Semestrální práce - evidence občanských průkazů

David Poslušný

Semestrální práce - evidence občanských průkazů

David Poslušný

Obsah

Seznam tabulek

Rejstřík

Úvod

Tato dokumentace obsahuje popis všech vytvořených částí semestrální práce v předmětu 4IZ238 XML - Teorie a praxe značkovacích jazyků. Dále lze v dokumentaci nalézt ukázky kódu a soupis elementů tvořící vytvořený značkovací jazyk. V semestrální práci bylo za úkol vytvořit XML dokument a k němu navrhnout odpovídající schéma i s využitím schematronu. Poté bylo nutné daný dokument, popřípadě dokumenty, převést do značkovacího jazyka HTML pomocí XSLT stylů a také využít procesor FO pro převod práce do formátu PDF.

Kapitola 1. XML dokument

Jako XML dokument jsem se v mé semestrální práce rozhodl vytvořit občanský průkaz, respektive evidenci občanských průkazů, protože dokumentů vytvářím více. Bylo tedy nutné zjistit, které všechny údaje se vyskytují na skutečném občanském průkazu a přetvořit je na XML elementy.

XML dokument se dělí v celku na 4 části, kde každá část vždy obsahuje nějaké údaje o držiteli průkazu. První část jsou osobní údaje jako např. jméno, příjmení, pohlaví. Druhá část obsahuje informace o datu a místu narození. V třetí části jsou zaznamenány informace o platnosti samotného průkazu a v poslední části je uvedeno trvalé bydliště, tj. adresa daného člověka. Informace o občanství a vydavateli průkazu nejsou zahrnuty do žádné části.

Popis každého konkrétního elementu v dokumentu vysvětluje následující tabulka:

Tabulka 1.1. Popis elementů

Element	Popis
obcansky_prukaz	kořenový element, který obsahuje atribut id, tj. identifikační číslo průkazu
osobni_udaje	element, který seskupuje osobní údaje a má jako atribut rodné číslo držitele průkazu
jmeno	křestní jméno držitele
prijmeni	příjmení držitele
pohlavi	pohlaví držitele
rodinny_stav	rodinný stav držitele (svobodný, rozvedená, vdaná...)
narozeni	element, který seskupuje údaje o datu a místu narození držitele
datum	element, který seskupuje údaje o nějakém datu - typ data je určen jeho atributem
den	den v měsíci
mesic	měsíc v roce
rok	rok, který společně s dnem a měsícem tvoří datum
misto	element seskupující informace o místu narození
mesto	město, popřípadě obec narození držitele
obvod	městská část či městský obvod daného města/obce
obcanstvi	informace o tom, v jaké zemi má držitel právem přiznané občanství
platnost	element seskupující data platnosti průkazu
adresa	element, který seskupuje podrobné informace o trvalém bydlišti držitele
ulice	ulice, v které držitel bydlí
cp	číslo popisné domu, ve kterém držitel bydlí
okres	okres do kterého patří město/obec v kterém držitel bydlí
vydavatel	informace o tom, který úřad tento průkaz vydal

Kapitola 2. XML schema

Soubor obsahující XML schéma `posd03_schema.xsd` v semestrální práci využívám pro validaci XML dokumentu. Schéma bylo vytvořeno podle metody slepého Benátčana a obsahuje strukturu a kontroly, které musí vytvořené XML dokumenty splňovat.

Ve schématu nejčastěji kontroluji formáty jednotlivých elementů přes regulární výrazy. Například pro rodné číslo, název českého města nebo identifikační číslo. Nutné bylo také zkontrolovat formáty dat u platnosti průkazu a u data narození. U jednoduchých textových údajů jako je například jméno, příjmení, pohlaví stačilo pouze omezit počet znaků.

Příklad kontroly rodného čísla

```
<xsd:simpleType name="rodne_cisloType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9]{6}\/[0-9]{4}"/>
    </xsd:restriction>
</xsd:simpleType>
```

Tato kontrola konkrétně využívá regulární výraz. Jedná se o lehký výraz, který nám zajistí, že prvních 6 znaků budou pouze číslice, dále musí být uvedeno lomítko, kterému následují 4 číslice. Pokud tento formát nebude splněn, schéma nám vyhodí chybu a daný pattern se nám ukáže, abychom poté data mohli do elementu napsat správně.

Příklad kontroly rodinného stavu

```
<xsd:simpleType name="rodinny_stavType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="svobodný"/>
        <xsd:enumeration value="ženatý"/>
        <xsd:enumeration value="rozvedený"/>
        <xsd:enumeration value="vdovec"/>
        <xsd:enumeration value="svobodná"/>
        <xsd:enumeration value="vdaná"/>
        <xsd:enumeration value="rozvedená"/>
        <xsd:enumeration value="vdova"/>
    </xsd:restriction>
</xsd:simpleType>
```

Rodinný stav nemůže nabývat jiných hodnot, než uvedených ve výčtu [rodinny_stavType]. V této kontrole se jedná o využití `<xsd:enumeration/>` kde můžeme výčtem určit, které hodnoty se přesně musí v daném elementu objevovat. Nelze tedy například do rodinného stavu napsat anglický výraz "single" místo "svobodný", nebo jakýkoliv jiný textový řetězec.

Ve schématu jsem také využil referenční integritu, konkrétně 2 pravidla, které kontrolují unikátnosti atributů v určitých elementech.

Ukázka referenčí integrity

```
<xsd:key name="unikatniTypyDatumu">
    <xsd:selector xpath="prefix:platnost/prefix:datum"/>
    <xsd:field xpath="@typ"/>
</xsd:key>
```

Z kódu lze vidět, že se zakličuje datum v platnosti, nikoliv v narození, a poté pomocí elementu [field] z jmenného prostoru [xsd] nastavíme, že daná data nesmí mít stejnou hodnotu atributu. Toto pravidlo jsem vytvořil za účelem, aby nebylo možné mít v dokumentu dvakrát datum vypršení či datum vydání.

Kapitola 3. Schematron

Dalším úkolem v semestrální práci bylo rozšířit předchozí XML schéma o schematron, či vytvořit samostatný soubor s danou technologií. V této práci jsem se rozhodl vytvořit nový soubor `posd03_schematron.sch` a do něj vložit danou kontrolu.

Kontrola pomocí schematronu

```
<pattern>
    <title>Kontrola platnosti průkazu</title>
    <rule context="prefix:obcansky_prukaz">
    <report test="prefix:platnost[1]/prefix:datum[2]/prefix:rok[1]
     !=
    (prefix:platnost[1]/prefix:datum[1]/prefix:rok[1]) + 10">
    Průkaz musí být platný 10 let od jeho vydání.
    Současná platnost je:
    <value-of select="prefix:platnost[1]/prefix:datum[2]/prefix:rok[1]
     -
    prefix:platnost[1]/prefix:datum[1]/prefix:rok[1]"/> let.
    </report>
    </rule>
</pattern>
```

Jak je vidět z titulku pravidla, jedná se o kontrolu platnosti průkazu. Ve schématu provedeme [test], který nám otestuje, jestli je rozdíl obou dat v platnosti 10 či nikoliv. Pomocí elementu [report] pak dostaneme zprávu o výsledku testu a také se uživateli vypíše současná platnost, tj. pokud například průkaz platí od 2005 do 2025, tak je to špatně, protože to není 10 let, ale 20 let. Pravidlo neřeší dny a měsíce, ale pouze roky. Smysl tohoto pravidla je, že v České republice se občanské průkazy vydávají s platností 5 let pro osoby mladší 15 let, s platností 10 let pro osoby starší 15 let a poté již s platností 35 let pro osoby staří 75 let. Počítat věk daného držitele z XML dokumentu a poté rozlišovat platnost jednotlivých průkazu mi přišlo moc složité, proto jsem tedy platnost standardizoval na 10 let.

Kapitola 4. HTML transformace

Transformaci do značkovacího jazyka HTML jsem provedl pomocí XSLT stylů uložených v souboru `posd03_xslt.xsl`. Soubor je tvořen pomocí templatů, které nám pomáhají daný výstup z XML dokumentu převést do HTML. Jelikož jsem měl více občanských průkazů a neměl jsem vytvořený žádný kořenový element [evidence] nebo něco obdobného, bylo nutné vytvořit nový soubor `posd03_evidence.xml`, v kterém jsem připojil všechny vytvořené XML dokumenty do jednoho souboru, aby jej mohly XSLT styly rozpoznat a úspěšně transformovat.

Soubor evidence, který seskupuje všechny XML dokumenty

```
<?xml version="1.0" encoding="UTF-8"?>
<evidence>
    <soubor name="posd03_dokument1.xml"></soubor>
    <soubor name="posd03_dokument2.xml"></soubor>
    <soubor name="posd03_dokument3.xml"></soubor>
    <soubor name="posd03_dokument4.xml"></soubor>
    <soubor name="posd03_dokument5.xml"></soubor>
    <soubor name="posd03_dokument6.xml"></soubor>
</evidence>
```

Daný soubor nám umožňuje na dokumenty odkazovat pomocí atributu [name]. Pokud bychom chtěli vypsat do HTML obsah jednotlivých dokumentů, lze tak učinit pomocí funkce [document(@name)/*]. Funkce nám vezme všechny dokumenty podle atributu a vypíše jejich celý obsah (tento fakt zaručí [*]).

Jako první jsem pomocí transformace vytvořil hlavní stránku s názvem `evidence.html`, v které je navigace, která odkazuje na jednotlivé stránky s podrobným výpisem informací z každého občanského průkazu. Dále je na stránce nestrukturovaný výčet zobrazený jako tabulka, kde můžeme najít všechny data na jednom místě. Zároveň je na stránce vložen obrázek nevyplněného vzoru občanského průkazu.

Vytvoření HTML kostry hlavní stránky

```
<xsl:template match="/">
   <xsl:result-document href="evidence.html">
    <html lang="cs">
     <head>
      <title>Evidence občanských průkazů</title>
       <link rel="stylesheet" type="text/css" href="styles.css"/>
       <meta name="author" content="David Poslušný"/>
       <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
     </head>
     <body>
      <h1>Evidence občanských průkazů</h1>
      <h3>Obsah evidence</h3>
      <xsl:call-template name="navigace"/>
      <p class="warning">
      *Poznámka: odkaz na konkrétní občanský průkaz je tvořen
      identifikačním číslem průkazu a celým jménem držitele.
      </p>
      <h2 class="heading-center">
      Nestrukturovaný výčet z evidence
      </h2>
      <xsl:for-each select="/evidence/soubor">
       <xsl:apply-templates select="document(@name)/*"/>
      </xsl:for-each>
```

Zde můžeme vidět kus kódu, který nám vytvoří základní kostru HTML pro hlavní stránku. Bylo nutné zde uvést všechny náležitosti HTML5, aby byla stránka validní. Do těla stránky jsem vložil odkaz

na template navigace a také se zde pomocí cyklu [for-each] vypisují informace z dokumentů pomocí dříve zmíněné funkce [document()]. Je zde však důležité určit kolikrát se má cyklus provést, v mém případě kolik je tam elementů [soubor], tj. kolik mám dokumentů.

Dále bylo nutné transformovat všechny dokumenty do samostatných stránek, a také na ně vytvořit odkaz. Vytvořil jsem tedy template pro kořenový element [obcansky_prukaz]. Uvnitř templatu bylo nezbytné pomocí elementu [result-document] vytvořit nový HTML dokument, stejným způsobem jako hlavní stránku. Do těchto jednotlivých stránek jsem pak elementem [apply-imports] vložil data, která k danému dokumentu patří a bylo potřeba už jen dokument "stylizovat".

Ukázka vytvoření stránek pro občanské průkazy

```
<xsl:template match="prefix:obcansky_prukaz">
   <xsl:result-document href="obcansky_prukaz_{generate-id(.)}.html">
    <html lang="cs">
     <head>
      <title><xsl:value-of select="./@id"/></title>
      <link rel="stylesheet" type="text/css" href="styles.css"/>
      <meta name="author" content="David Poslušný"/>
      <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
     </head>
     <body>
      <h1>Občanský průkaz - č. <xsl:value-of select="./@id"/></h1>
      <xsl:apply-imports/>
      <p style="text-align: center; margin-top: 50px;">
      <a class="nav-btn" href="evidence.html">
      Zpět na evidenci
      </a>
      </p>
      </body>
     </html>
    </xsl:result-document>
</xsl:template>
```

Pro navigaci a očíslování jednotlivých stránek jsem využil funkci [generate-id], která vytvoří unikátní identifikační číslo tomu elementu, v kterém se zrovna nacházíme, co se templatů týče. Je tedy nezbytné pomocí [xpath] výrazu najít cestu k elementu, který chceme využít. V mém případě se jedná o celý občanský průkaz a proto vyberu element [obcansky_prukaz].

Ukázka vytvoření navigace

```
<xsl:template name="navigace">
   <ul>
    <xsl:for-each select="//soubor">
     <li class="navbar">
      <a class="nav-btn"
      href=
      "obcansky_prukaz_{generate-id(document(@name)
      /prefix:obcansky_prukaz)}.html">
      <xsl:value-of select="document(@name)//@id"/>
       -
      <xsl:value-of select="document(@name)//prefix:jmeno"/>
      <xsl:text>&#032;</xsl:text>
      <xsl:value-of select="document(@name)//prefix:prijmeni"/>
      <xsl:apply-templates/>
      </a>
      <br/>
     </li>
    </xsl:for-each>
```

```
   </ul>
</xsl:template>
```

V tomto úseku kódu vytvářím navigaci pomocí zmíněné funkce [generate-id]. Identifikační číslo se vytvoří pro každý občanský průkaz a vloží se do HTML tagu [a] jako odkaz. Při vytváření jednotlivých stránek pro průkazy jsem do [URL] vložil opět stejnou funkci a pokud se jedná o stejný elemenent, tak ID se uloží přímo do odkazu na stránku a tím pádem se nám zajistí funkčnost navigace. Odkaz na stránku pak může vypadat například takto: `obcansky_prukaz_d2e1.html`

V neposlední řadě jsem k transformaci také připojil [CSS] styly, pomocí kterých jsem například vytvořil ohraničení tabulky, rozestup mezi textem na stránce anebo velikost písma či font celého dokumentu.

Ukázka kaskádových stylů

```
html {
    font-family: 'Segoe UI';
}
h1 {
    text-align: center;
}
.heading-center {
    text-align: center;
}
.nav-btn:hover {
    color: tomato;
}
.navbar {
    padding: 2.5px;
    font-size: 20px;
}
```

Kapitola 5. PDF transformace

Jako poslední část (mimo DocBooku) práce bylo dokumenty transformovat do formátu PDF. Na tuto transformaci jsem opět využil XSLT styly a také [FO procesor]. Výsledný soubor jsem pojmenoval jako `posd03_xsl_fo.xsl`. Jedná se o velmi podobnou transformaci jako u HTML, akorát je nutné výsledný XSLT styl, který obsahuje fo blocky a ostatní elementy daného jmenného prostoru, převést pomocí FO procesoru do PDF. Soubor vygenerovaný z procesoru se jmenuje `evidence.pdf`

Výsledný dokument je rozdělen na více částí a má jiný design než, který jsem použil v HTML stránkách. Úvodní strana obsahuje základní informace a také logo školy. Na druhé stránce je obsah s odkazy, takže je možné se kliknutím přenést na jiné místo v dokumentu. Poté již následují kapitoly: záznamy o platnosti průkazů, občanské průkazy a ukázka občanského průkazu.

Vytvoření layoutu dokumentu

```
<fo:layout-master-set>
            <fo:simple-page-master master-name="pdf-page"
                                   page-height="297mm"
                                   page-width="210mm"
                                   >
                <fo:region-body margin="15mm"/>
                <fo:region-before extent="10mm"/>
                <fo:region-after extent="10mm"/>
                </fo:simple-page-master>
</fo:layout-master-set>
```

Tato část kódu vytvoří rozložení stránky, tj. rozměry, odsazení v záhlaví a zápatí, jazyk a font dokumentu.

Vytvoření stránek pro občanské průkazy

```
<xsl:template match="prefix:obcansky_prukaz">
    <fo:block id="{generate-id(.)}"
              text-align="center"
              font-size="24px"
              font-weight="bold"
              margin-top="15px">
    Občanský průkaz - č. <xsl:value-of select="./@id"/>
    </fo:block>
    <xsl:apply-templates/>
</xsl:template>
```

Tento kousek kódu nám zajistí, že každý občanský průkaz se bude vypisovat na novou stránku a také dostane své identifikační číslo pro aktivní obsah pomocí funkce [generate-id] stejně jako v HTML transformaci. Element [apply-templates] vypíše obsah každého občanského průkazu na jeho příslušnou stránku.

Ukázka vypsání vydavatele průkazu

```
<xsl:template match="prefix:vydavatel">
      <fo:block font-size="20px"
                font-weight="bold"
                margin-top="15px"
                margin-bottom="15px">
      Údaje o vydavateli
      </fo:block>
      <fo:block break-after="page">
      Vydavatel: <xsl:value-of select="."/>
```

```
      </fo:block>
</xsl:template>
```

Kód řeší výpis konkrétní hodnoty vydavatele z občanského průkazu. Pomocí templatu jsem se do požadovaného elementu dostal, a proto hodnotu daného elementu vypíšu v [xpath] výrazu pomocí [.]. Atribut [break-after] provede konec stránky a tím pádem se ostatní data již budou vypisovat na další stránku. Tímto se pak každý občanský průkaz vypisuje na samostatné stránce.