Transformace XML dokumentů pomocí XSLT

Řešení příkladů

Irena Mlýnková, Martin Nečaský

1. příklad

Vyzkoušejte si implicitní šablony. Napište XSLT skript bez šablon. Napište XSLT skript s jednou šablonou matchující kořen vstupního XML dokumentu a generující kostru výstupního XHTML dokumentu (tj. elementy **html**, **head**, **body**). Do těla XHTML dokumentu implicitně zpracujte ostatní uzly ze vstupního XML dokumentu.

Řešení

XSLT skript bez šablon, respektive pouze s implicitními šablonami:

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

a XSLT skript s jednou šablonou generujícím kostru XHTML dokumentu a zpracovávajícím obsah XML dokumentu implicitními šablonami:

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

  <xsl:template match="/">
    <html>
      <head>
        <title>PROJEKTY</title>
      </head>
      <body>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

kde pro implicitní zpracování musíme volat **<xsl:apply-templates />** jinak by nebyl zbytek XML dokumentu zpracován. Naše šablona totiž matchuje kořen XML dokumentu a bez explicitního zavolání **<xsl:apply-templates />** do potomků kořene nepokračuje.

2. příklad

Upravte předchozí XSLT skript tak, aby tělo výstupního XHTML dokumentu zachycovalo strukturu vstupního XML dokumentu pomocí **div** elementů. Textové hodnoty ve vstupním XML dokumentu ignorujte.

Řešení

Do předchozího XSLT skriptu přidáme následující dvě šablony:

```
<xsl:template match="*">
  <div>
    <xsl:apply-templates />
  </div>
</xsl:template>
```

pro zachycení struktury vstupního XML dokumentu (každý element ve vstupním XML dokumentu je přetransformován do XHTML elementu **div**) a

```
<xsl:template match="text()" />
```

pro ignorování textových uzlů ve vstupním XML dokumentu.

3. příklad

Upravte XSLT skript z příkladu 1 tak, aby tělo výstupního XHTML dokumentu obsahovalo pro každý atribut ze vstupního XML dokumentu text *[ATT:nazev="hodnota"]* Žádná jiná data ze vstupního XML dokumentu se ve výstupu nesmí objevit.

Řešení

Do XSLT skriptu z řešení příkladu 1 přidáme následující tři šablony:

```
<xsl:template match="*">
  <xsl:apply-templates select="@*"/>
  <xsl:apply-templates />
</xsl:template>
```

která pro matchovaný element, vyvolá pro každý jeho atribut následující šablonu a pokračuje do potomků.

```
<xsl:template match="@*">
  <xsl:text>[ATT:</xsl:text>
  <xsl:value-of select="name()" />
  <xsl:text>="</xsl:text>
  <xsl:value-of select="." />
  <xsl:text>"]</xsl:text>
</xsl:template>
```

která pro matchovaný atribut vypíše požadovaný text a

```
<xsl:template match="text()" />
```

pro ignorování textových uzlů ve vstupním XML dokumentu.

4. příklad

Vytvořte XSLT skript, který vstupní XML dokument přetransformuje na XHTML dokument, který kopíruje hierarchickou strukturu skupin a projektů. Pro každou skupinu je uveden její název v elementu **h2** a neuspořádaný seznam podskupin či projektů. Pro každý projekt je uveden jeho název v elementu **h3**.

Řešení

Možné řešení:

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

  <xsl:template match="/">
    <html>
      <head><title>Hierarchie projektů</title></head>
      <body>
        <ul><xsl:apply-templates /></ul>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="project-group">
    <li>
      <h2><xsl:value-of select="info/title"/></h2>
      <ul><xsl:apply-templates /></ul>
    </li>
  </xsl:template>

  <xsl:template match="project">
    <li>
      <h3><xsl:value-of select="info/title"/></h3>
    </li>
  </xsl:template>

  <xsl:template match="node()" />

</xsl:stylesheet>
```

5. příklad

Doplňte předchozí XSLT skript tak, aby se na začátku generoval seznam XHTML odkazů na projekty. K identifikaci projektů využijte jejich označení (tj. hodnoty atributu **oznaceni**).

Řešení

V XHTML můžeme vytvořit odkaz na jinou část dokumentu reprezentovanou daným elementem pomocí elementu **a**, kde atribut **href** obsahuje hodnotu atributu **id** elementu, na který se odkazujeme. Šablonu matchující kořen tedy změníme tak, aby vygenerovala seznam odkazů na projekty, k čemuž využijeme označní projektů:

```
<xsl:template match="/">
 <html>
   <head><title>Hierarchie projektu</title></head>
   <body>
     <ol>
       <xsl:for-each select="//project">
         <a>
           <xsl:attribute name="href">
             <xsl:text>#</xsl:text>
             <xsl:value-of select="@oznaceni"/>
           </xsl:attribute>
           <xsl:value-of select="info/title" />
         </a>
       </xsl:for-each>
     </ol>
     <ul><xsl:apply-templates /></ul>
   </body>
 </html>
</xsl:template>
```

a šablonu matchující projekty změníme tak, aby generovala i atribut **id** s označením projektu:

```
<xsl:template match="project">
 <li>
   <h3>
     <xsl:attribute name="id">
       <xsl:value-of select="@oznaceni" />
     </xsl:attribute>
     <xsl:value-of select="info/title"/>
   </h3>
 </li>
</xsl:template>
```

6. příklad

Protože všechny projekty nemusejí mít svoje označení, nebude předchozí řešení fungovat obecně. Upravte ho tedy tak, aby fungovalo v každém případě.

Řešení

Využijeme pořadí uzlů v XML dokumentu, které každý uzel jednoznačně identifikuje. Budeme počítat pořadí daného elementu **project** v sekvenci všech elementů \texttt{project} v XML dokumentu. Pro seznam odkazů na projekty si tedy vezmeme sekvenci všech elementů **project** pomocí XPath výrazu **/descendant::project** (ne **//project** !) a pomocí funkce **position()** zjiš ujeme pozici právě transformovaného:

```
<xsl:template match="/">
 <html>
   <head><title>Hierarchie projektu</title></head>
   <body>
     <ol>
       <xsl:for-each select="//project">
         <a>
           <xsl:attribute name="href">
             <xsl:text>#p</xsl:text>
             <xsl:value-of select="position()"/>
           </xsl:attribute>
           <xsl:value-of select="info/title" />
         </a>
       </xsl:for-each>
     </ol>
     <ul><xsl:apply-templates /></ul>
   </body>
 </html>
</xsl:template>
```

Pro výpis samotných projektů to je trochu složitější, protože musíme sledovat hierarchickou strukturu. Můžeme ale pro daný element **project** spočítat počet všech předcházejících elementů **project** v XML dokumentu, čímž opět spočítáme pořadí právě transformovaného elementu **project** v sekvenci všech elementů **project** v XML dokumentu:

```
<xsl:template match="project">
 <li>
   <h3>
     <xsl:attribute name="id">
       <xsl:text>p</xsl:text>
       <xsl:value-of select="count(preceding::project)+1" />
     </xsl:attribute>
     <xsl:value-of select="info/title"/>
   </h3>
 </li>
</xsl:template>
```

7. příklad

Doplňte předchozí XSLT skript tak, aby se pro každý projekt vygenerovala také informace o vedoucím projektu, včetně jeho telefonního čísla. Pokud projekt nemá vedoucího, skript to rozpozná a vypíše, že projekt nemá vedoucího.

Řešení

Šablonu matchující projekty pozměníme, aby generovala i info o vedoucím, kterého najdeme průchodem do předků projektu:

```
<xsl:template match="project">
 <li>
   <h3>
     <xsl:attribute name="id">
       <xsl:text>p</xsl:text>
       <xsl:value-of select="count(preceding::project)+1" />
     </xsl:attribute>
     <xsl:value-of select="info/title"/>
   </h3>
   <p>
     <span style="font-weight:bold">Vedoucí: </span>
     <xsl:call-template name="person-info">
       <xsl:with-param
         name="person"
         select="(./ancestor-or-self::*/info/supervisor)[last()]" />
     </xsl:call-template>
   </p>
 </li>
</xsl:template>
```

a samotné info o daném vedoucím vygenerujeme pomocí speciální šablony volané explicitně jejím jménem:

```
<xsl:template name="person-info">
  <xsl:param name="person"/>
  <xsl:choose>
    <xsl:when test="$person">
      <xsl:value-of select="$person/firstname/text()"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="$person/surname/text()"/>
      <xsl:text>, tel.: </xsl:text>
      <xsl:value-of select="//teacher[surname = $person/surname]/phone"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>Projekt nemá vedoucího.</xsl:text>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

8. příklad

Přidejte ke každému projektu ještě jeho popis tak, že každý element **para** bude nahrazen elementem **p** a každý element **note** bude nahrazen elementem **div** obsahujícím na začátku XHTML kód **Poznámka: **.

Řešení

Možné řešení:

```
<xsl:template match="project">
 <li>
   <h3>
     <xsl:attribute name="id">
       <xsl:text>p</xsl:text>
       <xsl:value-of select="count(preceding::project)+1" />
     </xsl:attribute>
     <xsl:value-of select="info/title"/>
   </h3>
   <p>
     <span style="font-weight:bold">Vedoucí: </span>
     <xsl:call-template name="person-info">
       <xsl:with-param
         name="person"
         select="(./ancestor-or-self::*/info/supervisor)[last()]" />
     </xsl:call-template>
   </p>
   <xsl:apply-templates select="./description" />
 </li>
</xsl:template>

<xsl:template match="description">
 <p><xsl:apply-templates /></p>
</xsl:template>

<xsl:template match="para">
 <p><xsl:apply-templates /></p>
</xsl:template>

<xsl:template match="para/text()">
 <xsl:value-of select="." />
</xsl:template>

<xsl:template match="note">
 <div><span>Poznamka: </span><xsl:value-of select="." /></div>
</xsl:template>
```

9. příklad

Pro každý projekt ještě přidejte seznam kódů předmětů, ve kterých může být řešen. Kódy předmětů budou obsaženy v elementu **p** uvozeným XHTML kódem **Předměty: ** a budou odděleny čárkami. Za posledním kódem, pokud je kódů více, bude tečka.

Řešení

Možné řešení:

```
<xsl:template match="project">
 <li>
   <h3>
     <xsl:attribute name="id">
       <xsl:text>p</xsl:text>
       <xsl:value-of select="count(preceding::project)+1" />
     </xsl:attribute>
     <xsl:value-of select="info/title"/>
   </h3>
   <p>
     <span style="font-weight:bold">Vedoucí: </span>
     <xsl:call-template name="person-info">
       <xsl:with-param
         name="person"
         select="(./ancestor-or-self::*/info/supervisor)[last()]" />
     </xsl:call-template>
   </p>
   <p>
     <span style="font-weight:bold">Předměty: </span>
     <xsl:for-each select="./ancestor-or-self::*/info/course">
       <xsl:value-of select="." />
       <xsl:if test="not (position()=last())">
         <xsl:text>, </xsl:text>
       </xsl:if>
       <xsl:if test="(position()=last()) and (last()>1)">
         <xsl:text>.</xsl:text>
       </xsl:if>
     </xsl:for-each>
   </p>
   <xsl:apply-templates select="./description" />
 </li>
</xsl:template>
```