```
;; The first four lines of this file were added by Dracula.
;; They tell DrScheme that this is a Dracula Modular ACL2 program.
;; Leave these lines unchanged so that DrScheme can properly load this file.
#reader(planet "reader.ss" ("cce" "dracula.plt") "modular" "lang")
#| Team Steele
   Software Engineering I
   Txmlminidom

   Software that creates a document object model from XML input.
|#

(require "../interfaces/Ixmlminidom.lisp")
(require "../modules/Mxmlminidom.lisp")
(require "../modules/Mbasiclex.lisp")

(module Txmlminidom
  (import Ixmlminidom)

  (include-book "testing" :dir :teachpacks)
  (include-book "doublecheck" :dir :teachpacks)
  (include-book "audio" :dir :teachpacks)

  (play-wav "ragtime.wav" t)

  (defconst
    *Oface*
    "<bob><slidell id=\"porter\">12 &amp; 3<jumptoconclusions /></slidell></bob>")

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ; Functions to generate an xmlminidom tree
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (defun normalize-text (nodes)
    (if (< (length nodes) 2)
        nodes
        (let ((a (car nodes))
              (b (cadr nodes))
              (rest (cddr nodes)))
          (if (and
               (equal (car a) 'text)
               (equal (car b) 'text))
              (normalize-text
               (cons (mv 'text nil
                         (string-append (caddr a) (caddr b))) rest))
              (cons (if (equal (car a) 'text)
                        a
                        (mv (car a) (cadr a) (normalize-text (caddr a))))
                    (normalize-text (cons b rest)))))))

  (defrandom randomxmltext (min)
    (if (equal min 0)
        ""
        (string-append
         (coerce (list (code-char
                        (random-case
                         (random-between 33 46)
                         (random-between 48 60)
                         62
                         (random-between 64 126)
                         ))) 'string)
         (randomxmltext (- min 1)))))

  (defrandom randomattribute ()
    (mv (randomxmltext (random-between 1 30)) (random-string)))

  (defrandom randomnode (maxdepth)
    (random-case
     (mv 'text nil (randomxmltext (random-between 1 30)))
     (mv
      (randomxmltext (random-between 1 30))
      (random-list-of (randomattribute) :size (random-between 0 10))
      (normalize-text (random-list-of
       (randomnode (- maxdepth 1)) :size (random-between 0 maxdepth))))))

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ; Property to test if xml is invertible!
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (defproperty xml-readnode-serialize-dom-invertible-property :repeat 500
    (x :value (randomnode 5))
    (equal x (xml-readnode (xml-serialize-dom x))))

  (check-expect
   (xml-getattribute
    (xml-getnode
     (xml-readnode *Oface*)
     "slidell") "id")
   "porter")

  (check-expect
   (xml-gettext
    (xml-readnode *Oface*))
   "12 & 3")

  (check-expect
   (xml-escape (coerce "Bob & Jane's xml quote was, \"<hello />\"" 'list))
   "Bob &amp; Jane&apos;s xml quote was, &quot;&lt;hello /&gt;&quot;")

  (check-expect
   (xml-serialize-attributes
    (list
     (mv "name" "bob")
     (mv "age" "80")
     (mv "quote" "Bob & Jane's xml quote was, \"<hello />\"")))
   (concatenate 'string
                " name=\"bob\""
                " age=\"80\""
                " quote=\"Bob &amp; Jane&apos;s xml quote was, "
                "&quot;&lt;hello /&gt;&quot;\""))

  (check-expect
   (xml-serialize-dom (mv "Bob" nil nil))
   "<?xml version=\"1.0\"?><Bob/>")

  (check-expect
   (xml-serialize-dom
    (mv "Bob" nil
        (list
         (mv "Joe" nil nil)
         (mv 'text nil " hello ")
         (mv "Poop"
             (list
              (mv "type" "runny")
              (mv "where" "toilet")) nil))))
   (concatenate 'string
    "<?xml version=\"1.0\"?><Bob><Joe/> hello "
    "<Poop type=\"runny\" where=\"toilet\"/></Bob>"))

  (defconst *t1* (mv "bob" (list (mv "a" "b")) nil))
  (check-expect (xml-readnode (xml-serialize-dom *t1*)) *t1*)

  )

(link Test
      (Mbasiclex Mxmlminidom Txmlminidom))
```

```
(invoke Test)
```