Dokumentace pro semestrální práci Průvodce památkami

Klára Ťoupalíková

Dokumentace pro semestrální práci Průvodce památkami

Klára Ťoupalíková

Obsah

Úvod

Tato dokumentace popisuje semestrální práci z oblasti památek ČR. V kapitole 1 je popsána struktura XML dokumentu a jednotlivé komponenty. V druhé kapitole se zaměřím na použitá schémata a ve třetí na XSLT transformace do pdf a html.

Kapitola 1. XML

V této části práce je popsána struktura XML dokumentace pro průvodce památek.

XML dokument

XML dokument obashuje kořenový element průvodce. V kořenovém elementu **průvodce** musí být elementy názevDíla, autor a památky. **Památky** mohou mít v sobě nekonečně mnoho elementů **památka**, které tvoří samotné tělo projektu.

Struktura složitějších elementů

Složitějšími elementy je myšlen element mající v sobě dva a více různých podelementů.

průvodce	kořenový element, zastřešuje veškerý obsah dokumentu
památka	zastřešuje jednotlivé údaje o památce
umístění	obsahuje elementy **kraj** a **mapa**
hodnocení	je podelementem **reference** a obsahuje informace týkající se počtu hvězd, datumu uvedení recenze a samotnou recenzi v podobě komentáře

Kapitola 2. Schémata

XML schéma nám umožňuje definovat strukturu dokumentu pro průvodce. V této práci jsem použila schémata dvojího typu. Za základní schéma považuji XML Schema, ve kterém je definována většina pravidel. Za doplňkové schéma považuji kód psaný v Schematronu.

W3C XML Schema

Níže můžeme vidět ukázku kódu z W3C XML Schema, kde omezujeme element **Oblíbenost**.Pro element byl vytvořen vlastní typ (viz první část kódu), který je dále omezován v druhé části kódu na desetinné číslo s minimální a maximální možnou hodnotou.

```
<xs:complexType name="PamátkaTyp">
        <xs:sequence>
            <xs:element name="název" type="NázevTyp"/>
            <xs:element name="postaven" type="PostavenTyp"/>
            <xs:element name="oblíbenost" type="OblíbenostTyp"/>
            <xs:element name="památkaPoblíž" type="AutorTyp"/>
            <xs:element name="umístění" type="UmístěníTyp"/>
            <xs:element name="sloh" type="AutorTyp"/>
            <xs:element name="foto" type="FotoTyp"/>
            <xs:element name="popis" type="PopisTyp"/>
            <xs:element name="zajímavost" type="ZajímavostTyp"/>
            <xs:element name="majitelé" type="MajiteléTyp"/>
            <xs:element name="navštívitelný" type="NavštívitelnýTyp"/>
            <xs:element name="vstupné" type="VstupnéTyp"/>
            <xs:element name="přístupnost" type="PřístupnostTyp"/>
            <xs:element name="služby" type="SlužbyTyp"/>
            <xs:element name="reference" type="ReferenceTyp"/>
            <xs:element name="webovky" type="WebTyp"/>
        </xs:sequence>
        <xs:attribute name="druh" type="DruhTyp" use="required"/>
    </xs:complexType>

<xs:simpleType name="OblíbenostTyp">
        <xs:restriction base="xs:decimal">
            <xs:minInclusive value="0.0"/>
            <xs:maxInclusive value="5.0"/>
        </xs:restriction>
    </xs:simpleType>
```

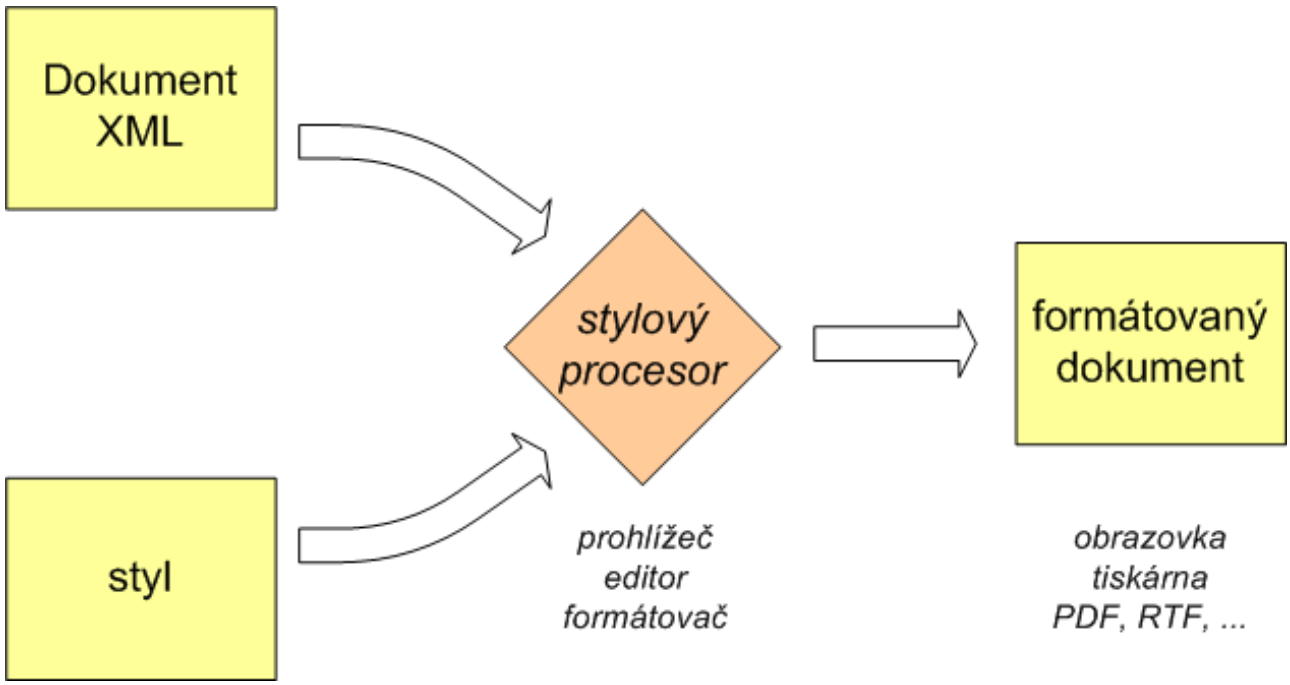
Schematron

Ukázka použitého schematronu - následující kód se stará o kontrolu, roku, kdy byla památka postavena.

```
<sch:pattern>
        <sch:title>Rok je zadán v pořádku</sch:title>
        <sch:rule context="o:postaven">
            <sch:assert test="max(../o:postaven) &lt; 2019">
                Památka musí být postavená dříve než v roce 2019.
            </sch:assert>
        </sch:rule>
    </sch:pattern>
```

Kapitola 3. XSLT transformace

Transformace XML dokumentu prochází jako na obrázku.



Použité XSLT transformace

V zadání projektu bylo vytvořit XSLT transformace dvojího typu:

1. transformace do HTML
2. transformace do PDF

Příklad odlišnosti 1

Přestože jde o stejný princip, používáme odlišné elementy k dosažení stejného/podobného výsledku. V obou případech generujeme obsah a další stránky s informacemi o jednotlivých památkách. (viz kód níže)

```
<xsl:template match="památky">
        <div id="kategorie">
            <h4>Co hledáte?</h4>
            <div class="flex-container">
                <table>
                    <tr>
                        <th>HRADY</th></tr>
                    <xsl:for-each select="památka[@druh='hrad']">
                        <xsl:sort select="název" lang="cs"/>
                        <tr>
                            <td>
                                <a href="{generate-id(.)}.html">
                                    <xsl:value-of select="název"/>
                                </a>
                            </td>
                        </tr>
                    </xsl:for-each>
```

```
                </table>
   ...
```

generování odkazu v HTML

```
<fo:block margin="3mm" padding="3mm" >
                            <xsl:for-each select="průvodce/památky/památka[@dru
                                <xsl:sort select="název" data-type="text" order=
                                <fo:block text-align-last="justify">
                                    <fo:basic-link internal-destination="{genera
                                        <xsl:value-of select="název"/>
                                        <fo:leader leader-pattern="dots"/>
                                        <fo:page-number-citation ref-id="{genera
                                    </fo:basic-link>
                                </fo:block>
                            </xsl:for-each>
                        </fo:block>
   ...
```

generování odkazu v FO- pro PDF

Příklad odlišnosti 2

Následující příklad ukazuje, jakým způsobem je transformace do HTML jednodušší a úspornější. V transformaci do PDF musíme vepisovat dodatečná metadata, která u HTML nejsou zapotřebí.

```
 <xsl:template match="počet_hvězd">
        <xsl:text>POČET HVĚZD: </xsl:text>
        <xsl:if test="text()='5'">
            <xsl:text>&#9733; &#9733; &#9733; &#9733; &#9733;</xsl:text>
        </xsl:if>
        <xsl:if test="text()='4'">
            <xsl:text>&#9733; &#9733; &#9733; &#9733;</xsl:text>
        </xsl:if>
        <xsl:if test="text()='3'">
            <xsl:text>&#9733; &#9733; &#9733;</xsl:text>
        </xsl:if>
        <xsl:if test="text()='2'">
            <xsl:text>&#9733; &#9733;</xsl:text>
        </xsl:if>
        <xsl:if test="text()='1'">
            <xsl:text>&#9733;</xsl:text>
        </xsl:if>
    </xsl:template>
```

zápis znaku hvězdy v HTML

```
<xsl:template match="počet_hvězd">
        <fo:block margin="2mm">
            <xsl:if test="text()='5'">
                <fo:inline font="14pt ZapfDingbats">&#x2605; &#9733; &#9733; &#
            </xsl:if>
            <xsl:if test="text()='4'">
                <fo:inline font="14pt ZapfDingbats">&#x2605; &#9733; &#9733; &#
            </xsl:if>
            <xsl:if test="text()='3'">
                <fo:inline font="14pt ZapfDingbats">&#x2605; &#9733; &#9733;</f
            </xsl:if>
            <xsl:if test="text()='2'">
```

```
                <fo:inline font="14pt ZapfDingbats">&#x2605; &#9733;</fo:inline
            </xsl:if>
            <xsl:if test="text()='1'">
                <fo:inline font="14pt ZapfDingbats">&#x2605;</fo:inline>
            </xsl:if>
        </fo:block>
    </xsl:template>
```

zápis znaku hvězdy v FO - PDF

Rejstřík