc v oblasti elektronického podpisu Úřadu pro ochranu osobních údajů.

Jedním ze stěžejních projektů Ministerstva informatiky v době jeho vzniku bylo spuštění Portálu veřejné správy (PVS). Jedná se o službu pro občany a organizace, která si klade za cíl vybudovat elektronickou veřejnou správu a působit jako brána pro elektronický styk se všemi úřady veřejné správy.

Portál veřejné správy se skládá ze dvou částí:

Informační část - klade si za cíl poskytovat občanům a organizacím veškeré potřebné informace týkající se státní správy na adrese http://portal.gov.cz

Transakční část - je tvořena aplikací Elektronická podání. Jedná se o sjednocení elektronické komunikace s orgány státní správy.

Portál je v ostrém provozu od října 2003.

7.2 Informační část PVS

7.2.1 Technické řešení

Subdodavatelem informační části Portálu veřejné správy je společnost IBM. Řešení je postaveno na třívrstvé architektuře, přičemž jednotlivé vrstvy jsou oddělené firewally a doplněné o systémy detekce narušení (Intrusion Detection Systems). Celé řešení se skládá z následujících SW produktů: WebSphere Portal Server for Multiplatforms, WebSphere Application Server, WS Personalization, WS Transcoding, Tivoli Web Site Analyzer (Tivoli Web Services Analyzer, WebSphere Site Analyzer), WebSphere Portal Content Publisher, IBM DB2, IBM Directory Server, Lotus Domino Application Server a Tivoli Storage Manager. Všechny aplikace běží v ostrém a testovacím prostředí na operačním systému Red Hat Linux.

Základní software prezentační části Portálu veřejné správy je WebSphere Portal Server for Multiplatforms. Tento systém umožňuje doručovat informace pomocí portletů nezávisle na zařízení a síti. Podporuje řadu zařízení od mobilních telefonů přes PDA a MDA až k webovým prohlížečům.

Celé řešení je samozřejmě otevřené komunikaci s aplikacemi třetích stran přes aplikační rozhraní, JCA konektory, integrační službu JEE (JMS, EJB, JDBC), přes XML (XML parsing a XSL transformation) a webové služby (SOAP, UDDI, WSRP). Integrace prezentační části (IBM-Linux) a transakční části (Microsoft-Windows) je prováděna pomocí XML a webových služeb.

7.2.2 Obsah a vzhled

Portál veřejné správy je zde pro občany, podnikatelské subjekty, orgány a organizace státní správy. Směrem k těmto subjektům je namířen obsah a vzhled prezentační části PVS. Grafické provedení PVS je velmi jednoduché, na každém kroku je zde patrné, že portál je zaměřen na poskytování informací. Po stránce poskytování informací slouží PVS zcela bezchybně. Informace jsou rozděleny na hlavní stránku Úvod a čtyři moduly Adresář, Zákony, Životní situace a Podání.

Obrázek 7.1: Portál veřejné správy

Úvodní strana

Na úvodní stránce informační části PVS se nachází základní informace o České republice, prezidentovi, vládě, ministerstvech a parlamentu. V neposlední řadě je zde sekce věnovaná Evropské unii. Konkrétní informace jsou často umístěny na stránkách jednotlivých institucí, ale PVS zde funguje jako rozcestník a musím říci, že v době psaní této práce byly všechny odkazy na PVS plně funkční.

Na úvodní stránce je dále možné nalézt užitečné odkazy (povodňové zpravodajství, výpis z katastru nemovitostí, mapu České republiky i se satelitními snímky a mnoho dalších). Nalezneme tu také sekci Oblasti státní správy a další aktuální či zajímavé odkazy (v době psaní této práce například odkazy na e-learningové kurzy a rozšiřování vysokorychlostního internetu v České republice).

Dále se na úvodní stránce nacházejí tři nepřehlédnutelné odkazy: Občan, Podnikatel a Cizinec. Za každým z těchto odkazů se skrývá stránka se sadou odkazů zaměřených na daný subjekt. V této souvislosti nutno podotknout, že PVS je k dispozici i v méně rozsáhlé anglické verzi. Úvodní stránka slouží také jako nástěnka, na které se objevují aktuální informace z veřejné správy, povinně zveřejňované informace a novinky na PVS.

Adresář

Obsahem tohoto modulu je adresář subjektů státní správy. V tomto adresáři je možné hledat podle regionů, podle druhu úřadu či podle činnosti úřadu.

Zákony

Tento modul nabízí všechny platné právní předpisy vydané ve Sbírce zákonů a to vždy v aktuálním znění nebo ve znění všech změn (novel) účinných právě v aktuální den. Tyto informace jsou přebírány ze systému ASPI (automatizovaného systému právních informací).

Životní situace

Modul zpracovávají jednotlivé resorty státní správy a obsahuje, jak název napovídá, popis běžných životních situací vztahujících se ke státní správě. Tyto situace jsou v sekcích pro jednotlivé subjekty občan, podnikatel a cizinec. Občan zde například nalezne popis řešení situací týkajících se změny trvalého bydliště či výměny občanského průkazu, podnikatel

zase řešení situací týkajících se bezpečnosti práce či nemocenského pojištění. Cizinec zde nalezne například řešení vstupních formalit České republiky či nabytí českého státního občanství.

Podání

Tento modul slouží pro registraci uživatelů, kteří chtějí komunikovat elektronicky s úřady státní správy. Po dokončení registrace může uživatel zasílat a přijímat formuláře z úřadů veřejné správy s využitím identifikátoru uživatele nebo s využitím digitálního certifikátu (bude spuštěno v druhé fázi implementace PVS).

7.2.3 Hodnocení

Pokud mám hodnotit obsahovou stránku Portálu veřejné správy, tak jí není prakticky co vytknout. Posun v oblasti informačního obsahu je neustálý díky připojování dalších úřadů státní správy. Potřebné informace si zde najde občan i podnikatel. Díky soustředění na jednom místě si není třeba pamatovat desítky webových adres, ale stačí jediná.

Výrazné změny ovšem zaznamenal samotný vzhled PVS. Od doby, kdy jsem PVS zhruba před rokem viděl poprvé, doznal značných změn. Snad k tomu přispěl i audit společnosti Dobrý web [20], který po stránce přístupnosti zhodnotil Portál Veřejné správy a jehož výtky se postupem času začaly odstraňovat. Mezi hlavní nedostatky patřila absence podtrhávání odkazů, příliš grafických prvků či absence vyhledávání na úvodní stránce. Tyto a další problémy byly odstraněny. Chyby, které do současné doby nebyly odstraněny, jsou například URL adresy neodpovídající standardům přístupnosti nebo to, že stránky často nejsou validní (to jde ale u portálu, do kterého mohou samostatně přispívat jednotlivé orgány státní správy, stěží pokaždé zaručit).

7.3 Transakční část PVS

7.3.1 Technické řešení

Subdodavatelem transakční části Portálu veřejné správy je firma Microsoft ve spolupráci s firmou Siemens Bussiness Services. Řešení zahrnuje standardní produkty firmy Microsoft: BizTalk Server, Microsoft SQL Server, Commerce Server, Microsoft Operations Manager

a ISA Server. Všechny aplikace běží na operačním systému Microsoft Windows 2000 Server.

Základními produkty, na kterých je postavena transakční část Portálu veřejné správy jsou Microsoft BizTalk Server a Microsoft SQL Server, nad nimiž byla vyvinuta speciální aplikace Government Gateway tvořící uživatelské rozhraní a funkční logiku systému skládající se ze tří částí:

- Modul registrace uživatelů (R&A) je nejsložitější součástí celého systému a nejvíce závisí na speciálně vyvinutém kódu a na databázi určené k ukládání dat o uživatelích.
- Transakční jádro je v podstatě standardní implementací BizTalk serveru, který představuje transportní mechanismus pro zprávy putující od uživatelů k úřadům přes modul DIS (viz dále). Tato implementace je doplněna o jednoduchou aplikaci zajišťující využití informací z databáze uživatelů k autentifikaci zpráv a k jejich správnému směrování.
- Servery rozhraní úřadu/resortu (DIS) jsou rovněž v podstatě standardní implementací BizTalk serveru. Byly ovšem doplněny o určité prvky zajišťující spolehlivé doručování a sledování zpráv dále do informačních systémů resortu/úřadu a zpětnou informaci o stavu zpracování těchto zpráv.

7.3.2 Registrace

Registraci k Portálu veřejné správy provádí modul R&A (Registration & Enrollment), který zároveň slouží pro ověřování uživatelů ve všech následujících transakcích. Registrace slouží pro vytvoření jednotné identity pro komunikaci s orgány veřejné správy. Rozlišujeme tři typy registrací: občan, organizace a zástupci. Občan vystupuje v komunikaci s Portálem veřejné správy sám za sebe zatímco pod organizací či zástupci jsou uvedeni jednotliví uživatelé. Tito uživatelé mohou navíc mít ještě vlastní asistenty.

Uživatel je osoba, která zákonným způsobem zastupuje organizaci či zástupce. Tato osoba má v aplikaci Elektronická podání nejvyšší práva, může tedy vytvářet, upravovat a mazat své asistenty, vytvářet nové uživatele, přihlásit organizaci k elektronickým službám, odhlásit organizaci od elektronických služeb a samozřejmě podávat elektronická podání k přihlášeným elektronickým službám.

Asistent stejně jako uživatel reprezentuje organizaci či zástupce. Asistent může odesílat elektronická podání ke službám, které mu byly přiděleny uživatelem.

Registrace slouží k vytvoření uživatelského účtu v aplikaci elektronická podání. Pro komunikaci s PVS jsou potřeba následující bezpečnostní prvky: portálový uživatelský identifikátor, heslo a osobní certifikát. Pokud se chceme zaregistrovat na PVS, je zapotřebí vyplnit osobní údaje (celé jméno, e-mailovou adresu), zvolit si heslo, vybrat orgány státní správy, se kterými chceme komunikovat, a následně vyplnit údaje požadované vybranými orgány státní správy pro naší identifikaci. Například pro elektronická podání ELDP (Evidenční listy důchodového pojištění) na ČSSZ musíme zadat přidělené registrační číslo a variabilní symbol. Registrační proces proběhne v pořádku pouze v tom případě, že se nám podaří zapsat alespoň k jedné službě jakéhokoliv orgánu státní správy.

Přehled služeb pro registraci typu Občané

- Daňová správa - elektronická podání

Přehled služeb pro registraci typu Organizace

- ČSSZ - Nemocenské pojištění
- ČSSZ - Důchodové pojištění
- Úřad vlády - Informační systém o státní službě a platech
- Předložení odvolání stavebního úřadu na Středočeský kraj
- Daňová správa - elektronická podání
- Celní správa - Intrastat - elektronická podání
- Roční výkaz o poštovních službách
- ČSSZ - Přehled o příjmech a výdajích OSVČ
- Ministerstvo dopravy - eTesty - elektronická podání
- Ministerstvo informatiky České republiky - E-fakturace

Přehled služeb pro registraci typu Zástupci

- ČSSZ - Důchodové pojištění
- ČSSZ - Nemocenské pojištění
- Daňová správa - elektronická podání
- Celní správa - Intrastat - elektronická podání
- ČSSZ - Přehled o příjmech a výdajích OSVČ
- Ministerstvo informatiky České republiky - E-fakturace

7.3.3 Registrace k testovací větvi

Předně musím upozornit, že veškerá práce vývojáře s Portálem veřejné správy spočívá na komunikaci s testovacím serverem (https://bezpecne.dev.gov.cz). Z pochopitelných důvodů jsem podání na ostrou větev Portálu veřejné správy (https://bezpecne.podani.gov.cz) nikdy neuskutečnil. Nicméně obě větve by měly být naprosto totožné s tím rozdílem, že podání prováděná na ostrou větev jsou dále zpracovávána a evidována.

V době, kdy jsem s komunikací s PVS začínal, byly k dispozici pouze dvě služby transakční části PVS - elektronické odevzdávání evidenčních listů důchodového pojištění (ELDP) a elektronické odevzdávání přihlášek a odhlášek k sociálnímu pojištění (PRIHL). Tyto služby provozuje Česká správa sociálního zabezpečení (ČSSZ). Mé první e-maily tedy byly adresovány ČSSZ, kde mi byly poskytnuty fiktivní údaje (registrační číslo a variabilní symbol) pro přihlášení na testovací větev Portálu veřejné správy a můj osobní certifikát vydaný certifikační autoritou ČSSZ pro podepisování zpráv.

Ke službě Elektronická podání je možné se přihlásit jako občan, organizace či zástupce. Na základě registračního čísla a variabilního symbolu jsem si nechal vygenerovat portálový uživatelský identifikátor (identifikační číslo PVS) pro každý z výše uvedených subjektů. Pomocí identifikátoru a hesla je možné přistupovat k Portálu veřejné správy přes webové rozhraní a spravovat svá nastavení (přihlášení ke službám či osobní údaje). Pro elektronická podání pomocí aplikací se používá identifikátor, heslo, osobní certifikát, certifikát úřadu a identifikační prvek (prvky) orgánu státní správy (např. v případě ČSSZ se jedná o variabilní symbol).

7.3.4 Přihlašování

Přihlašování na PVS se provádí na stránce https://bezpecne.podani.gov.cz (ostrá větev PVS) či https://bezpecne.dev.gov.cz (testovací větev PVS). K přihlášení použijeme uživatelský identifikátor a heslo. Následně můžeme dle své role měnit různá nastavení účtu.

7.3.5 Registrace a přihlašování pomocí certifikátu

Způsob přihlašování k transakční větvi PVS pomocí certifikátu byl v době dokončování práce ve stádiu testování. V ostré větvi by měl být dostupný od 1. 7. 2006 s tím, že klasická možnost přihlašování pomocí identifikátoru a hesla samozřejmě zůstane i nadále v provozu.

7.4 Komunikace s PVS

Tradičním způsobem komunikace s orgány státní správy jsou tištěné dokumenty, které je nutné osobně doručit na daný úřad. Postupem času zavedly některé úřady možnost podávat dokumenty na elektronických médiích, případně pomocí vlastní uzpůsobené webové aplikace.

Příchod Portálu veřejné správy přináší dvě nové možnosti způsobu komunikace s orgány státní správy. Prvním z nich je vytvoření dokumentu v elektronické podobě pomocí aplikace umožňující tvorbu formulářů (např. Microsoft Excel či 602XML) a následné odeslání dokumentu na příslušný orgán pomocí PVS. Tento způsob je vhodný například pro fyzické osoby, které vytvářejí tyto dokumenty „v ruce“.

Většina firem má však potřebné údaje uvedeny ve svém účetnictví, takže je výhodné je využít. V takovém případě přichází na řadu programy, které načítají data vytvořená účetními systémy a zasílají je ke zpracování na PVS. Tímto způsobem komunikace s PVS se budeme dále zabývat, neboť takový program je součástí praktické části této diplomové práce.

7.4.1 Odeslání podání

Jednotlivé kroky, které provádí komunikační program při zasílání podání na PVS:

- Načtení podání v podobě XML souboru
- Vytvoření GovTalk obálky (zpráva SUBMISSION_REQUEST)

- Podepsání podání (dle standardu PKCS#7 - SignedData)
- Zašifrování podání (dle standardu PKCS#7 - EnvelopedData)
- Dokončení tvorby zprávy (spojení GovTalk obálky s podpisem, příp. šifrou)
- Navázání spojení se serverem na adrese https://bezpecne.podani.gov.cz/submission (ostrá větev) nebo https://bezpecne.dev.gov.cz/submission (testovací větev) pomocí protokolu HTTPS
- Odeslání zprávy metodou POST (HTTP 1.1)
- Zpracování odpovědi od PVS

Program zpracuje odpověď od PVS a zjistí, zda bylo portálem přijato a přeposláno dále na server příslušného orgánu státní správy (DIS) nebo zda došlo k chybě již na PVS (například chybná struktura GovTalk obálky či neplatné přihlašovací údaje). V případě úspěšného přeposlání na server příslušného orgánu státní správy je v odpovědi vráceno CorrelationID, které slouží jako jedinečný identifikátor zprávy pro pozdější dotazy.

Pro podepsání podání se používají certifikáty vydané kvalifikovanou certifikační autoritou (1.CA, PostSignum a eIdentity). Pro elektronická podání služeb ELDP (evidenční listy důchodového pojištění) a PRIHL (přihlášky a odhlášky nemocenského pojištění) existuje ještě možnost využití podpisového klíče ČSSZ.

Podpisové klíče ČSSZ mají historický význam a sloužily vývojářům při spuštění prvních služeb PVS. Do 1. 2. 2006 byly vydávány certifikační autoritou ČSSZ, která samozřejmě není kvalifikovaná, a proto je nutné podání provedená s pomocí těchto klíčů do tří dnů doplnit o doručení papírového formuláře na OSSZ, kde podepíšete, že jste provedli elektronické podání na ČSSZ. Pro testovací účely však byly podpisové klíče naprosto ideální.

Pod pojmem GovTalk obálka rozumíme schéma XML souboru, které definuje elementy potřebné pro komunikaci s aplikací elektronická podání. Před vložením do GovTalk obálky jsou data (zašifrovaná, podepsaná) najdříve převedena na kódování BASE64. GovTalk obálka je tvořena ze dvou částí. První částí je obálka PVS, která je zpracována už na PVS. Druhou částí je obálka služby, která je zprácována až DIS serverem orgánu státní správy, ke kterému daná služba přísluší.

V příloze diplomové práce se nacházejí ukázky XML zpráv pro komunikaci s PVS. Prvním příkladem je ukázka GovTalk obálky pro PVS [A.1]. Dále následuje GovTalk obálka pro službu orgánu státní správy (konkrétně se jedná o službu ČSSZ/RELDP) [A.2]. Obě obálky s doplněnou podepsanou a zašifrovanou zprávou tvoří požadavek zasílaný na PVS [A.3]. Po odeslání požadavku je vrácena zpráva o úspěšném podání [A.4] či o chybě při podání [A.5].

7.4.2 Kontrola stavu podání

Pokud se podání nezdařilo, je vrácena chybová zpráva. V takovém případě se musíme pokusit opravit údaje v obálce pro PVS a pokusit se podání uskutečnit znovu. Pokud se podání zdařilo, vrací se nám v potvrzující zprávě CorrelationID, kterým je jednoznačně identifikováno naše podání. S tímto údajem budeme dále pracovat. Naše podání nyní tedy prošlo úspěšně odbavením na PVS a bylo předáno na příslušný orgán státní správy. Tam bude čekat ve frontě na zpracování. Po zpracování bude na PVS vrácena zpráva o tom, jak zpracování dopadlo. My tedy budeme v určitých časových intervalech (standardně 10 sekund, prakticky v libovolných intervalech dle zatížení serverů) klást dotazy na PVS. V nich budeme zjišťovat, zda naše podání už bylo na příslušném resortu zpracováno a pokud ano, tak s jakým výsledkem. Pro tvorbu této zprávy využijeme CorrelationID, které jednoznačně identifikuje naši zprávu. Dotazovací zpráva je zasílána na adresu https://bezpecne.podani.gov.cz/poll nebo https://bezpecne.dev.gov.cz/submission v případě testování.

V příloze diplomové práce se nachází příklad zprávy pro kontrolu stavu podání [A.6] a návratové zprávy z PVS - zpráva informující o stále probíhajícím zpracování [A.7], zpráva o úspěšném podání [A.8] a zpráva o zamítnutí podání [A.9].

7.4.3 Kontrola stavu všech podání

Aplikace elektronická podání nám umožňuje také vytvořit zprávu, ve které se budeme dotazovat na stav zpracování všech podání na PVS. Samozřejmě jde pouze o zprávy, které nebyly smazány automaticky (všechny zprávy starší 60 dnů) nebo pomocí dotazu DELETE_REQUEST (viz dále). Je dokonce možné si nechat zobrazit pouze zprávy zpracovávané v určitém časovém období.

V příloze diplomové práce se nachází příklad dotazu na stav všech podání [A.10] a ná-

sledná reakce na tuto zprávu [A.11].

7.4.4 Smazání podání

Posledním krokem komunikace s PVS je smazání záznamu o podání. Tento krok není povinný, systém automaticky maže podání, která jsou starší než 60 dnů. Na druhou stranu je doporučeno zprávy mazat, aby se zrychlila komunikace s PVS. Pro identifikaci zprávy opět použijeme CorrelationID.

V příloze diplomové práce se nachází příklad zprávy pro smazání podání [A.12] a návratové zprávy z PVS o úspěšném smazání [A.13].

7.5 Služby PVS

V tomto výčtu budou uvedeny služby, které jsou podporovány komunikačním programem, který je součástí této diplomové práce.

7.5.1 Evidenční listy důchodového pojištění

Podávání evidenčních listů důchodového pojištění (ELDP), někdy také pojmenované jako roční evidenční listy důchodového pojištění (RELDP), je nejstarší službou poskytovanou transakční částí PVS. Jedná se o službu České správy sociálního zabezpečení (ČSSZ). Pro podepisování dat je možné použít certifikát kvalifikované certifikační autority nebo podpisový klíč ČSSZ. Podpisové klíče ČSSZ sloužily pro odstartování komunikace prostřednictvím PVS. Jejich nevýhodou bylo to, že podání učiněná pomocí podpisového klíče musela být následně doplněna písemným prohlášením o provedení podání. Dále byly podpisové klíče používány tvůrci programů pro komunikaci s PVS prostřednictvím testovací větve PVS, kterým byly tyto klíče vydávány na požádání zdarma. Od 1. 2. 2006 přestala ČSSZ podpisové klíče vydávat.

Ukázka datové věty pro RELDP se nachází v příloze [A.14].

7.5.2 Přihlášky a odhlášky nemocenského pojištění

Přihlášky a odhlášky nemocenského pojištění (PRIHL) jsou druhou nejstarší službou poskytovanou transakční částí PVS. Jedná se o službu ČSSZ. Pro podepisování dat je možné

použít certifikát kvalifikované certifikační autority nebo podpisový klíč ČSSZ.

7.5.3 Přehled o příjmech a výdajích osob samostatně výdělečně činných

V současnosti nejmladší služba ČSSZ poskytovaná prostřednictvím PVS. Tato služba již nepodporuje podpisové klíče ČSSZ, takže je potřeba použít certifikát kvalifikované certifikační autority.

7.5.4 Ministerstvo Financí - daňová správa

Služby daňové správy Ministerstva financí poskytované pomocí PVS jsou mladší než služby ČSSZ. Jsou také k nevoli mnoha programátorů komunikačních programů postaveny na trochu jiných principech. Pro komunikaci s Daňovou správou pomocí Portálu veřejné správy je nutno použít zaručený elektronický podpis. Z toho vyplývá, že budete potřebovat kvalifikovaný certifikát vydaný akreditovanou certifikační autoritou (a to i v případě testovací větve PVS). Princip tvorby podání je oproti ČSSZ odlišný. V případě ČSSZ jsou do výsledného podání dodávána zvlášť zašifrovaná data a zvlášť podepsaná data. V případě MFDS jsou data nejdříve podepsána, následně zašifrována a vložena do výsledného podání. Samotné podání dle standardu GovTalk je také odlišné od podání na ČSSZ. V případě ČSSZ mělo podání pro každou službu svou vlastní hlavičku podání, zatímco v případě MFDS je libovolný typ akceptované služby odesílán pod shodnou hlavičkou MF_DS_EDP.

Kapitola 8

Implementace programu pro komunikaci s PVS

V této kapitole se budeme zabývat implementací programu pro komunikaci s Portálem veřejné správy. Program byl podle požadavku zadavatele diplomové práce vytvořen v programovacím jazyce Java. Aplikace byla pojmenována jako WinPortal (složením názvu firmy WinStrom, pro kterou byl program vytvářen, a slova Portal označujícího Portál veřejné správy). V následujících odstavcích nejdříve prozkoumáme konkurenční aplikace a následně už se budeme zabývat implementací vlastní aplikace. Po implementaci se budeme zabývat také obsluhou programu. Na konci kapitoly provedeme zhodnocení aplikace a podíváme se na případné další možnosti rozšíření aplikace.

8.1 Srovnání konkurenčních programů

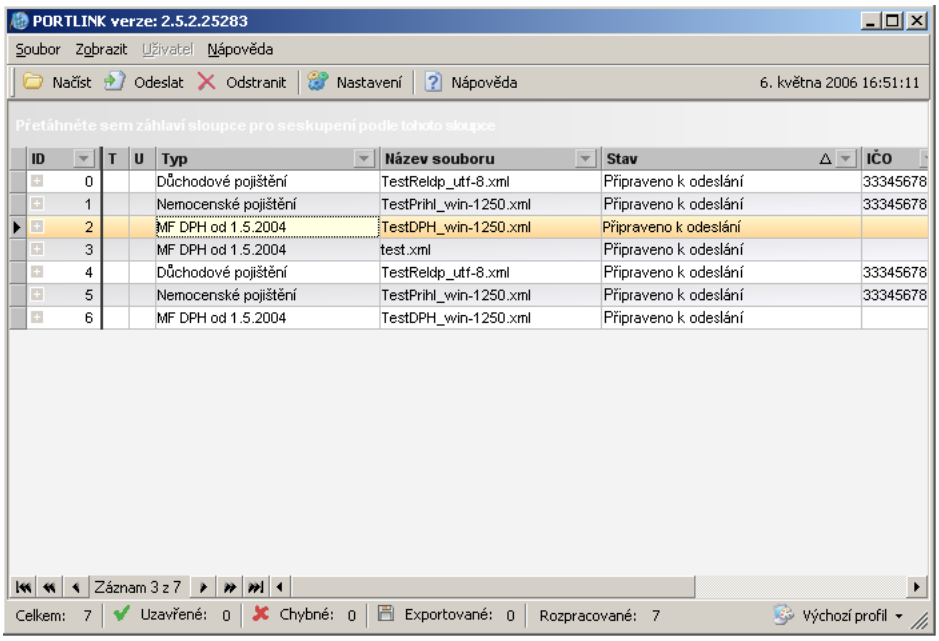
V této kapitole se podíváme na konkurenční aplikace používané pro komunikaci s Portálem veřejné správy. Zhodnocení konkurenčních aplikací bylo prvním krokem, který jsem provedl po zadání diplomové práce. Srovnání se ukázalo jako klíčové, neboť jsem si přesně mohl definovat, jakou funkčnost by mnou vytvářený program měl poskytovat a která se mi jeví jako zbytečná. Pro tento základní přehled jsem vybral následující tři programy: PortLink, MRP Elektronická Podání a Alis-PVS. První program je prakticky referenčním programem pro komunikaci s PVS. Původně byl vytvořen pod názvem eReldp, po přidání dalších služeb změnil název na PortLink. Dalším zástupcem je aplikace MRP Elektronická podání, která patří do absolutní špičky komunikačních programů s PVS. Třetím příkladem je aplikace

Alis-PVS, která zastupuje aplikace vytvořené v programovacím jazyce Java.

PortLink

Verze: 2.5.2
Výrobce: NZServis
Odkaz: http://www.nzservis.cz/www/portlink/
Platforma: MS .NET

Podporované služby:
ČSSZ: RELDP, PRIHL
MFDS: DPH, DPF, DPP, VIES, DSL, DNE, RHL, DAD



Obrázek 8.1: PortLink

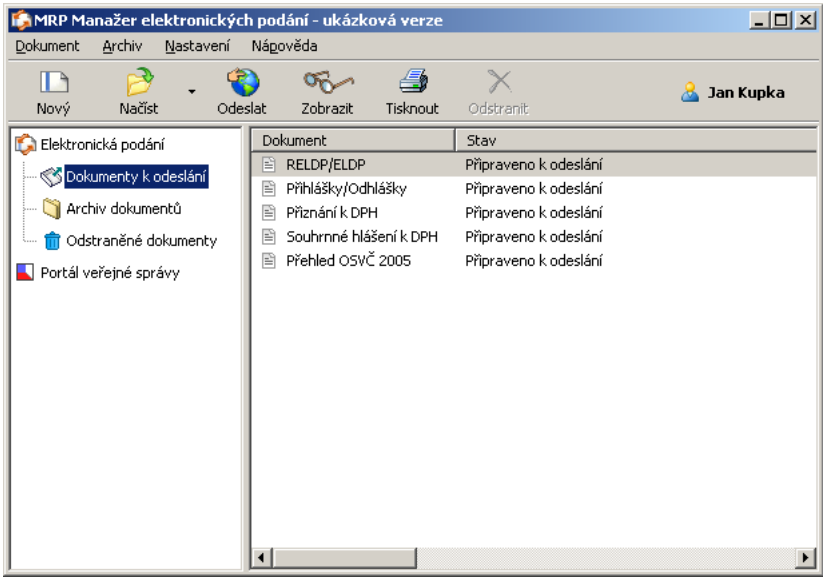
Aplikace PortLink, dříve známá pod názvem eReldp, je považována prakticky za referenční aplikaci pro komunikaci s Portálem veřejné správy. Mezi její přednosti patří kvalitní náhledy XML souborů a podpora velkého množství služeb. Na stránkách aplikace se nachází diskuzní fórum, které je určené pro zákazníky užívající program PortLink, ale před

vytvořením diskuzního fóra na PVS se zde dalo nalézt nejvíce informací o PVS. Mezi zápory programu patří příliš „tabulkovitý“ vzhled hlavního okna.

MRP Manažer elektronických podání

Verze: 3.30.033
Výrobce: MRP Electronics Software Informatics
Odkaz: http://www.mrp.cz/software/elpodani/main.asp

Podporované služby:
ČSSZ: RELDP, PRIHL, OSVC
MFDS: DPH, VIES



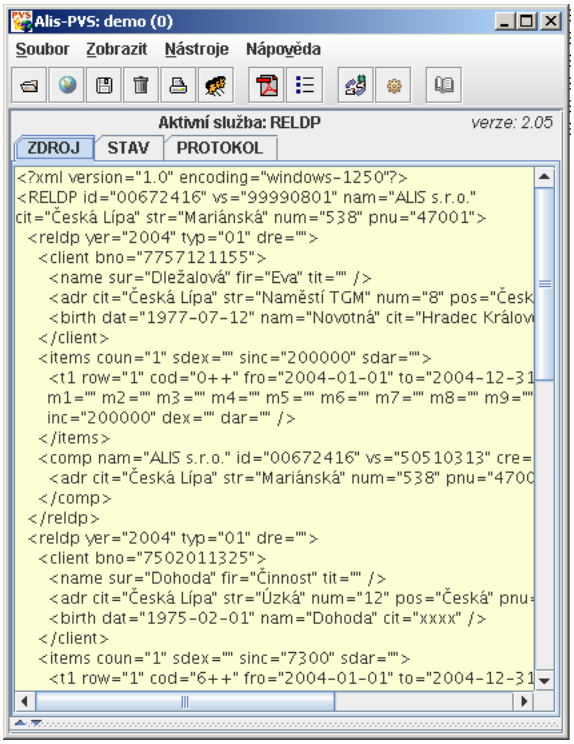
Obrázek 8.2: MRP Elektronická podání

Aplikace MRP Elektronická podání má kvalitně vytvořené uživatelské prostředí včetně náhledů XML souborů. V poslední verzi přidává možnost odesílání služby ČSSZ/OSVČ včetně možnosti vyplnění formuláře přímo v aplikaci.

Alis-PVS

Verze: 2.05
Výrobce: Alis s.r.o.
Odkaz: http://www.alis.cz/pvs/pvs.jsp
Platforma: Java

Podporované služby:
ČSSZ: RELDP, PRIHL
MFDS: DPH
Předložení Odvolání stavebního úřadu na Středočeský kraj



Obrázek 8.3: Alis-PVS

Aplikace Alis-PVS je stejně jako mnou vyvíjená aplikace napsána v Javě a využívá kryptografickou knihovnu Bouncy Castle. Mezi hlavní výhody aplikace patří přenositelnost na jiné operační systémy a možnost exportovat XML dokument do PDF souboru. Mezi hlavní zápory aplikace patří nepříliš dobře provedené grafické rozhraní pracující pouze

s jedním načteným podáním a náhled XML dokumentu umožněný jen transformací do PDF souboru.

8.2 Implementace aplikace

Aplikace je rozdělena do jedenácti balíků, které jsou tvořeny třídami s podobnou funkčností.

Seznam balíků aplikace

- cz.winstrom.winportal.data
- cz.winstrom.winportal.gui
- cz.winstrom.winportal.hlavni
- cz.winstrom.winportal.io
- cz.winstrom.winportal.komunikace
- cz.winstrom.winportal.konstanty
- cz.winstrom.winportal.kryptografie
- cz.winstrom.winportal.misc
- cz.winstrom.winportal.sablony
- cz.winstrom.winportal.vlakna
- cz.winstrom.winportal.xml

Balík data

Balík data obsahuje třídy reprezentující jednotlivé objekty v systému. Do tohoto balíku spadají třídy Instituce, Klic, Nastaveni a Zaznam. Třída Instituce reprezentuje instituci neboli orgán státní správy (např. ČSSZ, MFDS). Třída uchovává tři typy názvů služby, adresu souboru s šifrou pro danou službu a odkaz na seznam klíčů pro danou službu. Třída Klic reprezentuje jeden klíč dané instituce. Klíč je definován jako dvojice název-hodnota a příklady klíčů jsou například variabilní symbol (ČSSZ), datum narození (MFDS) či DIČ

(MFDS). Třída Nastaveni uchovává nastavení programu a třída Zaznam reprezentuje jeden záznam (podání) pro PVS.

Balík gui

Obsahem balíku gui jsou třídy reprezentující grafické uživatelské rozhraní aplikace. Jsou zde jednotlivé dialogy, hlavní formulář, filtry pro dialogy otevírání souborů, třída pro zobrazení úvodní načítací obrazovky (splash screen) a třída reprezentující datovou část tabulky.

Balík hlavni

Balik hlavni obsahuje jedinou stejně pojmenovanou třídu. Tato třída obsahuje metodu main() pro spuštění aplikace. Během spouštění zobrazí splash screen a zinicializuje data. Poté už předá řízení hlavnímu formuláři aplikace.

Balík io

Obsahem balíku io jsou třídy pro záznam průběhu komunikace s PVS. Třída Log zaznamenává do souboru !log.txt v adresáři log údaje o odeslání podání, přijetí či zamítnutí na PVS a o následném přijetí či odmítnutí podání daným orgánem státní správy. Zároveň třída Log ukládá obsah každého odesílaného podání (datovou větu) do adresáře log. Tento způsob záznamu není možné vypnout. Třída Trace naproti tomu ukládá každou zprávu odeslanou na PVS a každou zprávu přijatou z PVS. Tento způsob záznamu průběhu komunikace slouží převážně k testování aplikace a je možné ho programově vypnout v dialogu Nastavení.

Balík komunikace

Balík komunikace obsahuje stejnojmennou třídu, která zprostředkovává komunikaci aplikace s PVS pomocí HTTPS protokolu.

Balík konstanty

Obsahem balíku konstanty jsou třídy Adresy a Sluzby. Ve třídě Adresy jsou uloženy adresy ostré a testovací větve PVS. Ve třídě Sluzby jsou uloženy nastavení jednotlivých služeb, se kterými umí program komunikovat.

Balík kryptografie

Balík kryptografie obsahuje třídy MD5, PBE, Podpis a Sifrovani. Třída MD5 slouží pro vytvoření MD5 hashe. Třída PBE je určena k zašifrování a dešifrování hesla pro přístup k PVS. Třídy Podpis a Sifrovani slouží k podepsání a zašifrování datové věty při přípravě elektronického podání.

Balík misc

Obsahem balíku misc je jediná třída SwingWorker, která umožňuje vypisování zpráv o výsledcích pracujícího vlákna do grafického uživatelského rozhraní Swing.

Balík sablony

Balík sablony obsahuje třídy PVS, CSSZ a MFDS vytvářející zprávy pro komunikaci s PVS.

Balík vlakna

Obsahem balíku vlakna jsou třídy (reprezentující vlákna) provádějící časově náročné činnosti. Mezi takové operace patří komunikace s PVS (odesílání, ověřování a mazání podání), ukládání podání do souboru, instalace Java Cryptography Extension a instalace knihoven Bouncy Castle.

Balík xml

Balík xml obsahuje třídy pro práci s XML dokumenty. Práce s XML dokumenty je založena na modelu DOM a jazyku XPATH. Jádro balíku tedy tvoří třída DOM a pomocné třídy (ChybyParseru, PrazdnyText). Další třídy slouží pro načítání a ukládání dat z/do XML dokumentů. Třída XMLInstituce načítá a ukládá nastavení jednotlivých institucí. Třída XMLNastaveni slouží pro načítání a ukládání konfigurace programu. Třída XMLOdpovedi je určena pro načtení dat z odpovědi PVS. Třída XMLVeta načítá datovou větu z XML souboru a ukládá ji do objektu Zaznam. Třída XMLZaznam slouží pro načítání a ukládání všech záznamů v programu.

8.3 Struktura aplikace

Celá aplikace se skládá ze souboru winportal.jar (program) a pomocných adresářů. Soubor winportal.jar je spustitelný archiv jazyka Java. Aplikace byla vytvořena a testována v JDK 5.0 update 6. Pro spuštění je potřeba mít nainstalováno JRE 5.0 update 6 a vyšší. V případě nižších updatů nebyla funkčnost aplikace testována. V případě starších verzí JRE nebude aplikace fungovat.

Adresář bezpecnost obsahuje kryptografické knihovny Bouncy Castle (bcmail-jdk15-131.jar a bcprov-jdk15-131.jar) a soubory JCE pro rozšíření kryptografických možností Javy (local_policy.jar a US_export_policy.jar).

Adresář certifikaty obsahuje certifikáty pro šifrování a podepisování podání. Šifrovací certifikát pro ČSSZ se nachází v souboru cssz.cer. Šifrovací certifikát pro MFDS se nachází v souboru mfds.cer. Soubor uzivatel3446.pfx obsahuje testovací podpisový klíč vydaný pro účely testovacích podání pro ČSSZ.

Adresář data obsahuje ukázky datových vět sloužících pro testování aplikace.

Obsahem adresáře nastaveni jsou soubory s konfigurací programu. V souboru data.xml jsou uloženy jednotlivé záznamy (podání). V souboru instituce.xml se nacházejí nastavení pro jednotlivé instituce. V souboru nastaveni.xml je uložena konfigurace programu.

Adresář schemata obsahuje XSL soubory všech služeb dostupných v programu. Prostřednictvím XSL souborů jsou v programu zobrazovány náhledy podání.

Aplikace může dále obsahovat adresáře log a trace, které obsahují záznamy komunikace s PVS.

8.4 Instalace aplikace

Jak už bylo uvedeno, aplikace ke svému běhu potřebuje JRE 5.0 update 6 a vyšší. K úspěšnému běhu je zapotřebí ještě instalace knihoven Bouncy Castle a JCE pro rozšíření kryptografických možností Javy. Tuto instalaci můžeme provést nakopírováním souborů dle návodu v adresáři bezpecnost. Jednodušší možnost instalace knihoven se nachází v menu Nastavení po spuštění aplikace WinPortal. Po zvolení položky Instalace JCE budou všechny potřebné soubory nakopírovány do správných adresářů. Po restartu můžeme začít aplikaci plně využívat.

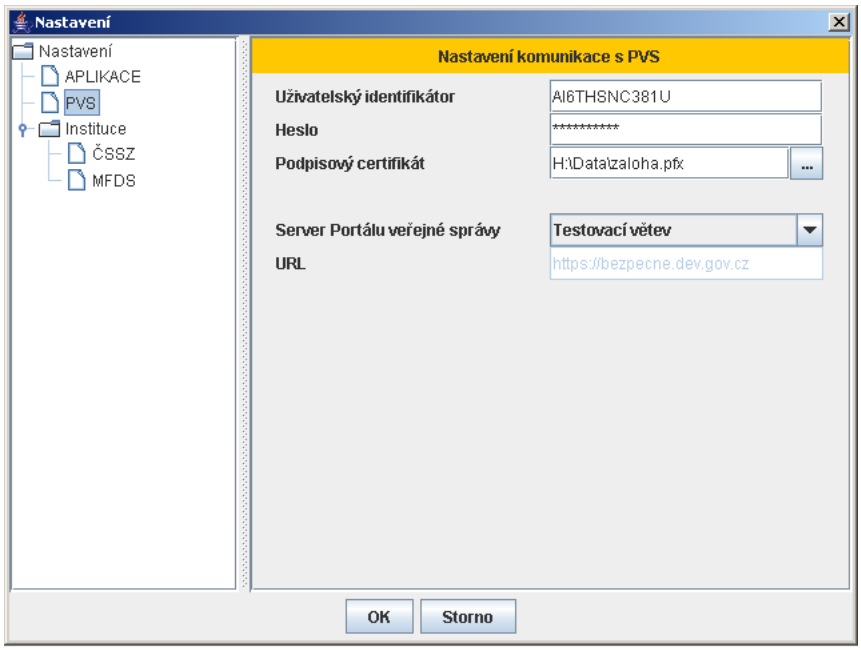
8.5 Obsluha aplikace

Po spuštění aplikace uvidíme pracovní plochu, kam budou přidávána jednotlivá podání. Nejdříve však musíme provést určitá nastavení, aby program mohl správně pracovat.

Obrázek 8.4: WinPortal - hlavní okno aplikace

Do nastavení programu se dostaneme pomocí menu (Nastavení - Nastavení) nebo stiskem tlačítka Nastavení na nástrojové liště. Jednotlivé panely s nastavením vybíráme v levé části dialogu.

Na panelu Aplikace najdeme možnost zapnutí nebo vypnutí záznamu odesílaných a přijímaných zpráv (TRACE), případně můžeme všechny zprávy vymazat (Vymazat TRACE). Důležitější je však panel PVS, na kterém nastavujeme údaje pro přístup k PVS. Zadáme vlastní uživatelský identifikátor a heslo. Dále vybereme podpisový certifikát. Pokud chceme uskutečnit testovací podání služeb RELDP a PRIHL, můžeme použít testovací podpisový klíč, který je součástí aplikace. Pro testovací podání služeb pro MFDS je zapotřebí podpisový certifikát vydaný kvalifikovanou certifikační autoritou. Stejně tak je samozřejmě zapotřebí kvalifikovaný certifikát pro všechna podání na ostrý server PVS. Poslední volbou panelu PVS je výběr testovací či ostré větve PVS. Další panely (ČSSZ a MFDS) představují nastavení pro jednotlivé instituce. Pro každou instituci musíme zvolit šifrovací certifikát (certifikáty standardně nalezneme v adresáři certifikaty v kořenovém adresáři aplikace). Každá instituce navíc požaduje vložení charakteristických údajů (jedná se o stejné údaje, které se vyplňují na PVS při přihlašování k jednotlivým službám). Pro služby ČSSZ se jedná o variabilní symbol. Pro MFDS je to buď datum narození (ve formátu dd/mm/yyyy) nebo daňové identifikační číslo.



Obrázek 8.5: WinPortal - Nastavení

Po uložení nastavení stiskem tlačítka OK se dostaneme zpět na hlavní pracovní plochu programu. Načtení podání provedeme pomocí menu (Podání - Načíst podání) nebo stiskem tlačítka Načíst podání na nástrojové liště. Po zvolení souboru bude datová věta načtena programem a přidána na pracovní plochu programu v podobě nového záznamu. Po výběru podání se nám zpřístupní nové možnosti v menu i v nástrojové liště.

Po stisku tlačítka Náhled podání otevřeme náhledový dialog, ve kterém bude zobrazen náhled našeho podání a případně si můžeme nechat zobrazit i celou datovou větu.

Další tlačítko, která je aktuálně přístupné, nese popisek Uložit podání. Po stisku tlačítka vybereme místo, kam chceme uložit podepsané a zašifrované podání. Následně zadáme heslo podpisového certifikátu a podání bude uloženo do souboru. Takové podání můžeme doručit na příslušný orgán státní správy na přenosném médiu.

Mnohem zajímavější funkci plní tlačítko Odeslat podání. Po zadání hesla podpisu je zahájeno předávání podání na PVS. Po jeho dokončení nám bude oznámeno, zda bylo podání na PVS úspěšně přijato či nikoliv. V případě úspěšného podání se nám odkryje tlačítko Zjistit stav podání, které umožňuje dotazování se na stav zpracování podání. V případě, že zpracování podání na příslušném DIS serveru orgánu státní správy ještě nebylo dokončeno, zobrazí se zpráva o probíhajícím zpracování. Pokud již zpracování bylo dokončeno, získáme

Obrázek 8.6: WinPortal - náhled podání

po stisku tohoto tlačítka oznámení o úspěšném či neúspěšném zpracování našeho podání. V případě neúspěchu se dozvíme popis chyby.

Po stisku tlačítka Smazat podání provedeme smazání úspěšného či neúspěšného podání. Stejně tak můžeme ze seznamu po stisku tohoto tlačítka smazat ještě neodeslané podání.

8.6 Shrnutí

Dostáváme se k hodnocení aplikace a částečně také k hodnocení transakční části Portálu veřejné správy. Není vůbec jednoduché shrnout do několika vět jeden rok strávený komunikací s Portálem veřejné správy.

Předně musím říci, že Portál veřejné správy zaznamenal během roku, kdy jsem se jím zabýval obrovský posun směrem vpřed. Vždyť v době, kdy jsem PVS viděl poprvé, tak na něm běžela pouze jediná služba. Brzy však následovala další a dnes už se počet služeb začíná počítat na desítky.

Tento růst PVS s sebou přinesl samozřejmě mnoho problémů. Počáteční přizpůsobení jedné službě znamenalo pro mnoho komunikačních programů při přidání nových služeb nutnost razantního přepracování. To se týkalo například programu eReldp, později známého

pod názvem PortLink. Tyto problémy se samozřejmě nevyhnuly ani aplikaci WinPortal.

Z těchto důvodů jsem byl nucen aplikaci WinPortal během vývoje několikrát upravovat. Původní plány na úplnou konfigurovatelnost aplikace pomocí XML souborů začaly brát rychle za své, neboť při přidání každé nové služby došlo ke změně, která nemohla být konfigurovatelná pouze XML souborem, ale muselo dojít k úpravě aplikace.

To ale zdaleka nebylo všechno. Velké problémy nastaly v té chvili, kdy své služby pomocí PVS začala nabízet Daňová správa Ministerstva financí. Komunikace se službami MFDS se totiž ukázala poměrně odlišná od služeb ČSSZ. Ať už jde o podepisování a šifrování, elementy datových vět nebo například umístění chybových hlášek, vše muselo být upraveno pro každou instituci zvlášť.

Bylo zajímavé pozorovat, s jakou rychlostí vytvářejí firmy nové verze svých produktů s novými službami. Zpočátku bylo získávání informací o nových službách velmi složité a dalo se získat pouze sledováním konkurenčních programů. S příchodem diskuzního fóra pro vývojáře [23] se začaly informace o nových službách objevovat i tam. Musím říci, že ač jsem u aplikace WinPortal neustále spíše doplňoval služby, které už konkurence nějakou dobu měla, například služba ČSSZ/OSVČ byla v aplikaci WinPortal úspěšně implementována dříve než byla v jakémkoliv konkurenčním programu. Ostatně v době psaní tohoto textu podporuje danou službu pouze aplikace MRP Elektronická podání.

Programů pro komunikaci s PVS je několik desítek. Firmy vytvářející účetní systémy brzy zjistily, že program pro komunikaci s PVS se pro ně může stát klíčovou konkurenční výhodou. Samozřejmě, že jejich implementace často nedosahují kvalit programů PortLink či MRP Elektronická podání. Avšak pokud spolehlivě pracují, není třeba zákazníkům v případě jejich zájmu o komunikaci s PVS doporučovat komunikační aplikace konkurenčních firem, ale výhodně jim nabídnout aplikaci vlastní. Taková aplikace může být navíc upravena speciálně pro práci s účetním systémem dané firmy. Je to výhodné také z toho důvodu, že firmám vyvíjejícím účetní systémy, nepřináší tvorba takové aplikace příliš mnoho starostí navíc. Důležité je ale neustálé sledování vývoje komunikace s PVS a přidávání nových funkcí a hlavně služeb orgánů státní správy.

Jednou z nejdůležitějších vlastností vyvíjené aplikace byla od počátečního návrhu snadná rozšiřitelnost o nové služby a nové instituce (orgány státní správy). Přidání nové služby pro stávající organizaci představuje jednoduchou úpravu programu a vytvoření nového transformačního souboru. Přidávání nové instituce do programu představuje poněkud složitější úpravu kódu, neboť každá instituce si klade vlastní speciální požadavky na vy-

tvořené podání. Pro seznámení se zdrojovým kódem aplikace je možné ze zdrojového kódu vygenerovat dokumentaci (JavaDoc). Problematické úseky kódu jsou navíc podrobně komentovány.

Kapitola 9

Shrnutí

9.1 Zhodnocení

Teoretická část diplomové práce přináší rozbor moderních technologií. Skutečným fenoménem dnešní doby je jazyk XML, který se stal standardem pro přenos dat přes internet. Včetně přidružených technologií se jedná o natolik rozsáhlé téma, že by vydalo na celou diplomovou práci. Minimum z jazyka XML obsažené v této diplomové práci by mělo dostačovat pro přiblížení komunikace s Portálem veřejné správy. Teoretická část o počítačové bezpečnosti seznamuje čtenáře se základními principy kryptografie. Zároveň je informuje o nejnovějších výzkumech jednotlivých kryptografických algoritmů. Na tuto obecnou část navazuje podkapitola o elektronickém podpisu, která obsahuje nejen teoretické poznatky, ale i praktické zkušenosti. Poslední kapitola teoretické části se zabývá webovskými technologiemi. Opět se jedná o poměrně rozsáhlé téma a z tohoto důvodu je seznam technologií zúžen na technologie potřebné pro komunikaci s Portálem veřejné správy.

Praktická část začíná seznámením s technologiemi použitými na Portálu veřejné správy. Detailně je popsána informační i transakční část PVS. Následuje seznámení s komunikačními principy pro odesílání podání. Dále jsou popsány vybrané služby, se kterými můžeme prostřednictvím Portálu veřejné správy komunikovat. Druhá kapitola praktické části popisuje aplikaci pro komunikaci s Portálem veřejné správy od zmapování konkurenčních aplikací přes architekturu aplikace a její obsluhu až po zhodnocení aplikace.

9.2 Závěr

Cílem diplomové práce bylo seznámit se s informační a transakční částí Portálu veřejné správy a s technologiemi využitelnými pro komunikaci s PVS. Získané znalosti byly využity při vývoji vlastní aplikace.

V teoretické části jsem se pokusil detailně přiblížit moderní technologie pro komunikaci s PVS. Vzhledem k rozsahu témat bylo třeba popsat pouze základy technologií a záležitosti, které se bezprostředně dotýkají komunikace s PVS. Pokusil jsem se také přiblížit praktické využití daných technologií.

Na teoretických základech byla postavena aplikace pro komunikaci s Portálem veřejné správy. Během svého vývoje tato aplikace postupně smazávala výhody konkurenčních aplikací. Samozřejmě nelze srovnávat aplikaci, která byla vytvořena jedním člověkem během necelého roku vývoje s aplikacemi, na jejichž vývoji pracují vývojářské týmy několik let. Konkurenční aplikace většinou vynikají propracovaným grafickým rozhraním a rozsáhlou podporou uživatelů. Na druhou stranu nejdůležitější částí aplikace je samotná komunikace s PVS a podpora základních služeb orgánů státní správy. V tomto směru podporuje mnou vyvinutá aplikace srovnatelný rozsah služeb jako konkurenční aplikace.

Vybral jsem si toto téma diplomové práce hlavně z toho důvodu, že se jedná o praktickou aplikaci nejmodernějších technologií, jejichž teoretické znalosti jsem získal ve škole. Přestože se jednalo o náročnou práci a získávání informací často nebylo jednoduché, jsem rád, že jsem si toto téma diplomové práce zvolil.

Příloha A

Komunikace s PVS

XML A.1: GovTalk obálka pro PVS

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>request</Qualifier>
      <Function>submit</Function>
      <TransactionID/>
      <AuditID/>
      <CorrelationID/>
      <ResponseEndPoint/>
      <Transformation>XML</Transformation>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>NH5PQJ743EHB</SenderID>
        <Authentication>
          <Method>MD5</Method>
          <Value>GRtBwY9qOkKW156P9VGKNQ==</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <GovTalkDetails>
    <Keys>
      <Key Type="vars">99200075</Key>
    </Keys>
  </GovTalkDetails>
</GovTalkMessage>
```

XML A.2: GovTalk obálka pro službu RELDP

```
<Body>
    <Message xmlns="http://www.cssz.cz/XMLSchema/emp/envelope" version="1.1">
      <Header>
        <Signature xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="bin.base64">
      </Signature>
        <Vendor productName="WinPortal" version="1.0"/>
      </Header>
      <Body xmlns:dt="urn:schemas-microsoft-com:datatypes" contentEncoding="raw"
    dt:dt="bin.base64" encrypted="yes"></Body>
    </Message>
</Body>
```

XML A.3: SUBMISSION REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>request</Qualifier>
      <Function>submit</Function>
      <TransactionID/>
      <AuditID/>
      <CorrelationID/>
      <ResponseEndPoint/>
      <Transformation>XML</Transformation>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>NH5PQJ743EHB</SenderID>
        <Authentication>
          <Method>MD5</Method>
          <Value>GRtBwY9qOkKW156P9VGKNQ==</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <GovTalkDetails>
    <Keys>
      <Key Type="vars">99200075</Key>
    </Keys>
  </GovTalkDetails>
  <Body>
    <Message xmlns="http://www.cssz.cz/XMLSchema/emp/envelope"
    version="1.1">
      <Header>
        <Signature xmlns:dt="urn:schemas-microsoft-com:datatypes"
      dt:dt="bin.base64">podpis_BASE64</Signature>
```

```
        <Vendor productName="WinPortal" version="1.0"/>
      </Header>
      <Body xmlns:dt="urn:schemas-microsoft-com:datatypes" contentEncoding="raw"
    dt:dt="bin.base64" encrypted="yes">sifra_BASE64</Body>
    </Message>
  </Body>
</GovTalkMessage>
```

XML A.4: SUBMISSION ACKNOWLEDGE

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>acknowledgement</Qualifier>
      <Function>submit</Function>
      <TransactionID/>
      <CorrelationID>83B1E47C927264F397F44066D9EDA2D8</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
      <GatewayTimestamp>2006-05-06T10:50:47.815</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails/>
  </Header>
  <GovTalkDetails>
    <Keys/>
  </GovTalkDetails>
  <Body>
    <Signature xmlns="http://www.podani.gov.cz/TxE/timestamp" Version="1.0">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <TimeStamp>
        <date>20060506</date>
        <time>10:50:47</time>
      </TimeStamp>
      <SignatureValue>elektronicka_znacka_BASE64</SignatureValue>
    </Signature>
  </Body>
</GovTalkMessage>
```

XML A.5: SUBMISSION ERROR

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>UndefinedClass</Class>
      <Qualifier>error</Qualifier>
```

```
      <Function>submit</Function>
      <TransactionID/>
      <CorrelationID>5FD9DB731A055CB7533D788FC151C9A4</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
      <GatewayTimestamp>2006-05-06T10:37:30.529</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails/>
  </Header>
  <GovTalkDetails>
    <Keys/>
    <GovTalkErrors>
      <Error>
        <RaisedBy>Gateway</RaisedBy>
        <Number>1046</Number>
        <Type>fatal</Type>
        <Text>Authentication Failure. The supplied user credentials failed
       validation for the requested service.</Text>
        <Location/>
      </Error>
    </GovTalkErrors>
  </GovTalkDetails>
  <Body/>
</GovTalkMessage>
```

XML A.6: POLL REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>poll</Qualifier>
      <Function>submit</Function>
      <TransactionID/>
      <AuditID/>
      <CorrelationID>69F4F2034F33E426495FD0042AA4B770</CorrelationID>
      <ResponseEndPoint/>
      <Transformation>XML</Transformation>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>NH5PQJ743EHB</SenderID>
        <Authentication>
          <Method>MD5</Method>
          <Value>GRtBwY9qOkKW156P9VGKNQ==</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
```

```
  </Header>
  <GovTalkDetails>
    <Keys>
      <Key Type="vars">99200075</Key>
    </Keys>
  </GovTalkDetails>
  <Body/>
</GovTalkMessage>
```

XML A.7: POLL ACKNOWLEDGE

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>acknowledgement</Qualifier>
      <Function>submit</Function>
      <TransactionID/>
      <CorrelationID>69F4F2034F33E426495FD0042AA4B770</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/poll
    </ResponseEndPoint>
      <GatewayTimestamp>2006-05-06T10:52:15.832</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails/>
  </Header>
  <GovTalkDetails>
    <Keys/>
  </GovTalkDetails>
  <Body>
    <Signature xmlns="http://www.podani.gov.cz/TxE/timestamp" Version="1.0">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <TimeStamp>
        <date>20060506</date>
        <time>10:52:10</time>
      </TimeStamp>
      <SignatureValue>elektronicka_znacka_BASE64</SignatureValue>
    </Signature>
  </Body>
</GovTalkMessage>
```

XML A.8: POLL RESPONSE

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>1.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
```

```
      <Qualifier>response</Qualifier>
      <Function>submit</Function>
      <CorrelationID>69F4F2034F33E426495FD0042AA4B770</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
      <Transformation>XML</Transformation>
      <TransactionID/>
      <GatewayTimestamp>2006-05-06T10:54:11.368</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>0000000132447206</SenderID>
        <Authentication>
          <Method>clear</Method>
          <Role/>
          <Value>**********</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <GovTalkDetails>
    <Keys>
      <Key Type="SpokeName">CSSZ_2_ORG</Key>
    </Keys>
  </GovTalkDetails>
  <Body>
    <Signature xmlns="http://www.cssz.cz/emp/timestamp" Version="1.0">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <TimeStamp>
        <date>20060506</date>
        <time>13:07:22</time>
      </TimeStamp>
      <SignatureValue>elektronicka_znacka_BASE64</SignatureValue>
    </Signature>
  </Body>
</GovTalkMessage>
```

XML A.9: POLL ERROR

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>1.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>error</Qualifier>
      <Function>submit</Function>
      <CorrelationID>6AABC1432549B28CD1C23986D2201E38</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
```

```
    <Transformation>XML</Transformation>
    <TransactionID/>
    <GatewayTimestamp>2006-05-06T11:07:14.083</GatewayTimestamp>
  </MessageDetails>
  <SenderDetails>
    <IDAuthentication>
      <SenderID>0000000132447206</SenderID>
      <Authentication>
        <Method>clear</Method>
        <Role/>
        <Value>**********</Value>
      </Authentication>
    </IDAuthentication>
  </SenderDetails>
</Header>
<GovTalkDetails>
  <Keys>
    <Key Type="SpokeName">CSSZ_2_ORG</Key>
  </Keys>
  <GovTalkErrors>
    <Error>
      <RaisedBy>Department</RaisedBy>
      <Number>3001</Number>
      <Type>business</Type>
      <Text>The submission of this document has failed due to departmental
    specific business logic in the Body tag.</Text>
      <Location/>
    </Error>
  </GovTalkErrors>
</GovTalkDetails>
<Body>
  <ErrorResponse xmlns="http://www.cssz.cz/XMLSchema/reldp/ErrorEnvelope">
    <Application>
      <Error>
        <RaisedBy>RELDP</RaisedBy>
        <Number>259</Number>
        <Text>RELDP - Údaj bno(Rodné číslo pojištěnce) není platné RČ
    (Rodné číslo není dělitelné 11.).; </Text>
      </Error>
      <Protocol>
        <error count="1" countErr="0" countWar="1" errMsg="RELDP - Údaj bno
    (Rodné číslo pojištěnce) není platné RČ (Rodné číslo není dělitelné 11.).;
     " errNumber="259" result="" type="RELDP" version="1.0">
          <head col1="rodné číslo" col2="Typ e-podání" col3="Rok resp. období"
     errMsg="Chyba" result="Výsledek zpracování"/>
          <rec xmlns="" col1="7511022014" col2="01" col3="2005"
    errMsg="RELDP - Údaj bno (Rodné číslo pojištěnce) není platné RČ
    (Rodné číslo není dělitelné 11.).; " errNum="259" result="WARNING"/>
        </error>
      </Protocol>
```

```
      </Application>
    </ErrorResponse>
    <Signature xmlns="http://www.cssz.cz/emp/timestamp" Version="1.0">
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <TimeStamp>
        <date>20060506</date>
        <time>13:22:11</time>
      </TimeStamp>
      <SignatureValue>elektronicka_znacka_BASE64</SignatureValue>
    </Signature>
  </Body>
</GovTalkMessage>
```

XML A.10: LIST REQUEST

```
<?xml version="1.0"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>request</Qualifier>
      <Function>list</Function>
      <TransactionID />
      <AuditID />
      <CorrelationID />
      <ResponseEndPoint />
      <Transformation>XML</Transformation>
      <GatewayTimestamp />
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>NH5PQJ743EHB</SenderID>
        <Authentication>
          <Method>MD5</Method>
          <Value>GRtBwY9qOkKW156P9VGKNQ==</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <GovTalkDetails>
    <Keys />
  </GovTalkDetails>
  <Body />
</GovTalkMessage>
```

XML A.11: LIST RESPONSE

```
<?xml version="1.0"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
```

```
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>response</Qualifier>
      <Function>list</Function>
      <TransactionID>
      </TransactionID>
      <CorrelationID>
      </CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
      <GatewayTimestamp>2006-05-06T11:29:48.216</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails />
  </Header>
  <GovTalkDetails>
    <Keys />
  </GovTalkDetails>
  <Body>
    <StatusReport>
      <SenderID>
      </SenderID>
      <StartTimeStamp>
      </StartTimeStamp>
      <EndTimeStamp>
      </EndTimeStamp>
      <StatusRecord>
        <TimeStamp>6.5.2006 13:29:20</TimeStamp>
        <CorrelationID>75CFEBEAB1C39FA8C847C6070734BDFD</CorrelationID>
        <Status>SUBMISSION_RESPONSE</Status>
      </StatusRecord>
      <StatusRecord>
        <TimeStamp>6.5.2006 13:29:28</TimeStamp>
        <CorrelationID>7C24DA3092A6D9EA445C931F8AFA0B14</CorrelationID>
        <Status>SUBMISSION_ERROR</Status>
      </StatusRecord>
    </StatusReport>
  </Body>
</GovTalkMessage>
```

XML A.12: DELETE REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>request</Qualifier>
```

```
      <Function>delete</Function>
      <TransactionID/>
      <AuditID/>
      <CorrelationID>6AABC1432549B28CD1C23986D2201E38</CorrelationID>
      <ResponseEndPoint/>
      <Transformation>XML</Transformation>
    </MessageDetails>
    <SenderDetails>
      <IDAuthentication>
        <SenderID>NH5PQJ743EHB</SenderID>
        <Authentication>
          <Method>MD5</Method>
          <Value>GRtBwY9qOkKW156P9VGKNQ==</Value>
        </Authentication>
      </IDAuthentication>
    </SenderDetails>
  </Header>
  <GovTalkDetails>
    <Keys>
      <Key Type="vars">99200075</Key>
    </Keys>
  </GovTalkDetails>
  <Body/>
</GovTalkMessage>
```

XML A.13: DELETE RESPONSE

```
<?xml version="1.0" encoding="utf-8"?>
<GovTalkMessage xmlns="http://www.govtalk.gov.uk/CM/envelope">
  <EnvelopeVersion>2.0</EnvelopeVersion>
  <Header>
    <MessageDetails>
      <Class>CSSZ_RELDP</Class>
      <Qualifier>response</Qualifier>
      <Function>delete</Function>
      <TransactionID/>
      <CorrelationID>6AABC1432549B28CD1C23986D2201E38</CorrelationID>
      <ResponseEndPoint PollInterval="10">https://bezpecne.dev.gov.cz/submission
    </ResponseEndPoint>
      <GatewayTimestamp>2006-05-06T11:18:11.256</GatewayTimestamp>
    </MessageDetails>
    <SenderDetails/>
  </Header>
  <GovTalkDetails>
    <Keys/>
  </GovTalkDetails>
  <Body/>
</GovTalkMessage>
```

XML A.14: datová věta RELDP

```
<?xml version="1.0" encoding="utf-8"?>
<RELDP cit="Plzeň" nam="ZČU" num="22" pnu="30100" str="Univerzitní" vs="99200075">
  <reldp typ="01" yer="2005">
    <client bno="7511022014">
      <name fir="Josef" sur="Novák"/>
      <adr cit="Plzeň" num="1" pnu="32300" pos="Plzeň" str="Sokolovská"/>
      <birth cit="Plzeň" dat="1975-11-02" nam="Novák"/>
    </client>
    <items coun="2" sdex="211" sinc="31827">
      <t1 cod="0++" dex="211" din="212" fro="2005-01-01" inc="31827" row="1"
    to="2005-07-31"/>
      <t2 cod="V" fro="2005-08-01" row="1" to="2005-12-31"/>
    </items>
    <comp cre="2005-07-22" fro="1993-01-01" nam="ZČU" vs="99200075">
      <adr cit="Plzeň" num="1" pnu="30100" str="Univerzitní"/>
    </comp>
  </reldp>
</RELDP>
```

Přehled zkratek

3DES - Triple Data Encryption Standard
AES - Advanced Encryption Standard
API - Application Program Interface
ASPI - Automatizovaný systém právních informací
CA - Certifikační autorita
CBC - Cipher block chaining
CFB - Cipher feedback mode
CRL - Certificate Revocation List
DES - Data Encryption Standard
DSA - Digital Signature Algorithm
DTD - Document Type Definition
DOM - Document Object Model
ECB - Electronic code book
EDE - Encryption-Decryption-Encryption
ELDP - Evidenční listy důchodového pojištění
HTTP - HyperText Transport Protocol
HTTPS - HyperText Transport Protocol Secure
JCE - Java Cryptography Extension
JAXP - Java API for XML Processing
JDOM - Java Document Object Model
MD - Message Digest
MFDS - Ministerstvo financí - Daňová správa
OCSP - Online certificate status protocol
OFB - Output feedback mode
OSVČ - Osoba samostatně výdělečně činná

PKI - Public Key Infrastructure
PKCS - Public Key Certification Standard
PRIHL - Přihlášky a odhlášky nemocenského pojištění
PVS - Portál veřejné správy
RC - Rivest Cipher
RSA - Rivest, Shamir, Adlemann
SAX - Simple API for XML
SGML - Standard Generalized Markup Language
SHA - Secure Hash Algorithm
SSL - Secure Sockets Layer
XML - eXtensible Markup Language
XSD - eXtensible Stylesheet Definition
XSL - eXtensible Stylesheet Language
XSLT - eXtensible Stylesheet Language Transformations
XSL-FO - eXtensible Stylesheet Language - Formatting Objects
W3C - World Wide Web Consortium

Literatura

[1] *Ministerstvo informatiky:* **Portál veřejné správy**
http://portal.gov.cz

[2] *Michael J. Young:* **XML krok za krokem**
iDnes, 2002

[3] *Miroslav Žák:* **XML začínáme programovat**
Grada, 2003

[4] *Petr Staníček:* **CSS kaskádové styly - Kompletní průvodce**
Computer Press, 2003

[5] *Jiří Kosek:* **Osobní stránky**
http://www.kosek.cz

[6] *seriál zive.cz:* **XML pro web aneb od teorie k praxi**
http://www.zive.cz

[7] *seriál interval.cz:* **XML - vývoj aplikací**
http://interval.cz

[8] *Pavel Herout:* **Učebnice jazyka Java**
Kopp, 2003

[9] *S. J. Chapman:* **Začínáme programovat v jazyce Java**
Computer Press, 2003

[10] *Brett Spell:* **Java - programujeme profesionálně**
Computer Press, 2002

[11] *Bogdan Kiszka:* **1001 tipů a triků pro programování v jazyce Java**
Computer Press, 2003

[12] *Jaroslav Měcháček:* **XML a Java**
Zpravodaj ÚVT MU. ISSN 1212-0901, 2001, roč.12, č.2, s.9-12

[13] *Jiří Žaloudek:* **Java Web Services**
http://www.volny.cz/zaloudekjiri/rp/index.html

[14] *Tomáš Doseděl:* **Počítačová bezpečnost a ochrana dat**
Computer Press, 2004

[15] *Kevin Mitnick & William Simon:* **Umění klamu**
Helion, 2003

[16] *Cryptoworld:* **webové stránky a e-zin**
http://crypto-world.info/

[17] *Ministerstvo informatiky:* **webové stránky**
http://www.micr.cz

[18] *RSA Security:* **normy PKCS**
http://www.rsasecurity.com/rsalabs/node.asp?id=2124

[19] *A. Taylor, B. Buege, R. Layman:* **Hacking bez tajemství Java a J2EE**
Computer Press, 2003

[20] *Dobrý web:* **Audit přístupnosti webových stránek PVS ČR**
http://i.iinfo.cz/urs-att/studie-02-auditpvs.pdf

[21] *The Legion of the Bouncy Castle:* **Bouncy Castle Crypto API**
http://www.bouncycastle.org/

[22] *Developpez.com:* **Cryptographie avec Bouncy Castle**
http://nyal.developpez.com/tutoriel/java/bouncycastle/

[23] *Portál veřejné správy:* **Diskuzní fórum pro podporu vývojářů**
https://bezpecne.dev.gov.cz/diskuze/

[24] *Česká správa sociálního zabezpečení:* **webové stránky**
http://www.cssz.cz

[25] *W3C:* **XML specifikace**
http://www.w3.org/TR/REC-xml

[26] *IETF:* **HTTP 1.1 specifikace**
http://www.ietf.org/rfc/rfc2616.txt

[27] *Netscape:* **SSL 3.0 specifikace**
http://wp.netscape.com/eng/ssl3/

[28] *IETF:* **TLS 1.0 specifikace**
http://www.ietf.org/rfc/rfc2246.txt

[29] *Microsoft:* **Případová studie transakční části PVS**
http://www.microsoft.com/cze/casestudies/MICR_portal.mspx

Evidenční list

Souhlasím s tím, aby moje diplomová práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

V Plzni dne 22. 5. 2006 Jan Kupka,

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou práci použil ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno	**Fakulta / katedra**	**Datum**	**Podpis**