```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; server-email.lisp
;
; Created on February 17, 2013 by Wesley R. Howell.
;
; Purpose:
; This file will execute the email transition between clients. In order
; for this to happen, the incoming email will need to be opened, then the
; to field will need to be parsed into a list. Once this list is generated
; we can then send a copy of the email message to the receipent clients.
;
; CHANGE LOG:
; 2/27/2013 - Fixed the IO issue and now we can Call an XML file and
;           Output a file from the same funciton.
;             Note! This only pull the first message in the XML file!
; 2/26/2013 - Updated references to be relative instead of absolute for
;             xml-scanner.lisp, io-utilities.lisp and list-utilities.lisp.
; 2/22/2013 - Finished the email input from file to file out
; 2/21/2013 - Worked on parsing XML documents and getting the email information
;             From the XML file
; 2/19/2013 - Moved from ./modules/user to ./modules/email sub directory.
; 2/18/2013 - Added to SVN Repo
; 2/18/2013 - Updated
; ---------------------------------------------------------------------
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(in-package "ACL2")
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Base Lines
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(defun break-at-set (delimiters xs)
  (if (or (not (consp xs))
          (member-equal (car xs) delimiters))
      (list '() xs)
      (let* ((first-thing (car xs))
             (break-of-rest (break-at-set delimiters (cdr xs)))
             (prefix (car break-of-rest))
             (suffix (cadr break-of-rest)))
          (list (cons first-thing prefix) suffix))))

(defun break-at (delimiter xs)
   (break-at-set (list delimiter) xs))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defun getEmailXML (email)
  (if (endp email)
      nil
      (let* ((hd (concatenate 'string
                              "<?xml version='1.0'?>"
                              "<!DOCTYPE user SYSTEM '../../../dtd/messages.dtd'>"
                              "<email>"))
             (to (concatenate 'string
                              "<to>" (car (car (car email))) "@"
                              (car (cdr (car (car email))))
                              "</to>"))
             (from (concatenate 'string
                                "<from>" (car (cadr email)) "@"
                                (car(cdr (cadr email)))"</from>"))
             (sub (concatenate 'string
                               "<subject>" (cadr (cdr email)) "</subject>"))
             (msg (concatenate 'string
                               "<content>" (cadr (cdr (cdr email)))
                               "</content>"))
             (ft (concatenate 'string "</email>"
                              )))
              (list hd to from sub msg ft))))
```

```
;;getContactStructure string
;;This function will generate a contact structure
;;The delimiter right now is set at "," but can easily be changed to
;;the "@" symbol
;;string - the string to get the contact structure from
(defun getContactStructure (str)
  (let* ((chrs (coerce str 'list))
         (name (car (break-at #\@ chrs)))
         (domain (cdr (break-at #\@ chrs))))
        (list  (coerce name 'string)
               (coerce (cdr (car domain))
                       'string))))

;;getEmailStructure file
;;This function will generate the email data structure based on the
;;FIRST email message in the tokenized XML file list.
;;file - the xml file on the computer to parse
(defun getEmailStructure (xml)
  (let* (;(xml (cdr (getEmailXMLTokens file)))
         (to (car (car (cdr (cdr xml)))))
         (from (car (car (cdr (cdr (cdr (cdr (cdr xml))))))))
         (sub (car (car (cddddr  (cddddr xml)))))
         (msg (car (car (cddddr (cddddr (cdddr xml)))))))
    (list (list (getContactStructure to))
                (getContactStructure from) sub msg)))


;(getEmailStructureList xml)
;This funciton will take the email XML tags and create a list
;of email messages.
;Warning! - this requires the xml format to the exact format as stated
;on the project wiki
;xml - the Tokenized list of xml tags
(defun getEmailStructureList (xml)
      (list (getEmailStructure xml)
      )
)

;getEmail file state
;;;Entry Point;;;
;This function's goal is to open an email XML file,
;parse the xml file and output each individual message to a file.
;The funciton below, runEmailOut, should be the recusive call for this
;function
;file - the file path to the file to open
;;state - the ACL2 state
(defun getEmail (str)
   (getEmailXML (getEmailStructure str))
   )
```