X-definice 3.1

Úvod do konstrukčního režimu

Václav Trojan

trojan@syntea.cz

Obsah

Tabulky

Tabulka 1- Použité pojmy a zkratky

Pojem	Vysvětlení
XML	Rozšířitelný značkovací jazyk
X-definice	1. jazyk na bázi XML navržený pro popis a zpracování XML dat 2. XML soubor, který obsahuje data ve fromě X-definice
validační režim	režim, ve kterém jsou zpracována vstupní XML data podle X-definice
konstrukční režim	režim, ve kterém je podle X-definice vytvořen XML objekt (případně JSON objekt nebo X-komponenta)
kvantifikátor	zápis minimálního a maximálního počtu výskytu prvku v modelu
atribut	XML atribut
element	XML element
textový uzel	vnitřní uzel elementu, který obsahuje neprázdný řetězec znaků
model	popis XML elementu v jazyce X-definice
Skript	programovací jazyk v modelech a dedeklaračních částech X-definic
kontext	data použitá pro konstrukci XML objektů
string	řetězec znaků přípustných v XML datech
ResultSet	Objekt, který je výsledek příkazu s databázovým dotazem (databázová tabulka)
NamedValue	Pojmenovaná hodnota. Objekt, obsahující jméno (string) a nějakou hodnotu (může být i "null")
Container	Objekt, obsahující sekvenci datových hodnot a mapu pojmenovaných hodnot
reportér	Objekt, obsahující hlášení, vzniklá v průběhu práce procesoru X-definic
procesor X-definic	implementace nástroje, který zpracovává X-definice

1 Úvod

Předpokladem pro čtení tohoto textu je, že čtenář je seznámen s režimem zpracování a validace a rozumí jazyku X-definic. Režim konstrukce popíšeme krok po kroku. Doufejme, že čtenář nakonec ocení výhody i jednoduchost tohoto způsobu vytváření XML dokumentů.

Zatímco ve validačním módu je činnost procesoru X-definic řízena vstupním XML dokumentem, v konstrukčním režimu je procesor řízen daty v X definici. To znamená, že na základě informací v X-definici se vytvoří výstupní XML data. Tedy namísto zpracování vstupního dokumentu, při kterém se v X-definici vyhledávají modely odpovídající vstupním datům (tj. proces je "řízen" vstupními daty), je v konstrukčním režimu použita X-definice jako předpis pro tvorbu výsledných dat (tj. proces je "řízen" X-definicí). Ke konstrukci výsledku jsou použity hodnoty, které jsou specifikovány ve Skriptu v sekci "create" v příslušných částech modelu. Poznamenejme hned v této souvislosti, že při konstrukci výsledného dokumentu může být zápis sekce "create" vynechán. V takovém případě se provede implicitní akce, která poskytne hodnotu, podle níž se provede konstrukce výsledku. Důležité je upozornit, že kód sekce "create" se provede jen v konstrukčním režimu a ve validačním režimu se ignoruje.

Poznámka: Všude v textu dokumentu kliknutím na internetový odkaz umístěný pod příkladem můžete tento příklad spustit a můžete dokonce zkusit provádět v něm úpravy.

Dotazy, poznámky a hlášení chyb můžete posílat na adresu: *xdef@syntea.cz*.

Aktuální verzi X-definic můžete stáhnout na adrese: *http://www.xdef.cz*.

2 Jak spustit konstrukční režim z Java programu

V Java programu spustíme konstrukční režim z objektu "cz.syntea.xdef.XDDocument" metodou "xcreate". Výsledkem této metody je objekt "org.w3c.dom.Element". Parametr této metody je objekt "javax.xml.namespace.QName", který odkazuje na model, podle kterého má být vytvořen výsledek (nebo,

pokud model nemá deklarován jmenný prostor, stačí uvést i string se jménem modelu). Samozřejmě musí v X-definici takový model existovat. Druhý parametr metody "xcreate" je reportér (objekt "cz.syntea.xdef.reporter.ReportWriter"), do kterého se zapisují případné chyby zjištěné během procesu konstrukce. Pokud je hodnota reportéru "null" a přitom jsou hlášeny nějaké chyby, pak je běh procesoru X-definic ukončen výjimkou RuntimeException, ve které je text zprávy o chybách.

Typická posloupnost Java příkazů pro spuštění konstrukce:

```
XDPool xpool = XDFactory. compileXD(...);
XDDocument xd = xpool. createXDDocument(name); /* jméno X-definice */
QName qname = new QName(uri, name); /* reference na model */
...
ArrayReporter reporter = new ArrayReporter();
Element elem = xd.xcreate(qname, reporter);
if (reporter.errors()) (
 //zpracování chyb
)
```

Pokud chceme pro konstrukční režim nastavit výchozí zdrojová data (mluvíme o "kontextu" – viz kapitola 5. Kontext pro konstrukci XML), zařadíme před volání metody "xcreate" volání metody "setXDContext", které jako parametr zadáme element obsahující data pro konstrukci:

```
xd.setXDContext("project/data.xml");
xd.xcreate(qname, reporter);
```

3 Explicitní zápis zdroje dat pro tvorbu XML

Představme si nejdříve, že pro konstrukční režim je sekce "create" povinná a že se příslušné objekty vytvářejí pouze z hodnot, které vzniknou na základě příkazu v této sekci.

Vezměme nejjednodušší možný příklad a položme si otázku, co asi musí být výsledkem příkazů v odpovídajících sekcích "create", aby byl procesor X-definic schopen vytvořit požadovaný výsledek. Jinými slovy, co procesor musí znát (tedy jakou hodnotu očekává v sekci "create") pro konstrukci elementu "A" popsaného v následující X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
  <A xd:script="create ???"/>
</xd:def>
```

Vzhledem k tomu, že element "A" neobsahuje žádná další data je zřejmé, že jediné, co procesor potřebuje znát, bude "ano" anebo "ne", tedy jestli se vůbec element "A" má vytvořit nebo ne. V našem případě k tomu bude stačit hodnota boolean. X-definice tedy může mít tvar:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true"/>
</xd: def>
```

http://xdef.syntea.cz/tutorial/cs/example/C201.html

výsledek bude:

```
<A/>
```

Jednoduché! Zkuste a uvidíte, že to tak je. Kdyby výsledkem příkazu sekce "create" byla hodnota "false", element "A" by se nevytvořil (a procesor by ohlásil chybu, že dokument nebyl vytvořen).

S hodnotou "boolean" bychom však obecně nevystačili. Vezměme složitější příklad. V elementu A budeme požadovat určitý počet elementů "B" jako potomky:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true">
      <B xd:script="occurs *; create ???"/>
   </A>
</xd:def>
```

Kdybychom do create sekce elementu "B" napsali "true", museli bychom si na výsledek počkat. Navíc by program skončil přetečením paměti, protože by se postupně vytvářely všechny elementy, odpovídající

podmínce výskytu. Raději to ani nezkoušejte. Problém ale vyřešíme snadno. Procesoru předáme jednoduše počet elementů, který požadujeme. Naše X-definice bude tedy vypadat např. takto:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create 1">
      <B xd:script="occurs *; create 3"/>
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C202.html

Vytvoří se tedy tři elementy "B" a výsledek bude:

```
<A><B/><B/><B/></A>
```

Pozorný čtenář může X-definici upravit také takto:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true">
      <B xd:script="occurs *; create 3"/>
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C203.html

Jak jsme viděli, pro vytvoření elementu, nám obecně stačí znát podmínku, kdy se element má vytvořit. Pokud je hodnota výrazu v sekci create Container nebo sekvence prvků např. z metody xpath, xquery, řádky tabulky databáze atd. můžeme se dívat na tuto hodnotu jako na iterátor, který nám dává postupně hodnoty pro tvorbu jednotlivých prvků, z nichž má být vytvořen daný element.

Další možností je tedy zadat příkazem "create" množinu prvků, podle kterých se má element vytvořit. Můžeme např. použít typ "Container", který, jak víme, může obsahovat sekvenci prvků. Každý prvek v sekvenční části objektu Cointainer (pokud není null) je podkladem pro tvorbu elementu podle modelu. Upravme náš příklad tak, že sekci "create" poskytneme množinu prvků (připomeňme, že zápis "[false, 0, 'abc']" je konstruktor pro typ "Container" se sekvencí se třemi prvky s hodnotami false , 0 a string "abc"):

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true">
      <B xd:script="occurs *; create [false, 0, 'abc']"/>
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C204.html

Vytvoří se opět tři elementy "B". Container má tři prvky a jakákoli hodnota prvku kromě "null" (i false nebo 0) způsobi generování elementu. Výsledek tedy bude:

```
<A><B/><B/><B/></A>
```

Následující X-definice nevytvoří žádný element "B":

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true">
      <B xd:script="occurs *; create []"/>
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C205.html

Výsledek:

```
<A/>
```

Je nasnadě, že stejného výsledku dosáhneme např. následující X-definicí:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create true">
      <B xd:script="occurs 0..*; create [null, null, null]"/>
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C206.html

Nyní si položme otázku, co by se stalo, kdybychom předali jako zdroj dat k tvorbě elementu jiný element. Mějme X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create new Element('X')"/>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C207.html

Program má k dispozici objekt, dokonce element. Řekněme hned na začátku, že nám na jménu elementu většinou nezáleží (výjimkou je element "xd:any" – pokud čtenář bude pokračovat v četbě, uvidí proč). Mějme element "X", který předáme jako zdroj dat pro konstrukci požadovaného elementu "A". Výsledek opět bude:

```
<A/>
```

Čtenář si jistě sám odpoví na otázku, co bude výsledkem následující X-definice:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create null">
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C208.html

Co nastane, když zdrojem dat bude string? Tedy:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
   <A xd:script="create 'blabla'">
      required string();
   </A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C209.html

I v tomto případě dostaneme jako výsledek element "A" (pokud hodnota stringu nebude "null"). Trochu nyní předběhněme další výklad. Ze stringu procesor vyrobí pomocný element, který bude mít textovou hodnotu převzatou z hodnoty stringu. Tento element pak bude poskytnut jako zdroj dat pro konstrukci elementu podle daného modelu (tato vlastnost umožňuje snadno vytvářet elementy s textovou hodnotou). Výsledek bude:

```
<A>blabla</A>
```

Popišme nyní situaci, která je vlastně pro reálné použití jedna z typických variant. Představme si, že chceme vytvářet náš XML dokument z jiného XML objektu. Zdroj dat pro tvorbu elementu můžeme poskytnout buď jako výsledek XPath výrazu, nebo třeba jako výsledek nějakého příkazu databáze.

Nejprve si ukažme, jak postupovat pomocí příkazu XPath. Mějme XML následující dokument, který použijeme jako zdroj dat pro tvorbu požadovaného dokumentu:

```
<X>
  <A/>
  <Y/>
  <Z/>
  <A/>
  <Z/>
</X>
```

Požadovaný výsledek nechť je element "A", jehož potomci " B" budou vytvořeny z elementů "A" ze zdroje dat a dále požadujme element "C" má být vytvořen z druhého výskytu elementu "Z". Požadovaný výsledek tedy bude:

```
<A>
  <B/>  (vytvořen z prvního "A")
  <B/>  (vytvořen z druhého "A")
  <C/>  (vytvořen z druhého "Z")
</A>
```

Při návrhu X-definice použijeme metodu Skriptu "xpath", u které jako první parametr uvedeme XPath výraz a jako druhý parametr uvedeme element, nad kterým se má XPath výraz provést. Poznamenejme, že výsledek XPath pro element je ve Skriptu hodnota typu Container (a s touto variantou jsme se už setkali), vytvořená z objektu NodeList, který je výsledkem XPath:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
  <xd:declaration>
    <![CDATA[
       /* Vytvoříme element do proměnné source. */
       Element source = xparse("<X><A/><Y/><Z/><A/><Z/></X>");
    ]]>
  </xd:declaration>
  <A xd:script="create source"> /* "source" je zdroj dat pro konstrukci. */
    <B xd:script="occurs 1..*; create xpath('/X/A', source)"/>
    <C xd:script="occurs 1; create xpath('/X/Z[2]', source)"/>
 </A>
```

```
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C210.html

Element A se vytvoří z elementu X (jak již víme, na jménu nezáleží). Elementy "B" se vytvoří ze všech elementů "A" ve zdroji a element "C" z druhého výskytu elementu "Z". Výsledek tedy bude element, jaký jsme požadovali.

Nyní si položme otázku, jak doplnit do výsledného XML dokumentu hodnoty atributů a textových hodnot. Zkusme vytvořit element s textovou hodnotou "Nazdar světe!". Jakou hodnotu budeme potřebovat k vytvoření textového uzlu nebo atributu? Není těžké uhodnout, že budeme potřebovat hodnotu typu string s požadovanou hodnotou:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<A xd:script="create true">
  string; create "Nazdar svete!"
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C211.html

Vidíme, že jsme dodali do create sekce pro tvorbu textového uzlu jednoduše požadovanou hodnotu. Pro atributy bude situace podobná, jako pro textové uzly. Výsledek následující X-definice si čtenář jistě domyslí sám:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<Osoba xd:script="create true"
  Jmeno="string; create 'Jan'"
  Prijmeni="string; create 'Novak'">
  string; create "Prima chlapec."
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C212.html

Dodejme, že pokud se příslušný objekt s hodnotou nemá vytvořit, můžeme dodat jako zdroj dat hodnotu "null" (a atribut "Plat" se nevytvoří):

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<Osoba xd:script="create true"
  Jmeno="string; create 'Jan'"
  Prijmeni="string; create 'Novak'"
  Plat="optional int; create null"/>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C213.html

Uvědomme si, že zdroj dat pro hodnotu atributu nebo textového uzlu je string. Pokud bychom poskytli jako hodnotu zdroje dat nějaký jiný typ, provede se automatická konverze na string (provede se implicitní metoda "toString" - pokud ovšem předkládaná hodnota nebyla null). Lze tedy zapsat:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<Osoba xd:script="create true"
  Jmeno="string; create 'Jan'"
  Prijmeni="string; create 'Novak'"
  Plat="optional int; create 0001234"/>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C214.html

Uvedené vedoucí nuly se konverzí ztratí a výsledek bude:

```
<Osoba Jmeno="Jan" Prijmeni="Novak" Plat="1234"/>
```

Nyní zkusme vytvořit element, jehož hodnoty budou převzaty z elementu, který předáme jako zdroj dat:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
   /* Vytvorime zdroj dat do promenne source. */
   Element source = xparse("<Osoba jmeno='Jan' prijmeni='Novak' plat='1234'/>");
  ]]>
</xd:declaration>

<Osoba xd:script= "create source"
       jmeno    = "string; create xpath('@jmeno', source)"
       prijmeni = "string; create xpath('@prijmeni', source)">
  <Plat xd:script= "create xpath('@plat', source)">
```

```
      string; create xpath('@plat', source);
  </Plat>
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C215.html

Výsledek bude:

```
<Osoba jmeno="Jan" prijmeni="Novak">
  <Plat>
    1234
  </Plat>
</Osoba>
```

Podobně můžeme vytvořit element na základě výsledku databázového příkazu. Nechť máme databázovou tabulku, která obsahuje řádky s hodnotami "jmeno", "prijmeni" a "plat". Tabulku předáme programu v objektu "osoby" typu ResultSet získaného příkazem "query" z databáze. Elementy se vytvoří, pokud jsou k dispozici řádky s příslušnými sloupci (všimněme si, že v parametru metody getItem(…), se ignorují velká a malá písmena, což je vlastnost databázových příkazů):

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>

  /*
   Pristup k databazi je obvykle vytvoren Java prikazem
     XDFactory.createSQLService(url, user, password)
   a predan X-definici pres externi promennou.
  */
  Service service = new Service("jdbc:derby://localhost:1527/sample;", "myself",
"blabla");

  /* Ziskani tabulky z databaze. */
  ResultSet osoby = service.query("SELECT * FROM MYTEST.OSOBA");
</xd:declaration>

<Seznam xd:script= "create true">
    <Osoba xd:script= "occurs *; create osoby"
         jmeno    = "string; create osoby.getItem('JMENO')"
         prijmeni = "string; create osoby.getItem('PRIJMENI')">
      <Plat xd:script= "create osoby.getItem('PLAT') != null">
        string; create osoby.getItem('PLAT');
      </Plat>
    </Osoba>
  </Seznam>

</xd:def>
```

Je zřejmé, že elementy "Osoba" se budou vytvářet jen tehdy, jsou-li řádky v objektu ResultSet k dispozici. Metoda "getItem(jméno)" vrátí string s hodnotou příslušného políčka řádky.

3.1 Přehled typů hodnot z příkazu sekce create

Jak jsme viděli, k vytvoření elementů lze použít různé typy dat, jak je uvedeno v následující tabulce. Připomeňme, že počet prvků je omezen maximem specifikovaným v kvalifikaci.

Tabulka 2- Typy hodnot sekce create v modelech elementů

Hodnota	Vysledek konstrukce
null	creates nothing
element	creates elements
int	if positive creates elements according to the value of number
boolean	elements are created if value is "true"
String	elements are created if value is nonempty string
Container	elements are created if value is nonempty Container.
ResultSet	elements are created from rows of the table fom ResultSet
other type	value is converted to String and behaves as String

Tvorbu atributů nebo textových uzlů způsobí libovolná hodnota která se automaticky převede na neprázdný string a není "null".

4 Skupiny (xd:sequence, xd:choice a xd:mixed)

V následujícím textu se podívejme jak se budou v režimu vytváření chovat skupiny "xd:sequence", "xd:choice" a "xd:mixed". Mějme opět na paměti, že X-definice slouží jako předpis, podle něhož se tvoří výsledek. Každou ze skupin probereme zvlášť.

4.1 xd:sequence

Ukažme si nejdříve konstrukci skupiny "xd:sequence". Skupina sequence prochází postupně všechny položky a provádí konstrukci podle kontextu. Pokud zapíšeme do Skriptu příkaz "create", použije se pro tvorbu prvků této skupiny výsledek tohoto příkazu podobně, jako pro tvorbu potomků elementu. Začněme jednoduchým příkladem:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<A>
  <xd:sequence xd:script= "occurs *; create 2">
     <B xd:script= "create true"/>
     required string; create "Text";
     <C xd:script= "create true"/>
  </xd:sequence>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C311.html

Výsledek bude:

```
<A>
  <B/>
  Text
  <C/>
  <B/>
  Text
  <C/>
</A>
```

Jak čtenář jistě očekával, prvky skupiny se vytvořily dvakrát (v příkazu "create" pro skupinu byl zadáno číslo 2).

Pokud bude zdroj dat pro tvorbu skupiny "xd:sequence" seznam uzlů, který je výsledem výrazu XPath bude situace podobná, jako při generování potomků elementu. Ukážeme si to na
příkladu, který jsme probrali v předešlé kapitole. Nechť tedy je zdroj dat pro vytvoření výsledku:

```
<X>
  <A/>
  <Y/>
  <Z/>
  <A/>
  <Z/>
</X>
```

Nechť element "A" je požadovaný výsledek. Jeho potomci jsou popsáni sekvencí, ve které elementy "B" budou vytvořeny z elementů "A" ze zdroje dat a element "C" bude vytvořen z druhého výskytu elementu "Z". Požadovaný výsledek tedy bude:

```
<A>
  <B/>  (má být vytvořen z prvního "A")
  <B/>  (má být vytvořen z  druhého "A")
  <C/>  (má být vytvořen z  druhého "Z")
</A>
```

Jako zdroj dat pro konstrukci sekvence předáme element "source" v následující X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  /* Vytvoříme element do proměnné source použité jako zdroj dat. */
  Element source = xparse("<X><A/><Y/><Z/><A/><Z/></X>");
```

```
  ]]>
</xd:declaration>

<A>
  <xd:sequence>
    <B xd:script="occurs 1..*; create xpath('/X/A', source)"/>
    <C xd:script="occurs 1; create xpath('/X/Z[2]', source)"/>
  </xd:sequence>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C312.html

4.2 xd:mixed

Podívejme na skupinu "xd:mixed". Skupina mixed prochází postupně všechny varianty, dokud nenajde nějaký odpovídající prvek a dokud nejsou vyčerpány všechny varianty, začíná znova od prvního prvku v modelu.

Mějme X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<A>
  <xd:mixed>
    <B xd:script= "create true" />
    required string; create "Text";
    <C xd:script= "create true" />
    </xd:mixed>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C321.html

Výsledek bude podobný, jako u skupiny "xd:sequence" (modely potomků skupiny se provádějí v pořadí, jak jsou do X-definice zapsány):

```
<A>
  <B/>
  Text
  <C/>
</A>
```

Podívejme se na složitější případ. Zkusíme vytvořit pomocí skupiny "xd:mixed" výsledek z dat, která předáme pomocí elementu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  /* Vytvoříme element do proměnné source pouzite jako zdroj dat. */
  Element source = xparse("<A><C/><B/></A>");
  ]]>
</xd:declaration>

<A xd:script="create 1">
  <xd:mixed xd:script= "create 1">
    <B xd:script="create xpath('B', source)"/>
    <C xd:script="create xpath('C', source)"/>
  </xd:mixed>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C322.html

Upozorněme, že kdybychom zpracovali touto X-definicí vstupní data ve validačním režimu podle elementu, který je v proměnné "source", bylo by pořadí potomků elementu "A" stejné, jako na vstupu:

```
<A>
  <C/>
  <B/>
</A>
```

Ale výsledek v režimu vytváření bude::

```
<A>
  <B/>
  <C/>
</A>
```

Důvodem je, že prvky skupiny "xd:mixed" se zpracovávají v pořadí, ve kterém jsou zapsány v X-definici (proces je řízen zápisem X-definice).

4.3 Výběr prvku - choice

Skupina choice prochází postupně všechny varianty, dokud nenajde nějaký odpovídající prvek. Položme si tedy nakonec otázku, co se stane, když použijeme skupinu "xd:choice":

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<A>
  <xd:choice xd:script= "*; create 2">
    <B xd:script= "create true" />
    required string; create "Text";
    <C xd:script= "create true" />
  </xd:choice>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C331.html

Procesor v tomto případě v každé iteraci (požadovali jsme dvě) vybere první vyhovující variantu (v našem případě element "B") a výsledek bude:

```
<A>
  <B/>
  <B/>
</A>
```

Kdybychom chtěli ovlivnit, kterou variantu má procesor vybrat, museli bychom do Skriptu nějakým způsobem zapsat, která varianta se má vybrat. Např.:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration> int i = 0; </xd:declaration>
<A>
  <xd:choice xd:script= "*; create 3">
    <B xd:script= "create i++ == 0" />
    string; create i++ == 2 ? "Text" : null;
    <C xd:script= "create i++ == 5" />
  </xd:choice>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C332.html

Výsledek pak bude:

```
<A>
  <B/>
  Text
  <C/>
</A>
```

Ponecháme na čtenáři, aby zdůvodnil proč (nápověda: po provedené variantě se další varianty nezpracovávají).

Zkusme nyní vytvořit pomocí skupiny "xd:choice" výsledek z dat, která předáme pomocí elementu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  /* Vytvoříme element do proměnné source pouzite jako zdroj dat. */
  Element source = xparse("<A><C/><B/></A>");
  ]]>
</xd:declaration>
<A>
  <xd:choice>
    <B  xd:script="create xpath('B', source)"/>
    <C  xd:script="create xpath('C', source)"/>
  </xd:choice>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C333.html

Výsledek bude:

```
<A>
```

```
  <B/>
</A>
```

Důvod proč se nevytvoří prvek "C", který je první v posloupnosti potomků, ale prvek "B" je stejný, jako v předešlých případech: procesor zpracovává postupně X-definici a provede jako první model "B".

5 Kontext pro konstrukci XML

V předchozím výkladu jsme důsledně specifikovali data pro tvorbu každého objektu výsledného XML. V zápisu Skriptu jsme pro každý prvek, který má být vytvořen, specifikovali v sekci "create" příkaz, který vrátil hodnotu, podle které procesor má vytvořit výsledek. Nepředpokládali jsme, že procesor X-definic "ví", co jsme udělali v předchozím kroku. Takový způsob zápisu často není nutný. Ve skutečnosti buď příkaz "create" můžeme zcela vynechat (pak se provede defaultní operace), nebo se v příkazech "create" můžeme odkázat na data z předchozí operace, ze kterých pak lze vytvořit celé části výsledného XML objektu. Zdrojová data použitá v konstrukčním režimu v tomto případě nazýváme "kontext" pro tvorbu XML objektů.

Začněme situací, o které jsme se již zmínili, totiž o možnosti vytvořit element z textových hodnot. Ukážeme si variantu, kdy explicitně neuvedeme hodnotu pro zdroj dat textové hodnoty elementu - ta se automaticky převezme z dat zdrojového elementu a stane se kontextem pro tvorbu dalších prvků. Vytvoření elementu "Plat" můžeme zapsat bez specifikace sekce "create" v popisu textové hodnoty uzlu. Element se vytvoří podle pravidla z předchozí kapitoly. Textová hodnota, u které jsme vynechali sekci create, se převezme automaticky z textové hodnoty zdrojového elementu, který se tak stane kontextem pro další zpracování:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  /* Vytvoříme element do proměnné source pouzite jako zdroj dat. */
  Element source = xparse("<A>1234</A>");
  ]]>
</xd:declaration>
<Osoba>
  <Plat xd:script= "create source">
    required int; /* hodnota se vezme z kontextu */
  </Plat>
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C341.html

Protože už víme, že ze stringu se automaticky vytvoří element s potomkem s textovou hodnotou (a ten se stane kontextem), můžeme také napsat X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<Osoba>
  <Plat xd:script = "create '1234'">
    required int;
  </Plat>
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C342.html

Vidíme, že pro tvorbu elementu "Plat" byl vytvořen zdroj dat, který se stal kontextem i pro tvorbu potomků tohoto elementu (v našem případě textové hodnoty elementu).

Nebo pro případ databáze (proměnná "source" je Resultset a metoda "getItem" vrátí string s hodonotou příslušného sloupce):

```
<Plat xd:script= "create source.getItem('PLAT')">
   required int;
</Plat>
```

Jak to bude s atributy elementu? Pokud vynecháme v popisu atributů sekci "create", budou pro konstrukci jednotlivých atributů v modelu použity pojmenované hodnoty z kontextu (např. odpovídající atributy elementu, položky sloupců v řádcích databázové tabulky, pojmenované hodnoty v objektu "Container" atd.). Tedy, pokud uvedeme jako zdroj dat element, který bude mít atributy "jmeno" a "prijmeni", budou hodnoty atributů v generovaném výsledku automaticky převzaty z tohoto kontextu. Z kontextu v následujícím příkladu se vygeneruje i element "Plat". X-definice bude mít tvar:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  Element source = xparse("<X jmeno='Jan' prijmeni='Novak'><Plat>1234</Plat>
  </X>");
  ]]>
</xd:declaration>
<Osoba xd:script= "create source"
       jmeno    = "string;"
       prijmeni = "string;">
  <Plat>
    required int;
  </Plat>
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C343.html

Výsledek bude:

```
<Osoba jmeno='Jan' prijmeni='Novak'><Plat>1234</Plat></Osoba>
```

Pokud bude mít element potomky, které se mají vyrobit ze zdroje dat, který jsme elementu předali, bude pro tvorbu potomků automaticky použit kontext. Kontext se pak opět stane kontextem pro další potomky. Je třeba si uvědomit, že jako zdroj dat pro tvorbu vnitřních elementů dat budou automaticky elementy kontextu se sejným jménem. Pokud se jedná o tvorbu textové hodnoty, bude použita odpovídající textová hodnota kontextu. Ukažme si, jak bude vytvořen element "A", kterému jako zdroj dat předáme element "X". Požadovaný tvar elementu A bude popsán X-definicí:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  Element source =
    xparse("<X a1='a1' a2='a2'>XXX<R a='1'><S>SSS</S></R><Q>QQQ</Q><P>PPP</P></X>");
  ]]>
</xd:declaration>
<A xd:script= "create source"
   a1 = "string;"
   a2 = "string;">
  <P>
    string;
  </P>
  <Q>
    string;
  </Q>
  <R a= "int">
    <S>
      string;
    </S>
  </R>
  string;
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C344.html

Všimněme si, že z kontextu byly vybrány prvky v pořadí, které odpovídá modelu, nikoliv zdroji dat. Výsledek bude (indentace výsledku je použita pro lepší čitelnost):

```
<A a1="a1" a2="a2">
    <P>
      PPP
    </P>
    <Q>
      QQQ
    </Q>
    <R a="1">
       <S>
         SSS
       </S>
    </R>
    XXX
</A>
```

Podívejme se ještě na chování skupin s kontextem.

Jsou-li položky v sekvenční části Containeru opět hodnoty typu Container, použijí se k vytvoření potomků modelu prvku:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
   /* Vytvorime zdroj dat do promenne source typu Container. */
   Container source = [%jmeno='Jan', %prijmeni='Novak', '1234'];
</xd:declaration>

<Osoba xd:script= "create source"
       jmeno    = "string;"
       prijmeni = "string;">
  <Plat>
     string;
  </Plat>
</Osoba>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C345.html

Výsledek bude:

```
<Osoba jmeno="Jan" prijmeni="Novak" ="1234">
  <Plat>
    1234
  </Plat>
</Osoba>
```

Pokud je sekvenčním prvkem Containeru opět Container, je použit pro tvorbu potomků elementu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
   /* Vytvorime zdroj dat do promenne source typu Container. */
   Container source = [%A=[%a='A', [%b='B', 'C'], 'D']];
</xd:declaration>

<A xd:script= "create source"
   a = "string">
  <B b = "string">
     string
  </B>
  string
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C346.html

Výsledek bude:

```
<A a="A">
  <B b="B">
   C
  </B>
  D
</A>
```

5.1 Kontext pro konstrukci skupiny xd:sequence

Zkusme následující příklad X-definice:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[ Element source = xparse("<X><B a='b'/><C a='c'/></X>"); ]]>
</xd:declaration>
<A>
  <xd:sequence xd:script="create source">
    <B a="string" />
    <C a="string" />
  </xd:sequence>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C411.html

Výsledek bude podle očekávání:

```
<A>
  <B a="b"/>
```

```
  <C a="c"/>
</A>
```

Vložme nyní sekvenci vnitřních prvků s kvantifikátorem umožňujícím více opakování vnitřních prvků a doplňme příkaz create tak, aby postupně generoval více sekvencí:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  Element source =
    xparse("<X><Y><B a='b'/><C a='c'/></Y><Y><C a='x'/></Y><Y><B a='y'/></Y></X>");
  ]]>
</xd:declaration>
<A>
  <xd:sequence xd:script="occurs *; create xpath('//Y', source)">
    <B xd:script="occurs ?" a="string" />
    <C xd:script="occurs ?" a="string" />
  </xd:sequence>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C412.html

Čtenář sám jistě snadno vysvětlí, proč výsledek bude (nápověda: výsledkem XPath výrazu jsou všechny elementy "Y", které jsou postupně předány jako kontext pro tvorbu skupiny):

```
<A>
  <B a="b"/>
  <C a="c"/>
  <C a="x"/>
  <B a="y"/>
</A>
```

5.2 Kontext pro konstrukci skupiny xd:choice

Začněme opět jednoduchým příkladem:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
 Element source = xparse("<X><C a='c'/><B a='b'/></X>");
]]>
</xd:declaration>
<A>
  <xd:choice xd:script="create source">
    <B a="string" />
    <C a="string" />
  </xd:choice>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C421.html

Výsledek bude:

```
<A>
  <B a="b"/>
</A>
```

Důvod, proč se vytvoří element "B" a nikoliv "C" spočívá v tom, že proces je řízen modelem v X-definicí. V té je model elementu "B" je popsán dříve, než element "C", který je však ve zdrojových datech první.

Modifikujme náš příklad pro skupinu s kvantifikátorem (occurs *) pro více výskytů obsahu skupiny:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
 Element source = xparse("<X><Y><B a='b'/></Y><Y><C a='x'/></Y><Y><B a='y'/></Y></X>");
]]>
</xd:declaration>
<A>
  <xd:choice xd:script="occurs *; create xpath('//Y',source)">
    <B a="string" />
    <C a="string" />
  </xd:choice>
</A>
```

```
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C422.html

Výsledek bude:

```
<A>
  <B a="b"/>
  <C a="x"/>
  <B a="y"/>
</A>
```

Čtenář jistě snadno vysvětlí proč.

5.3 Kontext pro konstrukci skupiny xd:mixed

Začněme opět jednoduchým příkladem:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  Element source = xparse("<X><C a='c'/><B a='b'/></X>");
  ]]>
</xd:declaration>
<A>
  <xd:mixed xd:script="create source">
    <B a="string" />
    <C a="string" />
  </xd:mixed>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C431.html

Jistě už víte, proč výsledné pořadí prvků ve skupině mixed bude odpovídat zápisu v X-definici a nikoli pořadí v kontextu:

```
<A>
  <B a="b"/>
  <C a="c"/>
</A>
```

Modifikujme podobně jako v předešlém odstavci náš příklad pro skupinu mixed:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
  <![CDATA[
  Element source =
    xparse("<X><Y><B a='b'/><C a='c'/></Y><Y><C a='x'/></Y><Y><B a='y'/></Y></X>");
  ]]>
</xd:declaration>
<A>
  <xd:mixed xd:script="create xpath('//Y',source)">
    <B a="string" />
    <C a="string" />
  </xd:mixed>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C432.html

Výsledek tentokrát bude:

```
<A>
  <B a="b"/>
  <C a="c"/>
  <C a="x"/>
  <B a="y"/>
</A>
```

Oproti skupině choice je tentokrát výsledek ve stejném pořadí, jako na vstupu.

6 Práce s výsledkem výrazu XPath (metoda "from")

Při tvorbě výsledného elementu často potřebujeme vybrat určitý objekt z kontextu a případně nastavit nový kontext pro potomky vytvářeného objektu.

V příkladech v předešlém textu jsme ukázali možnost použití výrazu XPath provedeném na XML datech. Je-li kontextem pro model elementu opět XML element, můžeme použít metodu "from", která provede XPath výraz nad aktuálním kontextem. Nastaveni nového kontextu pomocí metody "from" platí také pro všechny potomky vytvořené tímto příkazem. Ukažme si chování na příkladu. Mějme X-definici popisující element, který vytváříme z různých částí kontextu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
  Element source = xparse("<X a='a'><Y a='x'><Z a='b'/></Y><Y a='y'><Z
a='c'/></Y></X>");]]>
</xd:declaration>
<A xd:script="create source"> /* kontext s root elementem X */
  <B xd:script="create from('@a')"> /* kontext s atributem "a" elementu X */
    string; create from('text()'); /* vezme se hodnota tohoto atributu */
  </B>
  <C xd:script="occurs *; create from('Y/Z')"> /* posloupnost elementů Z */
    string; create from('@a'); /* jejich atributy "a" se použijí pro textovou hodnotu*/
  </C>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C501.html

Na začátku se nastaví kontext pro tvorbu elementu "A" na kořen zdrojového elementu. Pro tvorbu elementu "B" se použije atribut "a" z nastaveného kontextu a jeho hodnota se stane kontextem pro tvorbu elementu "B". Jak jsme již probrali v předešlém textu, z textové hodnoty předané metodou "from" (tedy hodnoty atributu "a" aktuálního kontextu) se vytvoří element s textovou hodnotou. Také už víme, že bychom mohli v tomto případě sekci "create" vynechat. Metoda "from" pro tvorbu elementů "C" vrátí všechny uzly, které odpovídají cestě "X/X" z aktuálního kontextu. Textová hodnota těchto uzlů se převezme z hodnot atributu "a" těchto uzlů. Výsledek tedy bude:

```
<A>
  <B>a</B>
  <C>b</C>
  <C>c</C>
</A>
```

Ukažme si X-definici, ve které hodnoty zapíšeme ve zdrojových datech do textu elementů místo do atributů Výsledek bude stejný jako v přechozí variantě. Všimněte si, že můžeme pro tvorbu textu elementů vynechat sekci "create":

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
  Element source = xparse("<X>a<X><X>b</X></X><X><X>c</X></X></X>");
]]>
</xd:declaration>
<A xd:script="create source">
  <B xd:script="create from('text()')">
   string;
  </B>
  <C xd:script="occurs *; create from('X/X')">
    string;
  </C>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C502.html

A naopak, z textových uzlů můžeme na vstupu vytvořit atributy na výstupu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
  Element source = xparse("<X>a<X><X>b</X></X><X><X>c</X></X></X>");
]]>
</xd:declaration>
```

```
<A xd:script="create source">
  <B xd:script="create from('text()')" a="string; create from('text()')"/>
  <C xd:script="occurs *; create from('X/X')" a="string; create from('text()')"/>
</A>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C503.html

Výsledek bude:

```
<A>
  <B a="a"/>
  <C a="b"/>
  <C a="c"/>
</A>
```

Zkusme se podívat na trochu složitější příklad z reálného světa. Nechť vstupní data jsou v následujícím elementu:

```
<Položka id="0123456789">
  <Klient role="1" název="Nějaká Firma s.r.o." IČO = "12345678" />
  <Klient role="2" jméno="František" příjmení= "Kovář" RČ = "311270/1234" />
  <Klient role="3" jméno="Jan" příjmení="Bílý" RČ="120455/2345" IČO="87654321" />
</Položka>
```

Z těchto dat vytvoříme element "Smlouva", v němž bude v atributu "číslo" převzato číslo z atributu "id" elementu "Položka". Navíc v něm vytvoříme atribut "datum", do kterého uložíme aktuální datum a čas převzatý z operačního systému. Vnitřní elementy "Vlastník", "Klient" a "Držitel" vytvoříme z elementů "Klient" v závislosti na hodnotě atributu "role". Element "Klient" vytvoříme, pokud atribut "role" má hodnotu "1" (všimněte si, že jsme v zápisu apostrofu uvnitř stringu použili zpětné lomítko). U atributů se stejným jménem jako v kontextu můžeme sekci "create" vynechat. Element "Držitel" vytvoříme z elementu "Klient"pokud hodnota atributu "role" je "2" a další atributy opět převezmeme. Element "Prostředník" vytvoříme z elementu "Klient" pokud atribut "role" má hodnotu "3" a atribut "jméno" složíme z hodnot atributů "jméno" a "příjmení" v kontextu. Atribut "IČO" opět převezmeme (atribut "RČ" nepoužijeme). X-definice, která provede tuto transformaci, bude mít tvar (všimněte si, že ve Skriptu lze string deklarovat na více řádcích):

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1">
<xd:declaration>
<![CDATA[
  Element source = xparse(
"<Data id='0123456789'>
   <Klient role='1' název='Nějaká Firma s.r.o.' IČO = '12345678' />
   <Klient role='2' jméno='František' příjmení= 'Kovář' RČ = '311270/1234' />
   <Klient role='3' jméno='Jan' příjmení='Bílý' RČ='120455/2345' IČO='87654321' />
</Data>");
]]>
</xd:declaration>
<Smlouva xd:script = "create source"
  datum = "required dateTime(); create now()"
  číslo = "required num(10); create from('@id')" >
  <Vlastník xd:script = "occurs 1; create from('Klient[@role=\'1\']')"
    IČO   = "required num(8)"
    název = "required string(1,30)" />
  <Držitel xd:script = "create from('Klient[@role=\'2\']')"
    RČ       = "required string(10,11) "
    jméno    = "required string(1,30)"
    příjmení = "required string(1,30)" />
  <Prostředník xd:script = "create from('Klient[@role=\'3\']')"
    název = "required string(1,30); create from('@jméno') + ' ' + from('@příjmení')"
    IČO   = "required num(8)" />
</Smlouva>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C504.html

Výsledek bude:

```
<Smlouva datum="2012-10-7T17:05:00" číslo="0123456789">
  <Vlastník název="Nějaká Firma s.r.o." IČO="12345678"/>
  <Držitel RČ="311270/1234" jméno="František" příjmení="Kovář"/>
  <Prostředník IČO="87654321" název="Jan Bílý"/>
</Smlouva>
```

7 Kombinace režimu validace a režimu konstrukce

Oba režimy X-definic, tj. režim validace a konstrukční režim lze kombinovat. Použijme předchozí příklad, ale nejdříve validujme data podle modelu "Data" (poznamenejme, že metoda "xparse" má druhý parametr, ve kterém je buď jméno X-definice, nebo "*" pokud X-definice nemá jméno). Při provádění příkazů v deklarační sekci se ověří vstupní data z proměnné "data" podle modelu "Data" (která musí být v X-definici samozřejmě deklarována jako "root") Výsledkem je validovaný element "Data" a je použit jako kontext pro model "Smlouva" v konstrukčním režimu podobně jako v předchozím příkladu:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1" root="Item">
<xd:declaration>
<![CDATA[
  String data =
"<Item id='0123456789'>
  <Client role='1' company='NSomeCompany Ltd' cId = '12345678' />
  <Client role='2' name='Peter' surname= 'Brown' pId = '700231/1234' />
  <Client role='3' name='John' surname='Smith' pId='551204/2345' cId='87654321' />
</Item>";
  Element source = xparse(data, "*");
]]>
</xd:declaration>

<Contract xd:script = "create source"
     date = "required dateTime(); create now()"
     number = "required num(10); create from('@id')" >
   <Owner xd:script   = "create from('Client[@role=\'1\']')"
       cId     = "required num(8)"
       company = "required string(1,30)" />
   <Holder xd:script = "create from('Client[@role=\'2\']')"
       pId     = "required string(10,11) "
       name    = "required string(1,30)"
       surname = "required string(1,30)" />
   <Policyholder xd:script = "create from('Client[@role=\'3\']')"
       name = "required string(1,30); create from('@name') + ' ' + from('@surname')"
       cId   = "required num(8)" />
</Contract>

<Item id='num(10)'>
  <Client role = 'int' company = 'string' cId = 'num(8)' />
  <Client role = 'int' name = 'string' surname = 'string' pId = 'string' />
  <Client role = 'int' name = 'string' surname ='string' pId = 'string' cId = 'string' />
</Item>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C601.html

Režim zpracování a konstrukční režim můžeme propojit i v opačném pořadí. Nejdříve provedeme zpracování zdrojového XML dokumentu ve validačním režimu a nakonec (v sekci "finally" modelu "Data" popisujícím validovaná data) vytvoříme výsledek pomocí metody "xcreate", která využije zpracovaný XML dokument jako kontext. Všimněte si, že v modelu "smlouva" byla vynechána sekce "create", protože jako kontext je automaticky nastaven dokument ze zpracovaných dat. Výsledek je nastaven pomocí metody "setResultElement" v sekci "finally". Vstupní data se dodají procesoru obvyklým způsobem. Ukázka je v následujícím příkladě:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1" root = "Data">

<Data id='num(10)' xd:script='finally returnElement(xcreate("Smlouva"))'>
  <Klient role = 'int' název = 'string' IČO = 'num(8)' />
  <Klient role = 'int' jméno = 'string' příjmení = 'string' RČ = 'string' />
  <Klient role = 'int' jméno = 'string' příjmení = 'string' RČ ='string' IČO = 'string'
/>
</Data>

<Smlouva datum = "required datetime; create now()"
  číslo = "required num(10); create from('@id')" >
  <Vlastník xd:script   = "create from('Klient[@role=\'1\']')"
    IČO   = "required num(8)"
    název = "required string(1,30)" />
  <Držitel xd:script = "create from('Klient[@role=\'2\']')"
    RČ       = "required string(10,11) "
    jméno    = "required string(1,30)"
    příjmení = "required string(1,30)" />
```

```
  <Prostředník xd:script = "create from('Klient[@role=\'3\']')"
    název = "required string(1,30); create from('@jméno') + ' ' + from('@příjmení')"
    IČO   = "required num(8)" />
</Smlouva>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C602.html

Nakonec uveďme příklad, ve kterém se vytvoří z několika měření větru a teploty výsledek ve tvaru html. Ve výsledku se na základě zpracování vstupních dat zobrazí naměřené denní údaje a průměrná teplota. Ukážeme, že během zpracování lze připravit hodnoty pro tvorbu výsledku (viz proměnné "sum" a "num" potřebné k výpočtu průměrné teploty).

Vstupní data mají tvar:

```
<Weather date = "2005-05-11" >
   <Measurement wind = "5.3" temperature = "13.0" time = "05:00" />
   <Measurement wind = "7.2" temperature = "15.2" time = "11:00" />
   <Measurement wind = "8.7" temperature = "18.1" time = "15:00" />
   <Measurement wind = "3.9" temperature = "16.5" time = "20:00" />
</Weather>
```

Provedeme-li následující X-definici:

```
<xd:def xmlns:xd = "http://www.syntea.cz/xdef/3.1" root = "Weather">
<xd:declaration>
  float sum = 0; int num = 0;
</xd:declaration>

<Weather xd:script="finally returnElement(xcreate('html'))"
   date = "optional date ()">
   <Measurement xd:script = "occurs +"
       wind = "required float(0, 150)"
       temperature = "required float(-40,60) /*limit of temperature*/
                     onTrue {num++; sum += getParsedFloat()}"
       time = "required time()" />
</Weather>

<html>
  <h3>string; create "Weather on " + from("@date")</h3>
  <li xd:script = "occurs +; create from('Measurement')">
    string; create "Time: " + from("@time") + ", wind: " + from("@wind") +
                   ", temperature: " + from("@temperature")
  </li>
  <h3 style="fixed 'background: yellow'">
    string; create num==0 ? "No data" : "Average temperature: " + sum/num
  </h3>
</html>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C603.html

Dostaneme výsledek:

```
<html>
  <h3>Weather on 2005-05-11</h3>
  <li>Time: 05:00:00, wind: 5.3, temperature: 13.0</li>
  <li>Time: 11:00:00, wind: 7.2, temperature: 15.2</li>
  <li>Time: 15:00:00, wind: 8.7, temperature: 18.1</li>
  <li>Time: 20:00:00, wind: 3.9, temperature: 16.5</li>
  <h3 style="background: yellow">Average temperature: 15.7</h2>
</html>
```

8 Template model

Chcete-li vytvořit XML data, ve kterých jsou konstantní hodnoty můžeme použít "template" model. Ten vznikne tak, že do Skriptu modelu zapíšeme klíčové slovo "template". Příkaz "template" způsobí, že se hodnoty textových uzlů a atributů z modelu převedou na Skript, v němž tyto hodnoty jsou deklarovány jako konstanty. Pokud potřebujeme na nějakém místě použít v modelu proměnnou hodnotu, musíme použít Skript. Docílíme to tak, že zapíšeme na místo textové atributu nebo textového uzlu hodnoty klíčové slovo "**$$$script:**". Text za tímto zápisem se pak zpracuje obvyklým způsobem jako Skript: Např.

```
<xd:def xmlns:xd='http://www.syntea.cz/xdef/3.1' root='html'
        xmlns='http://www.w3.org/1999/xhtml'>
```

```
<html xd:script="template">
  <head>
    <title>Datum</title>
      <meta http-equiv="Content-Type" content="text/html; charset=windows-1250"/>
  </head>
  <body bgcolor="yellow">
    <h1>Dnešní datum:</h1>
    <h2>$$$script: create now().toString('{L(cs,CZ)}EEEE, d. MMMM yyyy GG, hh:mm a')</h2>
    <h3>Toto je ukázka template modelu v kompozičním režimu.</h3>
  </body>
</html>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C701.html

Obsah textových hodnot template modelu se převede automaticky do Skriptu.

Template model z naší ukázky vnitřně vytvoří následující model:

```
<html>
  <head>
    <title>
      fixed "Datum";
    </title>
    <meta http-equiv="fixed 'Content-Type';"
          content="fixed 'text/html;';"
          charset="fixed 'windows-1250';"/>
  </head>
  <body bgcolor="fixed 'yellow';">
    <h1>
      fixed"Dnešní datum:";
    </h1>
    <h2>
      create now().toString('{L(cs,CZ)}EEEE, d. MMMM yyyy GG, hh:mm a')
    </h2>
    <h3>
      fixed"Toto je ukázka template modelu v kompozičním režimu.";
    </h3>
  </body>
</html>
```

Při spuštění konstrukčního režimu za zápisem "$$$script:" je hodnota vytvořena sekcí "create". Výsledek bude:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Datum</title>
  </head>
  <body bgcolor="yellow">
    <h1>Dnešní den je:</h1>
    <h2>Čtvrtek, 4. říjen 2012 po Kr., 11:38 dop.</h2>
    <h3>Toto je ukázka template modelu v kompozičním režimu.</h3>
  </body>
</html>
```

9 Example

Ukažme na závěr příklad, kde zadáme vstupní data jako kontext a vygenerujeme z nich html soubor. Mějme vstupní data se seznamem panovníků v českých zemích:

```
<panovnici>
  <panovnik>
    <jmeno>Tomáš Garrigue Masaryk</jmeno>
   <panoval titul="prezident">
      <od>1918</od>
      <do>1935</do>
    </panoval>
  </panovnik>
  <panovnik>
    <jmeno>Miloš Zeman</jmeno>
    <panoval titul="prezident">
      <od>2013</od>
    </panoval>
  </panovnik>
  <panovnik rod="přemyslovec">
```

```
    <jmeno>Václav I.</jmeno>
    <panoval titul="král český">
      <od>1230</od>
      <do>1253</do>
    </panoval>
  </panovnik>
  ...
</panovnici>
```

Některé části X-definice lze popsat jako template model – viz element "head" a první výskyt elementu "tr" (nadpis tabulky). Ostatní data budeme generovat ze vstupních dat.

Všimněte si, že údaje z kontextu o panovnících – tj. výsledek metody from("//panovník") – setřídíme podle začátku vlády (není-li údaj o konci vládnutí vyplněn, pak je údaj prázdný).

```
<xd:def xmlns:xd='http://www.syntea.cz/xdef/3.1' xmlns='http://www.w3.org/1999/xhtml'>
<html xd:script='template'>
   <head>
      <title>Panovníci českých zemí</title>
   </head>
   <body>
     <table border="create 'border'">
       <tr xd:script='template'>
         <td>Panovník</td>
         <td>vládl od</td>
         <td>vládl do</td>
         <td>po dobu</td>
       </tr>
       <tr xd:script='$$$script: string(); occurs *; create from("//panovnik").sort("")'>
         <td>create from('jmeno/text()')</td>
         <td>create from('panoval/od/text()')</td>
         <td>optional string(); create from('panoval/do/text()')</td>
         <td>optional string();
             create from('panoval/do/text()').isEmpty() ? null /*nevyplneno*/
               : from('number(panoval/do/text()) - number(panoval/od/text())');
         </td>
       </tr>
     </table>
   </body>
</html>
</xd:def>
```

http://xdef.syntea.cz/tutorial/cs/example/C711.html

Výstup bude:

```
<html xmlns='http://www.w3.org/1999/xhtml'>
  <head><title>Panovníci českých zemí</title></head>
  <body>
    <table border="border">
      <tr>
        <td>Panovník</td>
        <td>vládl od</td>
        <td>vládl do</td>
        <td>po dobu</td>
      </tr>
      <tr>
        <td>Václav I</td>
        <td>1230</td>
        <td>1253</td>
        <td>23</td>
      </tr>
      ...
      <tr>
        <td>Miloš Zeman</td>
        <td>2013</td>
        <td/>
        <td/>
      </tr>
    </table>
  </body>
</html>
```