```
!
! (c) Copyright 2006-%(copyright-year)%, Sage Software Canada Ltd. (Ontario, Canada)
! $Id: set_copyright_year.pvc.pvt,v 1.3 2008/02/05 19:44:30 fred.mcguirk Exp $
/*
 * Build information:
! ** @Author Fred McGuirk
! ** @Date Nov 22, 2007
! ** @OutputFile set_copyright_year.pvc

! ** This observer will change the value for copyright year prior to creating a new
! ** build of an application.
! **
! ** This program searches for '%(copyright-year)%' within the first 100 characters
! ** of the source file and will replace it with the current year.  It will then
! ** create a temporary file to be used as an alternative source file which will be
! ** deleted during the PostProcess of the build by this same observer.
 */

def class "set_copyright_year"
    like "EventManagerObserver"

    ! ** Private property to maintain the name of the temporary file that is created
    ! ** during the Pre-Process so that it can be deleted during the Post-Process.
    local tempSourceFile$

    ! ** This method is used to describe the purpose of this observer.  The
    ! ** text will be placed on the observers preference page to describe and
    ! ** should be limited to 80 characters (at most).
    function getDescription$()              getDescription

/*
    ! ** Get the flag that will be used to trigger the 'update()' method during
    ! ** execution of the built-in events for the plug-in.
    ! **
    ! ** If this observer is not a Post-process observer, the return value of
    ! ** this method should be set to one of the following values:
    ! ** @value _pvxConstants'_ideNone           Do not register as an observer
    ! ** @value _pvxConstants'_idePreProcess     Register: Pre-Process observer
    ! ** @value _pvxConstants'_idePrePostProcess Register: Pre/Post-Process observers
    ! ** @value _pvxConstants'_idePostProcess    Register: Post-Process observer
 */
    function getEventNotificationFlag()     getEventNotificationFlag
/*
    ! ** The logic to be executed when the observer is triggered.  This logic must
    ! ** check the major and minor codes to determine the current event and then
    ! ** decide what action is to be performed.
    ! **
    ! ** If this observer is set to watch both Pre-Process and Post-Process states
    ! ** for events, the logic in the 'update' method must check the state of the
    ! ** _pvxState'getArgument(_pvxConstants'_iEventNotificationFlag$) flag to
    ! ** determine the appropriate code to be executed.
    ! **
    ! ** @param state A reference to an object of class %PvxClass(PvxState)%
 */
    function update(initPvxState)           update
end def

getDescription:
    ! Set the description of this observer
    local theDescription$= \
        "Substitute modification year of source file for 'Copyright-year' tag."
return theDescription$
```

```
! Register as both a PreProcess and PostProcess observer
getEventNotificationFlag:
return _pvxConstants'_idePrePostProcess

/*
 * The update routine will be executed by the event manager during the
 * pre/post process logic depending on how this observer registers
 * itself through the getEventNotification() method.
 */
update:
enter aPvxState

    local psMajor$,psMinor$,source,dest$,enfState,x

    ! Get the Major/Minor codes that identify the current action
    psMajor$=aPvxState'getMajor$(), \
    psMinor$=aPvxState'getMinor$()

    /*
     * This observer is for the program build event only; all other events
     * should be ignored.
     */
    if (psMajor$=_pvxConstants'Incremental_Build$ \
            or psMajor$=_pvxConstants'Incremental_Build_Alt_Exe$) \
        and psMinor$=_pvxConstants'BuildType_BuildOne$ {

        ! Check the current state of processing for the ProvideX Event Manager
        enfState=aPvxState'getArgumentValue(_pvxConstants'_iEventNotificationFlag$)
        switch enfState
        case _pvxConstants'_idePreProcess

            ! Get the arguments from the current action that this logic will use
            reqSocket$=aPvxState'getArgumentValue$(_pvxConstants'Request_Socket$), \
            source$=aPvxState'getArgumentValue$(_pvxConstants'SrcFile$), \
            dest$=aPvxState'getArgumentValue$(_pvxConstants'Dest$)

            /*
             * Read the source file, check the first 100 characters for the
             * copyright year tag, replace with the current year, create a temporary
             * file and write the new source into it.
             */
            open input (hfn,isz=1,err=*next)source$;
            sourceFN=lfo

            if sourceFN {
                fileSize=num(fin(sourceFN,"records_used")), \
                utcMtime=num(fin(sourceFN,"UTC_MTime"),err=*next)

                ! Use year from modification time of source file
                if utcMtime>0 then \
                    utcMtime=int(utcMtime/86400)
                read (sourceFN,siz=fileSize+1,err=*next)theFile$
                close (sourceFN);
                sourceFN=0

                if not(nul(theFile$)) {
                    ! Check the first 1000 characters of the source file for the tag
                    found=pos("%(copyright-year)%"=lcs(mid(theFile$,1,1000)))
                    if found {

                        ! Substitute the modification year in place of the tag
                        theFile$= \
                            mid(theFile$,1,found-1)+dte(utcMtime:"%Y") \
```

```
                        +mid(theFile$,found+18)

                    /*
                     * Get a temporary file to be used as new source
                     */
                    x=new("pvx_utility")
                    tempSourceFile$=x'getWorkFile$("WRK_set_copyright_date")
                    drop object x

                    ! Create a new temporary file
                    serial tempSourceFile$,err=*next;
                    open lock (hfn,err=*next)tempSourceFile$;
                    tempFN=lfo

                    ! Update the information for the temporary source file
                    if tempFN {
                        ! Write the information to the temporary source file
                        write record (tempFN,err=*next)theFile$

                        ! Update task state information to override the source file
                        aPvxState'setArgument(_pvxConstants'SrcFile$,tempSourceFile$)
                        close (tempFN)
                    }
                }
            }
        }
        break
    case _pvxConstants'_idePostProcess
        ! erase the temporary source file
        if not(nul(tempSourceFile$)) {
            erase tempSourceFile$,err=*next
            tempSourceFile$=""
        }
        break
    end switch
}
return 0
end
```