UNIVERSITA' DEGLI STUDI DEL PIEMONTE ORIENTALE
SEDE DI ALESSANDRIA - FACOLTA' DI SCIENZE M.F.N.

CORSO DI LAUREA IN SCIENZE DELL'INFORMAZIONE

DOCUMENTAZIONE DI XMLSTUDIO:

EDITOR INTEGRATO PER DOCUMENTI XML

Sviluppo di un editor Java per l'inserimento di informazioni semantiche a documenti testuali secondo lo standard del linguaggio XML

Tesi di Laurea	**Relatore**	**Controrelatore**
di Paolo Guagliumi	**Prof.ssa Paola Giannini**	**Prof.Luigi Portinale**
pguagliumi@libero.it	**giannini@di.unito.it**	**portinal@unipmn.it**

I SESSIONE - ANNO ACCADEMICO 2001/2002

INDICE

DOCUMENTAZIONE JAVADOC

Class AboutBox

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Dialog
                          |
                          +--javax.swing.JDialog
                                |
                                +--AboutBox
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants, java.awt.event.WindowListener

---

class **AboutBox**
extends javax.swing.JDialog
implements java.awt.event.ActionListener, java.awt.event.WindowListener

---

Nested Class Summary

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

AboutBox(javax.swing.JFrame parent)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e)
void	**windowActivated**(java.awt.event.WindowEvent event)
void	**windowClosed**(java.awt.event.WindowEvent event)
void	**windowClosing**(java.awt.event.WindowEvent event)
void	**windowDeactivated**(java.awt.event.WindowEvent event)
void	**windowDeiconified**(java.awt.event.WindowEvent event)
void	**windowIconified**(java.awt.event.WindowEvent event)
void	**windowOpened**(java.awt.event.WindowEvent event)

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll,

processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

AboutBox

```
public AboutBox(javax.swing.JFrame parent)
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent event)
```

Specified by:

`windowClosing` in interface `java.awt.event.WindowListener`

---

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent event)
```

Specified by:

`windowIconified` in interface `java.awt.event.WindowListener`

---

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent event)
```

Specified by:

`windowDeiconified` in interface `java.awt.event.WindowListener`

---

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent event)
```

Specified by:

`windowClosed` in interface `java.awt.event.WindowListener`

---

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent event)
```

Specified by:

`windowDeactivated` in interface `java.awt.event.WindowListener`

---

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent event)
```

Specified by:

`windowActivated` in interface `java.awt.event.WindowListener`

---

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent event)
```

Specified by:

`windowOpened` in interface `java.awt.event.WindowListener`

Class CopiaCorretta

```
java.lang.Object
  |
  +--CopiaCorretta
```

---

class **CopiaCorretta**
extends java.lang.Object

Constructor Summary

`CopiaCorretta(java.io.File origineFile)`

Method Summary

private boolean	`LineaNonVuota(java.lang.String lineaInput)`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

CopiaCorretta

```
public CopiaCorretta(java.io.File origineFile)
```

Method Detail

LineaNonVuota

```
private boolean LineaNonVuota(java.lang.String lineaInput)
```

Class CreaDialog

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Dialog
                          |
                          +--CreaDialog
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, java.awt.event.WindowListener

---

class **CreaDialog**
extends java.awt.Dialog
implements java.awt.event.ActionListener, java.awt.event.WindowListener

---

Nested Class Summary

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary	
(package private) javax.swing.JButton	**cancel**
(package private) static java.lang.String	**input**
(package private) static java.lang.String	**input2**
(package private) static java.lang.String	**input3**
(package private) static java.lang.String	**input4**
(package private) javax.swing.JTextField	**jtesto1**
(package private) javax.swing.JTextField	**jtesto2**
(package private) javax.swing.JTextField	**jtesto3**
(package private) javax.swing.JTextField	**jtesto4**
(package private) javax.swing.JButton	**pulsanteDue**
(package private) javax.swing.JButton	**pulsanteTre**
(package private) javax.swing.JButton	**pulsanteUno**

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component

BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

CreaDialog(java.awt.Frame parent, java.lang.String titolo, boolean tipo, java.lang.String intervallo, java.lang.String NomePulsante)
CreaDialog(java.awt.Frame parent, java.lang.String titolo, boolean tipo, java.lang.String s1, java.lang.String s2, java.lang.String pulsanteOk, java.lang.String pulsanteAnnulla)
CreaDialog(java.awt.Frame parent, java.lang.String titolo, boolean tipo, java.lang.String s1, java.lang.String s2, java.lang.String s3, java.lang.String s4, java.lang.String pulsanteOk, java.lang.String pulsanteAnnulla)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e)
static java.lang.String	**getString**()
static java.lang.String	**getString2**()
static java.lang.String	**getString3**()
static java.lang.String	**getString4**()
void	**windowActivated**(java.awt.event.WindowEvent event)
void	**windowClosed**(java.awt.event.WindowEvent event)
void	**windowClosing**(java.awt.event.WindowEvent event)
void	**windowDeactivated**(java.awt.event.WindowEvent event)
void	**windowDeiconified**(java.awt.event.WindowEvent event)

void	**windowIconified**(java.awt.event.WindowEvent event)
void	**windowOpened**(java.awt.event.WindowEvent event)

Methods inherited from class java.awt.Dialog
addNotify, dispose, getAccessibleContext, getTitle, hide, isModal, isResizable, isUndecorated, paramString, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage,

coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

jtesto1

javax.swing.JTextField **jtesto1**

---

jtesto2

javax.swing.JTextField **jtesto2**

---

jtesto3

javax.swing.JTextField **jtesto3**

jtesto4

```
javax.swing.JTextField jtesto4
```

pulsanteUno

```
javax.swing.JButton pulsanteUno
```

pulsanteDue

```
javax.swing.JButton pulsanteDue
```

pulsanteTre

```
javax.swing.JButton pulsanteTre
```

cancel

```
javax.swing.JButton cancel
```

input

```
static java.lang.String input
```

input2

```
static java.lang.String input2
```

input3

```
static java.lang.String input3
```

input4

```
static java.lang.String input4
```

Constructor Detail

CreaDialog

```
public CreaDialog(java.awt.Frame parent,
                  java.lang.String titolo,
```

```
                  boolean tipo,
                  java.lang.String intervallo,
                  java.lang.String NomePulsante)
```

CreaDialog

```
public CreaDialog(java.awt.Frame parent,
                  java.lang.String titolo,
                  boolean tipo,
                  java.lang.String s1,
                  java.lang.String s2,
                  java.lang.String pulsanteOk,
                  java.lang.String pulsanteAnnulla)
```

CreaDialog

```
public CreaDialog(java.awt.Frame parent,
                  java.lang.String titolo,
                  boolean tipo,
                  java.lang.String s1,
                  java.lang.String s2,
                  java.lang.String s3,
                  java.lang.String s4,
                  java.lang.String pulsanteOk,
                  java.lang.String pulsanteAnnulla)
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

---

getString

```
public static java.lang.String getString()
```

---

getString2

```
public static java.lang.String getString2()
```

---

getString3

```
public static java.lang.String getString3()
```

---

getString4

```
public static java.lang.String getString4()
```

---

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent event)
```

Specified by:
`windowClosing` in interface `java.awt.event.WindowListener`

---

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent event)
```

Specified by:
`windowIconified` in interface `java.awt.event.WindowListener`

---

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent event)
```

Specified by:
`windowDeiconified` in interface `java.awt.event.WindowListener`

---

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent event)
```

Specified by:
`windowClosed` in interface `java.awt.event.WindowListener`

---

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent event)
```

Specified by:
`windowDeactivated` in interface `java.awt.event.WindowListener`

---

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent event)
```

Specified by:
`windowActivated` in interface `java.awt.event.WindowListener`

---

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent event)
```

Specified by:
`windowOpened` in interface `java.awt.event.WindowListener`

Class DTDGenerator

```
java.lang.Object
  |
  +--org.xml.sax.helpers.DefaultHandler
        |
        +--DTDGenerator
```

All Implemented Interfaces:

org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler

---

public class **DTDGenerator**
extends org.xml.sax.helpers.DefaultHandler

Nested Class Summary

private class	**DTDGenerator.AttributeDetails** AttributeDetails is a data structure to keep information about attribute types
private class	**DTDGenerator.ChildDetails** ChildDetails records information about the presence of a child element within its parent element.
private class	**DTDGenerator.ElementDetails** ElementDetails is a data structure to keep information about element types
private class	**DTDGenerator.StackEntry** StackEntry is a data structure we put on the stack for each nested element

Field Summary

(package private) java.util.TreeMap	**elementList**
(package private) java.util.Stack	**elementStack**
protected static int	**MAX_ENUMERATION_VALUES**
protected static int	**MAX_ID_VALUES**
protected static int	**MIN_ENUMERATION_INSTANCES**
protected static int	**MIN_ENUMERATION_RATIO**
protected static int	**MIN_FIXED**
protected static int	**MIN_ID_VALUES**

Constructor Summary

DTDGenerator()	

Method Summary

void	**characters**(char[] ch, int start, int length) Handle character data.
void	**endElement**(java.lang.String uri, java.lang.String localName, java.lang.String name) End of element.
private static int	**escape**(char[] ch, int start, int length, char[] out) Escape special characters for display.
private static java.lang.String	**escape**(java.lang.String in) Escape special characters in a String value.
private boolean	**isValidName**(java.lang.String s) Test whether a string is an XML name.
private boolean	**isValidNMTOKEN**(java.lang.String s) Test whether a string is an XML NMTOKEN.
void	**printDTD**(java.lang.String nomeSalvaDTD) When the whole document has been analysed, construct the DTD
void	**run**(java.lang.String filename)
void	**startElement**(java.lang.String uri, java.lang.String localName, java.lang.String name, org.xml.sax.Attributes attributes) Handle the start of an element.

Methods inherited from class org.xml.sax.helpers.DefaultHandler

endDocument, endPrefixMapping, error, fatalError, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setDocumentLocator, skippedEntity, startDocument, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

MIN_ENUMERATION_INSTANCES

```
protected static int MIN_ENUMERATION_INSTANCES
```

MAX_ENUMERATION_VALUES

```
protected static int MAX_ENUMERATION_VALUES
```

MIN_ENUMERATION_RATIO

```
protected static int MIN_ENUMERATION_RATIO
```

MIN_FIXED

```
protected static int MIN_FIXED
```

MIN_ID_VALUES

```
protected static int MIN_ID_VALUES
```

MAX_ID_VALUES

```
protected static int MAX_ID_VALUES
```

---

elementList

```
java.util.TreeMap elementList
```

---

elementStack

```
java.util.Stack elementStack
```

Constructor Detail

DTDGenerator

```
public DTDGenerator()
```

Method Detail

run

```
public void run(java.lang.String filename)
```

---

isValidName

```
private boolean isValidName(java.lang.String s)
```

Test whether a string is an XML name. TODO: This is currently an incomplete test, it treats all non-ASCII characters as being valid in names.

---

isValidNMTOKEN

```
private boolean isValidNMTOKEN(java.lang.String s)
```

Test whether a string is an XML NMTOKEN. TODO: This is currently an incomplete test, it treats all non-ASCII characters as being valid in NMTOKENs.

---

printDTD

```
public void printDTD(java.lang.String nomeSalvaDTD)
```

When the whole document has been analysed, construct the DTD

---

escape

```
private static int escape(char[] ch,
                         int start,
                         int length,
                         char[] out)
```

Escape special characters for display.

Parameters:

`ch` - The character array containing the string

`start` - The start position of the input string within the character array

`length` - The length of the input string within the character array

Returns:

The XML/HTML representation of the string

This static method converts a Unicode string to a string containing only ASCII characters, in which non-ASCII characters are represented by the usual XML/HTML escape conventions (for example, "<" becomes "<"). Note: if the input consists solely of ASCII or Latin-1 characters, the output will be equally valid in XML and HTML. Otherwise it will be valid only in XML. The escaped characters are written to the dest array starting at position 0; the number of positions used is returned as the result

---

escape

```
private static java.lang.String escape(java.lang.String in)
```

Escape special characters in a String value.

Parameters:

`in` - The input string

Returns:

The XML representation of the string
This static method converts a Unicode string to a string containing only ASCII characters, in which non-ASCII characters are represented by the usual XML/HTML escape conventions (for example, "<" becomes "<").
Note: if the input consists solely of ASCII or Latin-1 characters, the output will be equally valid in XML and HTML. Otherwise it will be valid only in XML.

---

startElement

```
public void startElement(java.lang.String uri,
                         java.lang.String localName,
                         java.lang.String name,
                         org.xml.sax.Attributes attributes)
                  throws org.xml.sax.SAXException
```

Handle the start of an element. Record information about the position of this element relative to its parent, and about the attributes of the element.

Specified by:

`startElement` in interface `org.xml.sax.ContentHandler`

Overrides:

`startElement` in class `org.xml.sax.helpers.DefaultHandler`
`org.xml.sax.SAXException`

---

endElement

```
public void endElement(java.lang.String uri,
                       java.lang.String localName,
                       java.lang.String name)
                throws org.xml.sax.SAXException
```

End of element. If sequenced, check that all expected children are accounted for.

Specified by:

`endElement` in interface `org.xml.sax.ContentHandler`

Overrides:

`endElement` in class `org.xml.sax.helpers.DefaultHandler`
`org.xml.sax.SAXException`

---

characters

```
public void characters(char[] ch,
                       int start,
                       int length)
                throws org.xml.sax.SAXException
```

Handle character data. Make a note whether significant character data is found in the element

Specified by:

`characters` in interface `org.xml.sax.ContentHandler`

Overrides:

`characters` in class `org.xml.sax.helpers.DefaultHandler`
`org.xml.sax.SAXException`

Class FiltroFile

```
java.lang.Object
  |
  +--javax.swing.filechooser.FileFilter
        |
        +--FiltroFile
```

---

class **FiltroFile**
extends javax.swing.filechooser.FileFilter

Field Summary

private java.lang.String	**filtro_descrizione**
private java.lang.String	**filtro_estensione**

Constructor Summary

FiltroFile(java.lang.String estensione, java.lang.String descrizione)

Method Summary

boolean	**accept**(java.io.File f)
java.lang.String	**getDescription**()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

filtro_estensione

```
private java.lang.String filtro_estensione
```

filtro_descrizione

```
private java.lang.String filtro_descrizione
```

Constructor Detail

FiltroFile

```
public FiltroFile(java.lang.String estensione,
                  java.lang.String descrizione)
```

Method Detail

getDescription

```
public java.lang.String getDescription()
```

Specified by:

`getDescription` in class `javax.swing.filechooser.FileFilter`

accept

```
public boolean accept(java.io.File f)
```

Specified by:

`accept` in class `javax.swing.filechooser.FileFilter`

Class Guida

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Dialog
                          |
                          +--javax.swing.JDialog
                                |
                                +--Guida
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

class **Guida**
extends javax.swing.JDialog
implements java.awt.event.ActionListener

Nested Class Summary

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

private javax.swing.JButton	**pulsanteOk**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

Guida(javax.swing.JFrame parent)

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent evento)

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

pulsanteOk

```
private javax.swing.JButton pulsanteOk
```

Constructor Detail

Guida

```
public Guida(javax.swing.JFrame parent)
```

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent evento)
```

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

Class ImpostaFont

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Dialog
                          |
                          +--javax.swing.JDialog
                                |
                                +--ImpostaFont
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, javax.swing.event.ListSelectionListener, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

class **ImpostaFont**
extends javax.swing.JDialog
implements java.awt.event.ActionListener, javax.swing.event.ListSelectionListener

---

Nested Class Summary

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.GraphicsEnvironment	**ambiente**
private java.awt.Font	**fontScelto**
private java.lang.String[]	**fontSize**
private java.lang.String[]	**fontStyle**
private java.lang.String[]	**fontType**
private javax.swing.JList	**listaDimensioniFont**
private javax.swing.JList	**listaFont**
private javax.swing.JList	**listaStiliFont**
private java.lang.String	**prev**
private javax.swing.JLabel	**preview**
private javax.swing.JButton	**pulsanteCancella**
private javax.swing.JButton	**pulsanteOk**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ImpostaFont(javax.swing.JFrame parent)	

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent evento)
java.awt.Font	**returnFont**()
void	**valueChanged**(javax.swing.event.ListSelectionEvent evento)

Methods inherited from class javax.swing.JDialog

addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage,

createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

ambiente

```
java.awt.GraphicsEnvironment ambiente
```

---

fontType

```
private java.lang.String[] fontType
```

---

fontStyle

```
private java.lang.String[] fontStyle
```

---

fontSize

```
private java.lang.String[] fontSize
```

---

prev

```
private java.lang.String prev
```

---

preview

```
private javax.swing.JLabel preview
```

---

fontScelto

```
private java.awt.Font fontScelto
```

---

pulsanteOk

```
private javax.swing.JButton pulsanteOk
```

---

pulsanteCancella

```
private javax.swing.JButton pulsanteCancella
```

---

listaFont

```
private javax.swing.JList listaFont
```

---

listaStiliFont

```
private javax.swing.JList listaStiliFont
```

---

listaDimensioniFont

```
private javax.swing.JList listaDimensioniFont
```

Constructor Detail

ImpostaFont

```
public ImpostaFont(javax.swing.JFrame parent)
```

Method Detail

returnFont

```
public java.awt.Font returnFont()
```

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent evento)
```

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

valueChanged

```
public void valueChanged(javax.swing.event.ListSelectionEvent evento)
```

Specified by:
`valueChanged` in interface `javax.swing.event.ListSelectionListener`

Class JOptionPaneEsteso

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Dialog
                          |
                          +--javax.swing.JDialog
                                |
                                +--JOptionPaneEsteso
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **JOptionPaneEsteso**
extends javax.swing.JDialog

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

JOptionPaneEsteso(javax.swing.JFrame parent, java.lang.String titolo, java.lang.String testoMessaggio)

Methods inherited from class javax.swing.JDialog

addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog

addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print,

printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

JOptionPaneEsteso

```
public JOptionPaneEsteso(javax.swing.JFrame parent,
```

```
java.lang.String titolo,
java.lang.String testoMessaggio)
```

Class PopupListener

```
java.lang.Object
  |
  +--java.awt.event.MouseAdapter
        |
        +--PopupListener
```

All Implemented Interfaces:

java.util.EventListener, java.awt.event.MouseListener

---

class **PopupListener**
extends java.awt.event.MouseAdapter

Field Summary

(package private) javax.swing.JPopupMenu	**oggettoMenuJPopup**

Constructor Summary

PopupListener(javax.swing.JPopupMenu parametroMenu)

Method Summary

private void	**mostraMenuPopup**(java.awt.event.MouseEvent e)
void	**mousePressed**(java.awt.event.MouseEvent e)
void	**mouseReleased**(java.awt.event.MouseEvent e)

Methods inherited from class java.awt.event.MouseAdapter

mouseClicked, mouseEntered, mouseExited

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

oggettoMenuJPopup

```
javax.swing.JPopupMenu oggettoMenuJPopup
```

Constructor Detail

PopupListener

```
public PopupListener(javax.swing.JPopupMenu parametroMenu)
```

Method Detail

mousePressed

```
public void mousePressed(java.awt.event.MouseEvent e)
```

Specified by:
`mousePressed` in interface `java.awt.event.MouseListener`
Overrides:
`mousePressed` in class `java.awt.event.MouseAdapter`

---

mouseReleased

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Specified by:
`mouseReleased` in interface `java.awt.event.MouseListener`
Overrides:
`mouseReleased` in class `java.awt.event.MouseAdapter`

---

mostraMenuPopup

```
private void mostraMenuPopup(java.awt.event.MouseEvent e)
```

Class RenderAlbero

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JLabel
                          |
                          +--javax.swing.tree.DefaultTreeCellRenderer
                                |
                                +--RenderAlbero
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.swing.SwingConstants, javax.swing.tree.TreeCellRenderer

---

class **RenderAlbero**
extends javax.swing.tree.DefaultTreeCellRenderer

Nested Class Summary

Nested classes inherited from class javax.swing.JLabel
javax.swing.JLabel.AccessibleJLabel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.tree.DefaultTreeCellRenderer
backgroundNonSelectionColor, backgroundSelectionColor, borderSelectionColor, closedIcon, hasFocus, leafIcon, openIcon, selected, textNonSelectionColor, textSelectionColor

Fields inherited from class javax.swing.JLabel
labelFor

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.SwingConstants
BOTTOM, CENTER, EAST, HORIZONTAL, LEADING, LEFT, NEXT, NORTH, NORTH_EAST, NORTH_WEST, PREVIOUS, RIGHT, SOUTH, SOUTH_EAST, SOUTH_WEST, TOP, TRAILING, VERTICAL, WEST

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

RenderAlbero()

Method Summary

java.awt.Component	**getTreeCellRendererComponent**(javax.swing.JTree albero, java.lang.Object valore, boolean sel, boolean espanso, boolean foglia, int riga, boolean haFocus)

Methods inherited from class javax.swing.tree.DefaultTreeCellRenderer

firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackgroundNonSelectionColor, getBackgroundSelectionColor, getBorderSelectionColor, getClosedIcon, getDefaultClosedIcon, getDefaultLeafIcon, getDefaultOpenIcon, getFont, getLeafIcon, getOpenIcon, getPreferredSize, getTextNonSelectionColor, getTextSelectionColor, paint, repaint, repaint, revalidate, setBackground, setBackgroundNonSelectionColor, setBackgroundSelectionColor, setBorderSelectionColor, setClosedIcon, setFont, setLeafIcon, setOpenIcon, setTextNonSelectionColor, setTextSelectionColor, validate

Methods inherited from class javax.swing.JLabel

checkHorizontalKey, checkVerticalKey, getAccessibleContext, getDisabledIcon, getDisplayedMnemonic, getDisplayedMnemonicIndex, getHorizontalAlignment, getHorizontalTextPosition, getIcon, getIconTextGap, getLabelFor, getText, getUI, getUIClassID, getVerticalAlignment, getVerticalTextPosition, imageUpdate, paramString, setDisabledIcon, setDisplayedMnemonic, setDisplayedMnemonic, setDisplayedMnemonicIndex, setHorizontalAlignment, setHorizontalTextPosition, setIcon, setIconTextGap, setLabelFor, setText, setUI, setVerticalAlignment, setVerticalTextPosition, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes,

getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent,

processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RenderAlbero

```
public RenderAlbero()
```

Method Detail

getTreeCellRendererComponent

```
public java.awt.Component getTreeCellRendererComponent(javax.swing.JTree albero,
                                                       java.lang.Object valore,
                                                       boolean sel,
                                                       boolean espanso,
                                                       boolean foglia,
                                                       int riga,
                                                       boolean haFocus)
```

Specified by:
`getTreeCellRendererComponent` in interface `javax.swing.tree.TreeCellRenderer`
Overrides:
`getTreeCellRendererComponent` in class `javax.swing.tree.DefaultTreeCellRenderer`

Class StampaGrafica

```
java.lang.Object
  |
  +--StampaGrafica
```

---

public class **StampaGrafica**
extends java.lang.Object

Constructor Summary

StampaGrafica(java.awt.PrintJob jobStampa, java.awt.Graphics oggettoGrafico, java.lang.String testo, java.lang.String nomeDocumento)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

StampaGrafica

```
public StampaGrafica(java.awt.PrintJob jobStampa,
                     java.awt.Graphics oggettoGrafico,
                     java.lang.String testo,
                     java.lang.String nomeDocumento)
```

Class StoriaFile

```
java.lang.Object
  |
  +--StoriaFile
```

---

public class **StoriaFile**
extends java.lang.Object

Field Summary

(package private) java.lang.String	**primo**
(package private) java.lang.String	**quarto**
(package private) java.lang.String	**secondo**
(package private) java.lang.String	**terzo**

Constructor Summary

StoriaFile()

Method Summary

void	**salvaStoriaFile**(java.io.File ultimofile)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

primo

java.lang.String **primo**

secondo

```
java.lang.String secondo
```

terzo

```
java.lang.String terzo
```

quarto

```
java.lang.String quarto
```

Constructor Detail

StoriaFile

```
public StoriaFile()
```

Method Detail

salvaStoriaFile

```
public void salvaStoriaFile(java.io.File ultimofile)
```

Class TavolaAttributi

```
java.lang.Object
  |
  +--javax.swing.table.AbstractTableModel
        |
        +--TavolaAttributi
```

All Implemented Interfaces:

java.io.Serializable, javax.swing.table.TableModel

---

class **TavolaAttributi**
extends javax.swing.table.AbstractTableModel

Field Summary

protected org.w3c.dom.NamedNodeMap	**attributiCorrenti**
protected org.w3c.dom.Node	**nodoCorrente**

Fields inherited from class javax.swing.table.AbstractTableModel
listenerList

Constructor Summary

(package private)	**TavolaAttributi**()

Method Summary

void	**assegnaNodo**(org.w3c.dom.Node nodo)
int	**getColumnCount**()
java.lang.String	**getColumnName**(int numeroColonna)
int	**getRowCount**()
java.lang.Object	**getValueAt**(int numeroRiga, int numeroColonna)
boolean	**isCellEditable**(int numeroRiga, int numeroColonna)

org.w3c.dom.Node	**restituisciNodoCorrente**()
void	**setValueAt**(java.lang.Object valoreAttributo, int numeroRiga, int numeroColonna)

Methods inherited from class javax.swing.table.AbstractTableModel
addTableModelListener, findColumn, fireTableCellUpdated, fireTableChanged, fireTableDataChanged, fireTableRowsDeleted, fireTableRowsInserted, fireTableRowsUpdated, fireTableStructureChanged, getColumnClass, getListeners, getTableModelListeners, removeTableModelListener

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

nodoCorrente

protected org.w3c.dom.Node **nodoCorrente**

---

attributiCorrenti

protected org.w3c.dom.NamedNodeMap **attributiCorrenti**

Constructor Detail

TavolaAttributi

TavolaAttributi()

Method Detail

assegnaNodo

public void **assegnaNodo**(org.w3c.dom.Node nodo)

---

restituisciNodoCorrente

```
public org.w3c.dom.Node restituisciNodoCorrente()
```

---

getRowCount

```
public int getRowCount()
```

---

getColumnCount

```
public int getColumnCount()
```

---

getColumnName

```
public java.lang.String getColumnName(int numeroColonna)
```

Specified by:
`getColumnName` in interface `javax.swing.table.TableModel`
Overrides:
`getColumnName` in class `javax.swing.table.AbstractTableModel`

---

getValueAt

```
public java.lang.Object getValueAt(int numeroRiga,
                                   int numeroColonna)
```

---

isCellEditable

```
public boolean isCellEditable(int numeroRiga,
                              int numeroColonna)
```

Specified by:
`isCellEditable` in interface `javax.swing.table.TableModel`
Overrides:
`isCellEditable` in class `javax.swing.table.AbstractTableModel`

---

setValueAt

```
public void setValueAt(java.lang.Object valoreAttributo,
                       int numeroRiga,
                       int numeroColonna)
```

Specified by:
`setValueAt` in interface `javax.swing.table.TableModel`
Overrides:
`setValueAt` in class `javax.swing.table.AbstractTableModel`

Class UndoableTextArea

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.text.JTextComponent
                          |
                          +--javax.swing.JTextArea
                                |
                                +--UndoableTextArea
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.util.EventListener, java.awt.event.FocusListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.Scrollable, java.io.Serializable, javax.swing.event.UndoableEditListener

---

class **UndoableTextArea**
extends javax.swing.JTextArea
implements javax.swing.event.UndoableEditListener, java.awt.event.FocusListener

---

Nested Class Summary

Nested classes inherited from class javax.swing.JTextArea
`javax.swing.JTextArea.AccessibleJTextArea`

Nested classes inherited from class javax.swing.text.JTextComponent
`javax.swing.text.JTextComponent.AccessibleJTextComponent`, `javax.swing.text.JTextComponent.KeyBinding`

Nested classes inherited from class javax.swing.JComponent
`javax.swing.JComponent.AccessibleJComponent`

Nested classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

java.awt.Color	**coloreTestoEvidenziato**
static int	**limiteundo**
javax.swing.undo.UndoManager	**oggettoUndoManager**

Fields inherited from class javax.swing.JTextArea

Fields inherited from class javax.swing.text.JTextComponent
DEFAULT_KEYMAP, FOCUS_ACCELERATOR_KEY

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

UndoableTextArea()

Method Summary

void	**annulla**()
void	**cancelladatiUndoManager**()
void	**creaUndoManager**()
void	**focusGained**(java.awt.event.FocusEvent fe)
void	**focusLost**(java.awt.event.FocusEvent fe)
void	**rimuoviUndoManager**()
void	**ripristina**()
void	**undoableEditHappened**(javax.swing.event.UndoableEditEvent e)

Methods inherited from class javax.swing.JTextArea

append, createDefaultModel, getAccessibleContext, getColumns, getColumnWidth, getLineCount, getLineEndOffset, getLineOfOffset, getLineStartOffset, getLineWrap, getPreferredScrollableViewportSize, getPreferredSize, getRowHeight, getRows, getScrollableTracksViewportWidth, getScrollableUnitIncrement, getTabSize, getUIClassID, getWrapStyleWord, insert, paramString, replaceRange, setColumns, setFont, setLineWrap, setRows, setTabSize, setWrapStyleWord

Methods inherited from class javax.swing.text.JTextComponent

addCaretListener, addInputMethodListener, addKeymap, copy, cut, fireCaretUpdate, getActions, getCaret, getCaretColor, getCaretListeners, getCaretPosition, getDisabledTextColor, getDocument, getDragEnabled, getFocusAccelerator, getHighlighter, getInputMethodRequests, getKeymap, getKeymap, getMargin, getNavigationFilter, getScrollableBlockIncrement, getScrollableTracksViewportHeight, getSelectedText, getSelectedTextColor,

getSelectionColor, getSelectionEnd, getSelectionStart, getText, getText, getToolTipText, getUI, isEditable, loadKeymap, modelToView, moveCaretPosition, paste, processInputMethodEvent, read, removeCaretListener, removeKeymap, removeNotify, replaceSelection, select, selectAll, setCaret, setCaretColor, setCaretPosition, setComponentOrientation, setDisabledTextColor, setDocument, setDragEnabled, setEditable, setFocusAccelerator, setHighlighter, setKeymap, setMargin, setNavigationFilter, setSelectedTextColor, setSelectionColor, setSelectionEnd, setSelectionStart, setText, setUI, updateUI, viewToModel, write

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout,

transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

limiteundo

```
public static final int limiteundo
```

See Also:

Constant Field Values

---

oggettoUndoManager

```
public javax.swing.undo.UndoManager oggettoUndoManager
```

coloreTestoEvidenziato

```
public java.awt.Color coloreTestoEvidenziato
```

Constructor Detail

UndoableTextArea

```
public UndoableTextArea()
```

Method Detail

creaUndoManager

```
public void creaUndoManager()
```

rimuoviUndoManager

```
public void rimuoviUndoManager()
```

cancelladatiUndoManager

```
public void cancelladatiUndoManager()
```

undoableEditHappened

```
public void undoableEditHappened(javax.swing.event.UndoableEditEvent e)
```

Specified by:
undoableEditHappened in interface javax.swing.event.UndoableEditListener

annulla

```
public void annulla()
```

ripristina

```
public void ripristina()
```

focusGained

```
public void focusGained(java.awt.event.FocusEvent fe)
```

Specified by:
focusGained in interface java.awt.event.FocusListener

focusLost

```
public void focusLost(java.awt.event.FocusEvent fe)
```

Specified by:

`focusLost` in interface `java.awt.event.FocusListener`

Class XmlNodo

```
java.lang.Object
  |
  +--javax.swing.tree.DefaultMutableTreeNode
        |
        +--XmlNodo
```

All Implemented Interfaces:

java.lang.Cloneable, javax.swing.tree.MutableTreeNode, java.io.Serializable, javax.swing.tree.TreeNode

---

class **XmlNodo**
extends javax.swing.tree.DefaultMutableTreeNode

Nested Class Summary

Nested classes inherited from class javax.swing.tree.DefaultMutableTreeNode

Field Summary

Fields inherited from class javax.swing.tree.DefaultMutableTreeNode
allowsChildren, children, EMPTY_ENUMERATION, parent, userObject

Constructor Summary

XmlNodo(org.w3c.dom.Node node)	

Method Summary

void	aggiungiXmlNodo(XmlNodo nodoFiglio)
org.w3c.dom.Node	restituisciXmlNodo()
void	rimuovi()

java.lang.String	**toString**()

Methods inherited from class javax.swing.tree.DefaultMutableTreeNode
add, breadthFirstEnumeration, children, clone, depthFirstEnumeration, getAllowsChildren, getChildAfter, getChildAt, getChildBefore, getChildCount, getDepth, getFirstChild, getFirstLeaf, getIndex, getLastChild, getLastLeaf, getLeafCount, getLevel, getNextLeaf, getNextNode, getNextSibling, getParent, getPath, getPathToRoot, getPreviousLeaf, getPreviousNode, getPreviousSibling, getRoot, getSharedAncestor, getSiblingCount, getUserObject, getUserObjectPath, insert, isLeaf, isNodeAncestor, isNodeChild, isNodeDescendant, isNodeRelated, isNodeSibling, isRoot, pathFromAncestorEnumeration, postorderEnumeration, preorderEnumeration, remove, remove, removeAllChildren, removeFromParent, setAllowsChildren, setParent, setUserObject

Methods inherited from class java.lang.Object
equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

XmlNodo

```
public XmlNodo(org.w3c.dom.Node node)
```

Method Detail

restituisciXmlNodo

```
public org.w3c.dom.Node restituisciXmlNodo()
```

---

toString

```
public java.lang.String toString()
```

Overrides:

toString in class javax.swing.tree.DefaultMutableTreeNode

---

aggiungiXmlNodo

```
public void aggiungiXmlNodo(XmlNodo nodoFiglio)
                    throws java.lang.Exception
```

java.lang.Exception

rimuovi

```
public void rimuovi()
             throws java.lang.Exception
      java.lang.Exception
```

Class XMLRoutine

```
java.lang.Object
  |
  +--XMLRoutine
```

---

class **XMLRoutine**
extends java.lang.Object

Field Summary

(package private) static int	**spaziTab**

Constructor Summary

(package private)	**XMLRoutine**()

Method Summary

static void	**write**(org.w3c.dom.Document documento, java.io.Writer out)
static void	**write**(org.w3c.dom.Node nodo, java.io.Writer out)
static void	**writeDocumento**(org.w3c.dom.Document documento, java.io.Writer out)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

spaziTab

```
static int spaziTab
```

Constructor Detail

XMLRoutine

XMLRoutine()

Method Detail

writeDocumento

```
public static void writeDocumento(org.w3c.dom.Document documento,
                                  java.io.Writer out)
                           throws java.lang.Exception
      java.lang.Exception
```

---

write

```
public static void write(org.w3c.dom.Document documento,
                         java.io.Writer out)
                  throws java.lang.Exception
      java.lang.Exception
```

---

write

```
public static void write(org.w3c.dom.Node nodo,
                         java.io.Writer out)
                  throws java.lang.Exception
      java.lang.Exception
```

Class XMLStudio

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--javax.swing.JFrame
                                |
                                +--XMLStudio
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, javax.swing.event.CaretListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants, java.awt.event.WindowListener

---

public class **XMLStudio**
extends javax.swing.JFrame
implements java.awt.event.ActionListener, java.awt.event.WindowListener, javax.swing.event.CaretListener

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
`javax.swing.JFrame.AccessibleJFrame`

Nested classes inherited from class java.awt.Frame
`java.awt.Frame.AccessibleAWTFrame`

Nested classes inherited from class java.awt.Window
`java.awt.Window.AccessibleAWTWindow`

Nested classes inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) javax.swing.JButton	**aggiornaxml**
(package private) javax.swing.JButton	**aggiungitagxml**
protected javax.swing.JTree	**alberoXml**
(package private) javax.swing.JButton	**annulla**
(package private) javax.swing.JButton	**apri**
UndoableTextArea	**areaTesto**
(package private) javax.swing.JButton	**benformatoxml**
private java.awt.Font	**carattereDefault**
private javax.swing.JLabel	**colonnerighe**
private java.awt.Color	**coloreSfondoDefault**
private java.awt.Color	**coloreTestoDefault**
(package private) javax.swing.JButton	**copia**
(package private) javax.swing.JButton	**creaxml**
private java.lang.String	**directoryCorrente**
private java.lang.String	**divisoreOrizzontale**
private java.lang.String	**divisoreVerticale**
protected org.w3c.dom.Document	**documentoXml**

private java.io.File	**fileCorrente**
private boolean	**fileGiaSalvato**
private java.lang.String	**fileXmlTagCorrente**
private java.lang.String	**finestraX**
private java.lang.String	**finestraY**
(package private) HelpBroker	**hb**
(package private) javax.swing.JButton	**helpcontesto**
(package private) HelpSet	**hs**
(package private) javax.swing.JButton	**incolla**
javax.swing.JMenuItem	**jmi_aggiungiattributonodotag**
javax.swing.JMenuItem	**jmi_aggiunginodotag**
javax.swing.JMenuItem	**jmi_eliminaattributonodotag**
javax.swing.JMenuItem	**jmi_eliminanodotag**
javax.swing.JMenuItem	**jmi_modificaattributonodotag**
javax.swing.JMenuItem	**jmi_modificanodotag**
javax.swing.JMenuItem	**jmi_salva**
(package private) javax.swing.JCheckBoxMenuItem	**jmi_wordwrap**
private java.lang.String	**lookandfeelCorrente**
(package private) javax.swing.JButton	**memolinea**
private java.lang.String	**memorizzaLinea**
protected javax.swing.tree.DefaultTreeModel	**modelloAlbero**
protected TavolaAttributi	**modellotavolaAttributiXml**
(package private)	**nuovo**

javax.swing.JButton	
private XmlTag	**oggettoXmlTag**
private java.lang.String	**parola**
javax.swing.JPopupMenu	**popupAreaTesto**
private int	**riga**
(package private) javax.swing.JButton	**ripristina**
(package private) javax.swing.JButton	**salva**
private java.awt.Dimension	**screenSize**
(package private) javax.swing.JButton	**sostituisci**
private javax.swing.JSplitPane	**splitPaneCentrale**
private javax.swing.JSplitPane	**splitPaneCentrale2**
(package private) javax.swing.JButton	**stampa**
javax.swing.JLabel	**stato**
(package private) javax.swing.JButton	**taglia**
protected javax.swing.JTable	**tavolaAttributiXml**
(package private) javax.swing.JButton	**trova**
(package private) javax.swing.JButton	**trovasuccessivo**
private StoriaFile	**ultimiFile**
(package private) javax.swing.JButton	**vaimemolinea**
private java.lang.Boolean	**wordWrapDefault**

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

private	**XMLStudio**()

Method Summary

void	**rmed**(iava.awt.event.ActionEvent e)

void	**aggiornaAttributiNodoInTavola**(org.w3c.dom.Node
private void	**aggiungiIntestazioneCSS**(java.lang.String nomeFi
private void	**aggiungiIntestazioniXml**()
private void	**azioneAboutBox**()
private void	**azioneAggiungiTagXml**()
private void	**azioneAnnulla**()
private void	**azioneApriFile**()
private void	**azioneApriUltimoFile**(java.lang.String stringaul
private boolean	**azioneBenFormatoXml**()
private void	**azioneChiudiFile**()
private void	**azioneCopia**()
private void	**azioneCreaAlberoXml**()
private void	**azioneCreaAlberoXmlVuoto**()
private void	**azioneCreaCss**()
private void	**azioneCreaDocumentoXmlValido**()
private boolean	**azioneCreaDtd**()
private void	**azioneCreaXml**()
private void	**azioneDimensioneFinestra**()
private void	**azioneGuida**()
private void	**azioneImpostaDirectory**()
private void	**azioneImpostaFileXmlTag**()
private void	**azioneImpostaLookMac**()

private void	**azioneImpostaLookMetal**()
private void	**azioneImpostaLookMotif**()
private void	**azioneImpostaLookSystem**()
private void	**azioneImpostaLookWindows**()
private void	**azioneIncolla**()
private void	**azioneMemorizzaLinea**()
private void	**azioneNuovoFile**()
private void	**azioneRiavviaEditor**()
private void	**azioneRicordaSalvaFile**()
private void	**azioneRimuoviTagXml**()
private void	**azioneRipristina**()
private void	**azioneRipristinaCaratteriSpeciali**()
private void	**azioneSalvaFile**()
private void	**azioneSalvaFileNome**()
private void	**azioneSalvaPulsante**(boolean abilitato)
private void	**azioneSelezionaTagXml**()
private void	**azioneSelezionaTutto**()
private void	**azioneSostituisci**()
private void	**azioneSostituisciApostrofo**()
private void	**azioneSostituisciAutomatico**(java.lang.String pa java.lang.String parolaSostituisci)
private void	**azioneSostituisciE**()
private void	**azioneSostituisciMaggiore**()

private void	**azioneSostituisciManuale**(java.lang.String parol java.lang.String parolaSostituisci)
private void	**azioneSostituisciMinore**()
private void	**azioneSostituisciQuota**()
private void	**azioneStampaGrafica**()
private void	**azioneStampaLinux**()
private void	**azioneStampaUnix**()
private void	**azioneStampaWindows**()
private void	**azioneTaglia**()
private void	**azioneTipoCarattere**()
private void	**azioneTipoColoreSfondo**()
private void	**azioneTipoColoreTesto**()
private void	**azioneTrova**()
private java.lang.String	**azioneTrovaCaratteriNonValidi**()
private boolean	**azioneTrovaStringa**(java.lang.String parola)
private void	**azioneTrovaSuccessivo**()
private void	**azioneTrovaTag**()
private void	**azioneUscita**()
private void	**azioneVaiALinea**()
private void	**azioneVaiAMemorizzaLinea**()
private void	**azioneWordWrap**()
void	**caretUpdate**(javax.swing.event.CaretEvent eventc
private void	**carica_ImpostazioniDefault**()

private javax.swing.JScrollPane	**crea_AreaTesto**()
private javax.swing.JToolBar	**crea_Barra**()
private javax.swing.JMenuBar	**crea_MenuBar**()
private javax.swing.JPanel	**crea_PannelloAlberoXml**()
private javax.swing.JSplitPane	**crea_SplitPaneCentrale**()
private javax.swing.JPanel	**crea_StatusBar**()
private javax.swing.tree.DefaultMutableTreeNode	**creaAlberoNodi**(org.w3c.dom.Node nodoRadice)
private void	**creaJavaHelp**()
private void	**creaMenuPopup**()
void	**cursoreClessidra**()
void	**cursoreNormale**()
private void	**elaboraSalvaCSS**(java.io.File fileCss)
private void	**eseguiImpostazioniDefault**(java.util.Properties
private static void	**espandiAlbero**(javax.swing.JTree alberoJTree)
private static void	**espandiAlbero**(javax.swing.JTree alberoJTree, javax.swing.tree.TreePath path, javax.swing.tree.TreeNode nodo)
java.lang.String	**getfileXmlTagCorrente**()
private void	**impostaProprietaAreaTesto**(javax.swing.JTextArea
private void	**initLookAndFeel**(java.lang.String look)
static void	**main**(java.lang.String[] args)
private boolean	**possoVisualizzareNodo**(org.w3c.dom.Node nodocorr
private java.lang.String	**restituisciNodoperCSS**()
org.w3c.dom.Node	**restituisciNodoSelezionato**()

XmlNodo	**restituisciNodoSelezionatoAlbero**()
private void	**rimuoviCarattere**()
private void	**ripristinaImpostazioniDefault**()
private void	**salva_ImpostazioniDefault**()
private boolean	**selezioneContieneFine**(java.lang.String testo, java.lang.String contiene)
private boolean	**selezioneContieneInizio**(java.lang.String testo, java.lang.String contiene)
private void	**vaiALinea**(java.lang.String linea)
void	**windowActivated**(java.awt.event.WindowEvent ever
void	**windowClosed**(java.awt.event.WindowEvent event)
void	**windowClosing**(java.awt.event.WindowEvent event)
void	**windowDeactivated**(java.awt.event.WindowEvent ev
void	**windowDeiconified**(java.awt.event.WindowEvent ev
void	**windowIconified**(java.awt.event.WindowEvent ever
void	**windowOpened**(java.awt.event.WindowEvent event)

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame
addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage,

setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp,

move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
getFont, postEvent

Field Detail

screenSize

```
private java.awt.Dimension screenSize
```

---

stato

```
public javax.swing.JLabel stato
```

---

colonnerighe

```
private javax.swing.JLabel colonnerighe
```

---

carattereDefault

```
private java.awt.Font carattereDefault
```

---

coloreTestoDefault

```
private java.awt.Color coloreTestoDefault
```

---

coloreSfondoDefault

```
private java.awt.Color coloreSfondoDefault
```

---

wordWrapDefault

```
private java.lang.Boolean wordWrapDefault
```

---

directoryCorrente

```
private java.lang.String directoryCorrente
```

---

fileXmlTagCorrente

```
private java.lang.String fileXmlTagCorrente
```

---

finestraX

```
private java.lang.String finestraX
```

---

finestraY

```
private java.lang.String finestraY
```

---

divisoreOrizzontale

```
private java.lang.String divisoreOrizzontale
```

---

divisoreVerticale

```
private java.lang.String divisoreVerticale
```

---

lookandfeelCorrente

```
private java.lang.String lookandfeelCorrente
```

---

memorizzaLinea

```
private java.lang.String memorizzaLinea
```

---

riga

```
private int riga
```

---

fileCorrente

```
private java.io.File fileCorrente
```

---

fileGiaSalvato

```
private boolean fileGiaSalvato
```

---

parola

```
private java.lang.String parola
```

---

ultimiFile

```
private StoriaFile ultimiFile
```

---

hs

```
HelpSet hs
```

---

hb

```
HelpBroker hb
```

---

areaTesto

```
public UndoableTextArea areaTesto
```

---

popupAreaTesto

```
public javax.swing.JPopupMenu popupAreaTesto
```

---

splitPaneCentrale

```
private javax.swing.JSplitPane splitPaneCentrale
```

---

splitPaneCentrale2

```
private javax.swing.JSplitPane splitPaneCentrale2
```

---

oggettoXmlTag

```
private XmlTag oggettoXmlTag
```

---

documentoXml

```
protected org.w3c.dom.Document documentoXml
```

---

alberoXml

```
protected javax.swing.JTree alberoXml
```

---

modelloAlbero

```
protected javax.swing.tree.DefaultTreeModel modelloAlbero
```

---

tavolaAttributiXml

```
protected javax.swing.JTable tavolaAttributiXml
```

---

modellotavolaAttributiXml

```
protected TavolaAttributi modellotavolaAttributiXml
```

---

nuovo

```
javax.swing.JButton nuovo
```

---

apri

```
javax.swing.JButton apri
```

---

salva

```
javax.swing.JButton salva
```

---

stampa

```
javax.swing.JButton stampa
```

---

taglia

`javax.swing.JButton` **`taglia`**

---

copia

`javax.swing.JButton` **`copia`**

---

incolla

`javax.swing.JButton` **`incolla`**

---

annulla

`javax.swing.JButton` **`annulla`**

---

ripristina

`javax.swing.JButton` **`ripristina`**

---

aggiornaxml

`javax.swing.JButton` **`aggiornaxml`**

---

benformatoxml

`javax.swing.JButton` **`benformatoxml`**

---

aggiungitagxml

`javax.swing.JButton` **`aggiungitagxml`**

---

memolinea

`javax.swing.JButton` **`memolinea`**

---

vaimemolinea

`javax.swing.JButton` **`vaimemolinea`**

---

creaxml

javax.swing.JButton **creaxml**

---

trova

javax.swing.JButton **trova**

---

trovasuccessivo

javax.swing.JButton **trovasuccessivo**

---

sostituisci

javax.swing.JButton **sostituisci**

---

helpcontesto

javax.swing.JButton **helpcontesto**

---

jmi_salva

public javax.swing.JMenuItem **jmi_salva**

---

jmi_aggiunginodotag

public javax.swing.JMenuItem **jmi_aggiunginodotag**

---

jmi_modificanodotag

public javax.swing.JMenuItem **jmi_modificanodotag**

---

jmi_eliminanodotag

public javax.swing.JMenuItem **jmi_eliminanodotag**

---

jmi_aggiungiattributonodotag

public javax.swing.JMenuItem **jmi_aggiungiattributonodotag**

---

jmi_modificaattributonodotag

public javax.swing.JMenuItem **jmi_modificaattributonodotag**

---

jmi_eliminaattributonodotag

```
public javax.swing.JMenuItem jmi_eliminaattributonodotag
```

---

jmi_wordwrap

```
javax.swing.JCheckBoxMenuItem jmi_wordwrap
```

Constructor Detail

XMLStudio

```
private XMLStudio()
```

Method Detail

initLookAndFeel

```
private void initLookAndFeel(java.lang.String look)
```

---

creaJavaHelp

```
private void creaJavaHelp()
```

---

eseguiImpostazioniDefault

```
private void eseguiImpostazioniDefault(java.util.Properties p)
```

---

carica_ImpostazioniDefault

```
private void carica_ImpostazioniDefault()
```

---

salva_ImpostazioniDefault

```
private void salva_ImpostazioniDefault()
```

---

ripristinaImpostazioniDefault

```
private void ripristinaImpostazioniDefault()
```

---

crea_MenuBar

```
private javax.swing.JMenuBar crea_MenuBar()
```

---

crea_Barra

```
private javax.swing.JToolBar crea_Barra()
```

---

crea_StatusBar

```
private javax.swing.JPanel crea_StatusBar()
```

---

crea_PannelloAlberoXml

```
private javax.swing.JPanel crea_PannelloAlberoXml()
```

---

restituisciNodoSelezionatoAlbero

```
public XmlNodo restituisciNodoSelezionatoAlbero()
```

---

restituisciNodoSelezionato

```
public org.w3c.dom.Node restituisciNodoSelezionato()
```

---

aggiornaAttributiNodoInTavola

```
public void aggiornaAttributiNodoInTavola(org.w3c.dom.Node nodo)
```

---

creaMenuPopup

```
private void creaMenuPopup()
```

---

impostaProprietaAreaTesto

```
private void impostaProprietaAreaTesto(javax.swing.JTextArea areaTesto)
```

---

crea_AreaTesto

```
private javax.swing.JScrollPane crea_AreaTesto()
```

---

rimuoviCarattere

```
private void rimuoviCarattere()
```

---

crea_SplitPaneCentrale

```
private javax.swing.JSplitPane crea_SplitPaneCentrale()
```

---

windowClosing

```
public void windowClosing(java.awt.event.WindowEvent event)
```

Specified by:
`windowClosing` in interface `java.awt.event.WindowListener`

---

windowIconified

```
public void windowIconified(java.awt.event.WindowEvent event)
```

Specified by:
`windowIconified` in interface `java.awt.event.WindowListener`

---

windowDeiconified

```
public void windowDeiconified(java.awt.event.WindowEvent event)
```

Specified by:
`windowDeiconified` in interface `java.awt.event.WindowListener`

---

windowClosed

```
public void windowClosed(java.awt.event.WindowEvent event)
```

Specified by:
`windowClosed` in interface `java.awt.event.WindowListener`

---

windowDeactivated

```
public void windowDeactivated(java.awt.event.WindowEvent event)
```

Specified by:
`windowDeactivated` in interface `java.awt.event.WindowListener`

---

windowActivated

```
public void windowActivated(java.awt.event.WindowEvent event)
```

Specified by:
`windowActivated` in interface `java.awt.event.WindowListener`

---

windowOpened

```
public void windowOpened(java.awt.event.WindowEvent event)
```

Specified by:
`windowOpened` in interface `java.awt.event.WindowListener`

---

caretUpdate

```
public void caretUpdate(javax.swing.event.CaretEvent evento)
```

Specified by:
`caretUpdate` in interface `javax.swing.event.CaretListener`

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

cursoreClessidra

```
public void cursoreClessidra()
```

---

cursoreNormale

```
public void cursoreNormale()
```

---

azioneNuovoFile

```
private void azioneNuovoFile()
```

---

azioneChiudiFile

```
private void azioneChiudiFile()
```

---

azioneApriFile

```
private void azioneApriFile()
```

---

azioneApriUltimoFile

```
private void azioneApriUltimoFile(java.lang.String stringaultimofile)
```

---

azioneSalvaFileNome

```
private void azioneSalvaFileNome()
```

---

azioneSalvaPulsante

```
private void azioneSalvaPulsante(boolean abilitato)
```

---

azioneSalvaFile

```
private void azioneSalvaFile()
```

---

azioneRicordaSalvaFile

```
private void azioneRicordaSalvaFile()
```

---

azioneStampaWindows

```
private void azioneStampaWindows()
```

---

azioneStampaLinux

```
private void azioneStampaLinux()
```

---

azioneStampaUnix

```
private void azioneStampaUnix()
```

---

azioneStampaGrafica

```
private void azioneStampaGrafica()
```

---

azioneUscita

```
private void azioneUscita()
```

---

azioneAnnulla

```
private void azioneAnnulla()
```

---

azioneRipristina

```
private void azioneRipristina()
```

---

azioneTaglia

```
private void azioneTaglia()
```

---

azioneCopia

```
private void azioneCopia()
```

---

azioneIncolla

```
private void azioneIncolla()
```

---

azioneSelezionaTutto

```
private void azioneSelezionaTutto()
```

---

azioneTrovaStringa

```
private boolean azioneTrovaStringa(java.lang.String parola)
```

---

azioneTrovaCaratteriNonValidi

```
private java.lang.String azioneTrovaCaratteriNonValidi()
```

---

azioneTrova

```
private void azioneTrova()
```

---

azioneTrovaTag

```
private void azioneTrovaTag()
```

---

azioneTrovaSuccessivo

```
private void azioneTrovaSuccessivo()
```

---

azioneSostituisciE

```
private void azioneSostituisciE()
```

---

azioneSostituisciMinore

```
private void azioneSostituisciMinore()
```

---

azioneSostituisciMaggiore

```
private void azioneSostituisciMaggiore()
```

---

azioneSostituisciApostrofo

```
private void azioneSostituisciApostrofo()
```

---

azioneSostituisciQuota

```
private void azioneSostituisciQuota()
```

---

azioneRipristinaCaratteriSpeciali

```
private void azioneRipristinaCaratteriSpeciali()
```

---

azioneSostituisciAutomatico

```
private void azioneSostituisciAutomatico(java.lang.String parola,
                                         java.lang.String parolaSostituisci)
```

---

azioneSostituisciManuale

```
private void azioneSostituisciManuale(java.lang.String parola,
                                      java.lang.String parolaSostituisci)
```

---

azioneSostituisci

```
private void azioneSostituisci()
```

---

vaiALinea

```
private void vaiALinea(java.lang.String linea)
```

---

azioneVaiALinea

```
private void azioneVaiALinea()
```

---

azioneMemorizzaLinea

```
private void azioneMemorizzaLinea()
```

---

azioneVaiAMemorizzaLinea

```
private void azioneVaiAMemorizzaLinea()
```

---

aggiungiIntestazioniXml

```
private void aggiungiIntestazioniXml()
```

---

azioneCreaXml

```
private void azioneCreaXml()
```

---

azioneImpostaDirectory

```
private void azioneImpostaDirectory()
```

---

azioneImpostaFileXmlTag

```
private void azioneImpostaFileXmlTag()
```

---

azioneDimensioneFinestra

```
private void azioneDimensioneFinestra()
```

---

azioneTipoColoreSfondo

```
private void azioneTipoColoreSfondo()
```

---

azioneTipoColoreTesto

```
private void azioneTipoColoreTesto()
```

---

azioneTipoCarattere

```
private void azioneTipoCarattere()
```

---

azioneWordWrap

```
private void azioneWordWrap()
```

---

azioneImpostaLookMetal

```
private void azioneImpostaLookMetal()
```

---

azioneImpostaLookWindows

```
private void azioneImpostaLookWindows()
```

---

azioneImpostaLookSystem

```
private void azioneImpostaLookSystem()
```

---

azioneImpostaLookMotif

```
private void azioneImpostaLookMotif()
```

---

azioneImpostaLookMac

```
private void azioneImpostaLookMac()
```

---

azioneGuida

```
private void azioneGuida()
```

---

azioneAboutBox

```
private void azioneAboutBox()
```

---

azioneRiavviaEditor

```
private void azioneRiavviaEditor()
```

---

azioneBenFormatoXml

```
private boolean azioneBenFormatoXml()
```

---

azioneCreaAlberoXmlVuoto

```
private void azioneCreaAlberoXmlVuoto()
```

---

azioneCreaAlberoXml

```
private void azioneCreaAlberoXml()
```

---

creaAlberoNodi

```
private javax.swing.tree.DefaultMutableTreeNode
creaAlberoNodi(org.w3c.dom.Node nodoRadice)
```

---

espandiAlbero

```
private static void espandiAlbero(javax.swing.JTree alberoJTree)
```

---

espandiAlbero

```
private static void espandiAlbero(javax.swing.JTree alberoJTree,
                                  javax.swing.tree.TreePath path,
                                  javax.swing.tree.TreeNode nodo)
```

---

possoVisualizzareNodo

```
private boolean possoVisualizzareNodo(org.w3c.dom.Node nodocorrente)
```

---

getfileXmlTagCorrente

```
public java.lang.String getfileXmlTagCorrente()
```

---

azioneAggiungiTagXml

```
private void azioneAggiungiTagXml()
```

---

azioneRimuoviTagXml

```
private void azioneRimuoviTagXml()
```

---

selezioneContieneFine

```
private boolean selezioneContieneFine(java.lang.String testo,
                                      java.lang.String contiene)
```

---

selezioneContieneInizio

```
private boolean selezioneContieneInizio(java.lang.String testo,
                                        java.lang.String contiene)
```

---

azioneSelezionaTagXml

```
private void azioneSelezionaTagXml()
```

---

azioneCreaDtd

```
private boolean azioneCreaDtd()
```

---

azioneCreaDocumentoXmlValido

```
private void azioneCreaDocumentoXmlValido()
```

---

restituisciNodoperCSS

```
private java.lang.String restituisciNodoperCSS()
```

---

elaboraSalvaCSS

```
private void elaboraSalvaCSS(java.io.File fileCss)
```

---

azioneCreaCss

```
private void azioneCreaCss()
```

---

aggiungiIntestazioneCSS

```
private void aggiungiIntestazioneCSS(java.lang.String nomeFileCSS)
```

---

main

```
public static void main(java.lang.String[] args)
```

Class XmlTag

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--XmlTag
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

class **XmlTag**
extends javax.swing.JPanel
implements java.awt.event.ActionListener

Nested Class Summary

(package private) class	XmlTag.XmlEditorRileva

Nested classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary	
protected javax.swing.JButton	**aggiungiattributonodoxmltag**
protected javax.swing.JButton	**aggiunginodoxmltag**
protected javax.swing.JTree	**alberoXmlTag**
protected javax.swing.JButton	**apritag**
protected javax.swing.JButton	**cancellaattributonodoxmltag**
protected javax.swing.JButton	**cancellanodoxmltag**
protected javax.swing.JButton	**cancellaSelezioneJTable**
protected org.w3c.dom.Document	**documentoXmlTag**
protected javax.swing.JButton	**editanodoxmltag**
protected javax.swing.tree.DefaultTreeCellEditor	**editorCellaAlbero**
protected java.io.File	**fileXmlTagCorrente**
protected javax.swing.tree.DefaultTreeModel	**modelloAlberoXmlTag**
protected TavolaAttributi	**modelloTavolaAttributiXmlTag**
protected javax.swing.JButton	**modificaattributonodoxmltag**
protected org.w3c.dom.Node	**nodoAttualmenteEditato**
static java.lang.String	**nomeClasseXmlTag**
protected javax.swing.JButton	**nuovotag**
(package private) XMLStudio	**oggettoXMLStudioGlobale**
protected javax.swing.JButton	**salvanometag**
protected javax.swing.JButton	**salvatag**
protected javax.swing.JTable	**tavolaAttributiXmlTag**

Fields inherited from class javax.swing.JPanel

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

XmlTag(XMLStudio oggettoXMLStudio)	

Method Summary

protected	void	**abilitaPulsantiAttributi**()
protected	void	**abilitaPulsantiNodi**()
	void	**actionPerformed**(java.awt.event.ActionEvent e)
	void	**aggiornaAttributiNodoInTavola**(org.w3c.dom.Node
protected	void	**azioneAggiungiAttributoNodo**()

protected void	**azioneAggiungiNodoXmlTag**()
protected void	**azioneApriDocumentoXmlTag**()
protected void	**azioneCancellaNodo**()
protected void	**azioneCancellaSelezioneJTable**()
protected void	**azioneEditaNodoXmlTag**()
protected void	**azioneEliminaAttributoNodo**()
protected void	**azioneModificaAttributoNodo**()
protected void	**azioneNuovoDocumentoXmlTag**()
protected void	**azioneRicordaSalva**()
protected void	**azioneSalvaDocumentoXmlTag**(boolean salvaNome)
protected void	**azioneSalvaGenerica**(java.io.File filesalva)
protected void	**caricaDocumentoXmlTag**(java.lang.String fileXml
private javax.swing.tree.DefaultMutableTreeNode	**creaAlberoNodi**(org.w3c.dom.Node nodoRadice)
javax.swing.JToolBar	**creaToolBar**()
private static void	**espandiAlbero**(javax.swing.JTree alberoJTree)
private static void	**espandiAlbero**(javax.swing.JTree alberoJTree, javax.swing.tree.TreePath path, javax.swing.tree.TreeNode nodo)
private static boolean	**nomeXmlValido**(java.lang.String testo)
private boolean	**possoVisualizzareNodo**(org.w3c.dom.Node nodocorr
org.w3c.dom.Node	**restituisciNodoSelezionato**()
XmlNodo	**restituisciNodoSelezionatoAlbero**()

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

nuovotag

```
protected javax.swing.JButton nuovotag
```

---

apritag

```
protected javax.swing.JButton apritag
```

---

salvatag

```
protected javax.swing.JButton salvatag
```

---

salvanometag

```
protected javax.swing.JButton salvanometag
```

---

aggiunginodoxmltag

```
protected javax.swing.JButton aggiunginodoxmltag
```

---

editanodoxmltag

```
protected javax.swing.JButton editanodoxmltag
```

---

cancellanodoxmltag

```
protected javax.swing.JButton cancellanodoxmltag
```

---

aggiungiattributonodoxmltag

```
protected javax.swing.JButton aggiungiattributonodoxmltag
```

---

modificaattributonodoxmltag

```
protected javax.swing.JButton modificaattributonodoxmltag
```

---

cancellaattributonodoxmltag

```
protected javax.swing.JButton cancellaattributonodoxmltag
```

---

cancellaSelezioneJTable

```
protected javax.swing.JButton cancellaSelezioneJTable
```

---

nomeClasseXmlTag

```
public static final java.lang.String nomeClasseXmlTag
```

See Also:

Constant Field Values

---

documentoXmlTag

```
protected org.w3c.dom.Document documentoXmlTag
```

---

fileXmlTagCorrente

```
protected java.io.File fileXmlTagCorrente
```

---

alberoXmlTag

```
protected javax.swing.JTree alberoXmlTag
```

---

modelloAlberoXmlTag

```
protected javax.swing.tree.DefaultTreeModel modelloAlberoXmlTag
```

---

editorCellaAlbero

```
protected javax.swing.tree.DefaultTreeCellEditor editorCellaAlbero
```

---

nodoAttualmenteEditato

```
protected org.w3c.dom.Node nodoAttualmenteEditato
```

---

tavolaAttributiXmlTag

```
protected javax.swing.JTable tavolaAttributiXmlTag
```

---

modelloTavolaAttributiXmlTag

```
protected TavolaAttributi modelloTavolaAttributiXmlTag
```

---

oggettoXMLStudioGlobale

```
XMLStudio oggettoXMLStudioGlobale
```

Constructor Detail

XmlTag

```
public XmlTag(XMLStudio oggettoXMLStudio)
```

Method Detail

creaToolBar

```
public javax.swing.JToolBar creaToolBar()
```

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

restituisciNodoSelezionatoAlbero

```
public XmlNodo restituisciNodoSelezionatoAlbero()
```

---

restituisciNodoSelezionato

```
public org.w3c.dom.Node restituisciNodoSelezionato()
```

---

aggiornaAttributiNodoInTavola

```
public void aggiornaAttributiNodoInTavola(org.w3c.dom.Node nodo)
```

---

creaAlberoNodi

```
private javax.swing.tree.DefaultMutableTreeNode
creaAlberoNodi(org.w3c.dom.Node nodoRadice)
```

---

espandiAlbero

```
private static void espandiAlbero(javax.swing.JTree alberoJTree)
```

---

espandiAlbero

```
private static void espandiAlbero(javax.swing.JTree alberoJTree,
                                  javax.swing.tree.TreePath path,
                                  javax.swing.tree.TreeNode nodo)
```

---

possoVisualizzareNodo

```
private boolean possoVisualizzareNodo(org.w3c.dom.Node nodocorrente)
```

---

azioneNuovoDocumentoXmlTag

```
protected void azioneNuovoDocumentoXmlTag()
```

---

azioneApriDocumentoXmlTag

```
protected void azioneApriDocumentoXmlTag()
```

---

caricaDocumentoXmlTag

```
protected void caricaDocumentoXmlTag(java.lang.String fileXmlTag)
```

---

azioneSalvaGenerica

```
protected void azioneSalvaGenerica(java.io.File filesalva)
```

---

azioneSalvaDocumentoXmlTag

```
protected void azioneSalvaDocumentoXmlTag(boolean salvaNome)
```

---

azioneRicordaSalva

```
protected void azioneRicordaSalva()
```

---

azioneAggiungiNodoXmlTag

```
protected void azioneAggiungiNodoXmlTag()
```

---

azioneEditaNodoXmlTag

```
protected void azioneEditaNodoXmlTag()
```

---

azioneCancellaNodo

```
protected void azioneCancellaNodo()
```

---

azioneAggiungiAttributoNodo

```
protected void azioneAggiungiAttributoNodo()
```

---

azioneModificaAttributoNodo

```
protected void azioneModificaAttributoNodo()
```

---

azioneEliminaAttributoNodo

```
protected void azioneEliminaAttributoNodo()
```

---

nomeXmlValido

```
private static boolean nomeXmlValido(java.lang.String testo)
```

---

abilitaPulsantiNodi

```
protected void abilitaPulsantiNodi()
```

---

abilitaPulsantiAttributi

```
protected void abilitaPulsantiAttributi()
```

---

azioneCancellaSelezioneJTable

```
protected void azioneCancellaSelezioneJTable()
```

LISTATI DEL PROGRAMMA

Listato AboutBox.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · AboutBox.java - Definizione classe per l'implementazione dell'AboutBox
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;


// **** definizione classe per definire l'AboutBox ****

class AboutBox extends JDialog implements ActionListener, WindowListener
{
	// costruttore della classe

	public AboutBox(JFrame parent)
	{
		super(parent, "Informazioni sul programma", true); // richiama la superclasse per creare la JDialog relativa all'AboutBox

		JPanel pannello1, pannello2, pannello3;

		JLabel label1 = new JLabel(new ImageIcon("./risorse/icona.gif")); // aggiunge l'immagine alla label
		Border b1 = new BevelBorder(BevelBorder.LOWERED); // definisce un bordo smussato
		Border b2 = new EmptyBorder(5, 5, 5, 5); // definisce un bordo vuoto con le dimensioni specificate, alto, sinistra, basso, destra
		label1.setBorder(new CompoundBorder(b1, b2)); // definizione di un bordo nella label composto da due bordi in uno singolo
		pannello1 = new JPanel(); // definisce un pannello e vi aggiunge la label
		pannello1.add(label1);

		String messaggio = "XMLStudio - Versione 1.0\n" + "Editor integrato per documenti XML";
		JTextArea areaTxt = new JTextArea(messaggio); // crea una JTextArea e vi aggiunge il messaggio
		areaTxt.setBorder(new EmptyBorder(5, 10, 5, 10)); // imposta nell'area testuale un bordo empty
		areaTxt.setFont(new Font("Helvetica", Font.BOLD, 12)); // imposta il carattere per l'area di testo
		areaTxt.setEditable(false); // quest'area di testo non e' editabile
		areaTxt.setBackground(getBackground()); // imposta il background dell'area di testo come il background attuale della JDialog
		pannello2 = new JPanel();
		pannello2.setLayout(new BoxLayout(pannello2, BoxLayout.Y_AXIS)); // dispone il pannello2 secondo questo layout verticale
		pannello2.add(areaTxt); // aggiunge al pannello2 la prima frase

		messaggio = "Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it\n\nDocente relatore: Prof.ssa Paola Giannini - giannini(
		JTextArea areaTxt2 = new JTextArea(messaggio); // crea una JTextArea e vi aggiunge il messaggio
```

```
                areaTxt2.setBorder(new EmptyBorder(5, 10, 5, 10)); // imposta nell'area testuale un bordo empty
                areaTxt2.setFont(new Font("Arial", Font.PLAIN, 12)); // imposta il carattere per l'area di testo
                areaTxt2.setEditable(false); // quest'area di testo non e' editabile
                areaTxt2.setBackground(getBackground()); // imposta il background dell'area di testo come il background attuale della JDialog
                pannello2.add(areaTxt2); // aggiunge al pannello2 la seconda frase

                JButton bottoneOK = new JButton("OK"); // crea un bottone "OK"
                bottoneOK.addActionListener(this);
                pannello3 = new JPanel(); // aggiunge il bottone "OK" ad un nuovo pannello3
                pannello3.add(bottoneOK);

                getContentPane().add(pannello1, BorderLayout.WEST); // aggiunge il pannello contenente l'immagine alla finestra
                getContentPane().add(pannello2, BorderLayout.CENTER); // aggiunge il pannello contenente le due frasi al centro della finestra
                getContentPane().add(pannello3, BorderLayout.SOUTH); // aggiunge il pannello contenente il bottone OK nella parte bassa della finestra

                pack(); // causa alla finestra di essere ridimensionata secondo la dimensione dei suoi sottocomponenti
                Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // calcola le dimensioni dello schermo
                setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // sposta la JDialog nella posizione centrale
                setVisible(true); // la JDialog è visibile
                setResizable(false); // non è possibile modificare la dimensione della JDialog
                addWindowListener(this);
        }

        // **** metodi invocati quando avviene un action event ****

        public void actionPerformed(ActionEvent e)
        {
                String evento = e.getActionCommand();

                if (evento.equals("OK"))
                {
                        setVisible(false);
                        dispose();
                }

        }

        // **** il metodo windowClosing gestisce la chiusura della finestra ****

        public void windowClosing(WindowEvent event)
        {
                setVisible(false);
                dispose();
        }

// **** sette metodi relativi alla finestra dell'editor, implementabili se necessario ****

        public void windowIconified(WindowEvent event)
        {
        }

        public void windowDeiconified(WindowEvent event)
        {
        }

        public void windowClosed(WindowEvent event)
        {
        }

        public void windowDeactivated(WindowEvent event)
```

```
        {
        }

        public void windowActivated(WindowEvent event)
        {
        }

        public void windowOpened(WindowEvent event)
        {
        }
}
```

Listato CopiaCorretta.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · CopiaCorretta.java - Definizione classe per la rimozione delle linee tab
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.io.*;
import java.util.regex.*;


// **** classe per la scrittura del file xml privato delle linee vuote (nel file dei tag xml) contenenti solo caratteri tab, \t, \n ****

class CopiaCorretta
{
	// costruttore della classe

	public CopiaCorretta(File origineFile)
	{
		String linea; // definisce una singola linea da analizzare
		int c; // definisce un singolo carattere per la lettura del file carattere per carattere

		try
		{
		BufferedReader in = new BufferedReader(new FileReader(origineFile));
		BufferedWriter out = new BufferedWriter(new FileWriter("./tag/TagXmlBackup.xml")); // file di backup del corrente file di tag xml

		    while ((linea = in.readLine()) != null) // preleva una linea dal file di origine e controlla che non sia nulla
		    {
			if (LineaNonVuota(linea)) // solo se la linea ha dei caratteri che non sono solo di tabulazione o di invio allora
			{
				out.write(linea, 0, linea.length()); // scrivi la linea su file
				out.newLine(); // scrivi una nuova linea
			}
		}

		in.close(); // chiude gli stream, ha realizzato il backup di origineFile privato di caratteri \t e \n
		out.close();

		in = new BufferedReader(new FileReader("./tag/TagXmlBackup.xml"));
		out = new BufferedWriter(new FileWriter(origineFile)); // file da riscrivere ora senza linee \t e \n

		while ((c = in.read()) != -1)
		out.write(c);

		in.close(); // ora origineFile e' stato riscritto senza caratteri \t e \n
		out.close();
	}
```

```
		catch (Exception e)
		{
			JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(null, "Informazione", "Eccezione rilevata nel salvataggio del nuovo documento ta
		}
		}


		// **** controlla se lineaInput contiene anche dei caratteri oppure solo i \t (e quindi i \n) ****

		private boolean LineaNonVuota(String lineaInput)
		{
			Pattern p = Pattern.compile(".*[a-zA-Z]+.*"); // individua qualsiasi carattere seguito da almeno 1 occorrenza di una lettera, seguita da qualsias
			Matcher m = p.matcher(lineaInput);
			boolean b = m.matches(); // in b vi e' l'informazione se lineaInput e' conforme al pattern definito nell'espressione regolare

//			if (b == true)
//				System.out.println(lineaInput + "#true");
//			else
//				System.out.println(lineaInput + "#false");

			return b;
		}
}
```

Listato CreaDialog.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · CreaDialog.java - Definizione classe per la creazione di una finestra Dialog
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


// **** definizione classe per creare finestra Dialog ****

// Una Dialog e' una finestra top-level con un titolo e un bordo, tipicamente usata per acquisire input dall'utente

class CreaDialog extends Dialog implements ActionListener, WindowListener
{

	// componenti utilizzati per le finestre dialog

	JTextField jtesto1, jtesto2, jtesto3, jtesto4; // componente per l'editing di una singola linea di testo

	JButton pulsanteUno, pulsanteDue, pulsanteTre, cancel;

	// restituisce i seguenti valori

	static String input, input2, input3, input4;

	// esempio CreaDialog cd = new CreaDialog(this, "Vai alla linea numero", true, "1 - " + areaTesto.getLineCount(), "Vai"); // crea una finestra Dialog

	public CreaDialog(Frame parent, String titolo, boolean tipo, String intervallo, String NomePulsante)
	{
		super(parent, titolo, tipo);

		jtesto1= new JTextField(18); // crea un campo testuale per l'editing di una singola linea lungo 18 caratteri
		pulsanteUno = new JButton(NomePulsante); // crea un nuovo pulsante
		pulsanteUno.addActionListener(this); // aggiunge al pulsante la possibilità di rilevare eventi
		JLabel jlabel = new JLabel(intervallo); // crea una nuova JLabel con "intervallo" come testo
		JPanel pannello = new JPanel(); // crea un nuovo pannello contenitore
		pannello.add(jlabel); // aggiunge la jlabel al pannello
		pannello.add(jtesto1); // aggiunge il campo testuale al pannello
		pannello.add(pulsanteUno); // aggiunge il pulsante al pannello
		add(pannello); // aggiunge il pannello
		pack(); // causa alla Dialog di essere ridimensionata secondaLabelo il layout e la dimensione dei suoi sottocomponenti

		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // calcola le dimensioni dello schermo
		setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // posiziona al centro dello schermo
```

```
		addWindowListener(this);  // per la gestione dell'evento WindowClosing della finestra dell'editor
		setVisible(true); // mostra la Dialog
}

// esempio CreaDialog cd = new CreaDialog(this, "Sostituisci", true, "Trova", "Sostituisci", "Ok", "Annulla");

public CreaDialog(Frame parent, String titolo, boolean tipo, String s1, String s2, String pulsanteOk, String pulsanteAnnulla)
{
		super(parent, titolo, tipo);

		JLabel primaLabel = new JLabel(s1);
		JLabel secondaLabel = new JLabel(s2);

		jtesto1 = new JTextField(8); // crea un campo testuale per l'editing di una singola linea lungo 8 caratteri
		jtesto2 = new JTextField(8);

		pulsanteTre = new JButton(pulsanteOk); // crea due pulsanti
		cancel = new JButton(pulsanteAnnulla);
		pulsanteTre.addActionListener(this);
		cancel.addActionListener(this);


		JPanel pannello = new JPanel(); // aggiunge a pannello primaLabel
		pannello.add(primaLabel);
		pannello.add(jtesto1);

		JPanel pannello2 = new JPanel(); // aggiunge a pannello2 secondaLabel
		pannello2.add(secondaLabel);
		pannello2.add(jtesto2);

		JPanel pannello3 = new JPanel(); // aggiunge a pannello4 i due pulsanti
		pannello3.add(pulsanteTre);
		pannello3.add(cancel);

		setLayout(new GridLayout(3,1,2,2)); // imposta il GridLayout ed aggiunge i quattro pannelli
		add(pannello);
		add(pannello2);
		add(pannello3);

		pack(); // causa alla Dialog di essere ridimensionata secondaLabelo il layout e la dimensione dei suoi sottocomponenti
		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // calcola le dimensioni dello schermo
		setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // posiziona al centro dello schermo

		addWindowListener(this);  // per la gestione dell'evento WindowClosing della finestra dell'editor
		setVisible(true); // mostra la Dialog
}

// esempio CreaDialog cd = new CreaDialog(this, "Imposta la dimensione finestra", true, "X (800)", "Y (600)", "Pannello orizzontale (400)", "Pannello verti

public CreaDialog(Frame parent, String titolo, boolean tipo, String s1, String s2, String s3, String s4, String pulsanteOk, String pulsanteAnnulla)
{
		super(parent, titolo, tipo);

		JLabel primaLabel = new JLabel(s1);
		JLabel secondaLabel = new JLabel(s2);
		JLabel terzaLabel = new JLabel(s3);
		JLabel quartaLabel = new JLabel(s4);

		jtesto1 = new JTextField(8); // crea un campo testuale per l'editing di una singola linea lungo 8 caratteri
		jtesto2 = new JTextField(8);
```

```
		jtesto3 = new JTextField(4);
		jtesto4 = new JTextField(4);

		pulsanteDue = new JButton(pulsanteOk); // crea due pulsanti
		cancel = new JButton(pulsanteAnnulla);
		pulsanteDue.addActionListener(this);
		cancel.addActionListener(this);

		JPanel pannello = new JPanel(); // aggiunge a pannello primaLabel
		pannello.add(primaLabel);
		pannello.add(jtesto1);

		JPanel pannello2 = new JPanel(); // aggiunge a pannello2 secondaLabel
		pannello2.add(secondaLabel);
		pannello2.add(jtesto2);

		JPanel pannello3 = new JPanel(); // aggiunge a pannello3 terzaLabel
		pannello3.add(terzaLabel);
		pannello3.add(jtesto3);

		JPanel pannello4 = new JPanel(); // aggiunge a pannello3 quartaLabel
		pannello4.add(quartaLabel);
		pannello4.add(jtesto4);

		JPanel pannello5 = new JPanel(); // aggiunge a pannello4 i due pulsanti
		pannello5.add(pulsanteDue);
		pannello5.add(cancel);

		setLayout(new GridLayout(5,1,2,2)); // imposta il GridLayout ed aggiunge i quattro pannelli
		add(pannello);
		add(pannello2);
		add(pannello3);
		add(pannello4);
		add(pannello5);

		pack(); // causa alla Dialog di essere ridimensionata secondaLabelo il layout e la dimensione dei suoi sottocomponenti
		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // calcola le dimensioni dello schermo
		setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // posiziona al centro dello schermo

		addWindowListener(this);  // per la gestione dell'evento WindowClosing della finestra dell'editor
		setVisible(true); // mostra la Dialog
}

// risponde ad azioni dell'utente ed esegue del codice: in attesa di eventi dai pulsanti

public void actionPerformed(ActionEvent e)
{

		String event = e.getActionCommand();

		if (e.getSource() == pulsanteUno) // e' stata rilevata la pressione del pulsanteUno
		{
			input = jtesto1.getText().trim(); // preleva il testo e rimuove gli spazi bianchi
			if(input.length() == 0)
				input = null;
			dispose(); // chiude la finestra Dialog
		}
		else
			if (e.getSource() == pulsanteTre) // e' stata rilevata la pressione del pulsanteDue
			{
```

```
                        input = jtesto1.getText().trim(); // preleva il testo e rimuove gli spazi bianchi
                        input2 = jtesto2.getText().trim();

                        if(input.length() == 0)
                                input = null;
                if(input2.length() == 0)
                                input2 = null;

                        dispose(); // chiude la finestra Dialog
                }
                else
                        if (e.getSource() == pulsanteDue) // e' stata rilevata la pressione del pulsanteDue
                        {
                                input = jtesto1.getText().trim(); // preleva il testo e rimuove gli spazi bianchi
                                input2 = jtesto2.getText().trim();
                                input3 = jtesto3.getText().trim();
                                input4 = jtesto4.getText().trim();

                                if(input.length() == 0)
                                        input = null;
                                if(input2.length() == 0)
                                        input2 = null;
                                if(input3.length() == 0)
                                        input3 = null;
                                if(input4.length() == 0)
                                        input4 = null;

                                dispose(); // chiude la finestra Dialog
                        }
                        else
                                if (e.getSource() == cancel) // e' stata rilevata la pressione del pulsante Cancel
                                {
                                        dispose(); // chiude la finestra Dialog
                                }
}

// il metodo getString restituisce l'input inserito dall'utente nel primo campo

public static String getString()
{
        return input;
}

// il metodo getString2 restituisce l'input inserito dall'utente nel secondo campo

public static String getString2()
{
        return input2;
}

// il metodo getString3 restituisce l'input inserito dall'utente nel terzo campo

public static String getString3()
{
        return input3;
}

// il metodo getString4 restituisce l'input inserito dall'utente nel quarto campo

public static String getString4()
```

```
		{
			return input4;
		}

		// **** il metodo windowClosing gestisce la chiusura della finestra ****

		public void windowClosing(WindowEvent event)
		{
			dispose(); // chiude la finestra Dialog
		}

	// **** sette metodi relativi alla finestra Dialog, implementabili se necessario ****

		public void windowIconified(WindowEvent event)
		{
		}

		public void windowDeiconified(WindowEvent event)
		{
		}

		public void windowClosed(WindowEvent event)
		{
		}

		public void windowDeactivated(WindowEvent event)
		{
		}

		public void windowActivated(WindowEvent event)
		{
		}

		public void windowOpened(WindowEvent event)
		{
		}
}
```

Listato DTDGenerator.java

```
/* ************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · DTDGenerator.java - Definizione classe per la generazione del file DTD

   - Codice originale di Michael H.Kay, versione 7.0.
   - DTDGenerator may be freely used, distributed, or modified, under the terms
   of the Mozilla Public License Version 1.0.

   http://users.iclway.co.uk/mhkay/saxon/saxon5-5-1/dtdgen.html

  ************************************************************************ */


// **** importa i packages utili ****

import org.xml.sax.*;
import java.util.*;
import java.io.*;
import javax.xml.parsers.SAXParserFactory;


// **** definizione classe per salvare un file .dtd a partire da un file .xml ****

public class DTDGenerator extends org.xml.sax.helpers.DefaultHandler
{
   protected static int MIN_ENUMERATION_INSTANCES = 10; // minimum number of appearances of an attribute for it to be considered a candidate for an enumeration t
   protected static int MAX_ENUMERATION_VALUES = 20; // maximum number of distinct attribute values to be included in an enumeration
   protected static int MIN_ENUMERATION_RATIO = 3; // an attribute will be regarded as an enumeration attribute only if the number of instances divided by the number

   protected static int MIN_FIXED = 5; // minimum number of attributes that must appear, with the same value each time, for the value to be regarded as FIXED
   protected static int MIN_ID_VALUES = 10; // minumum number of attribute values that must appear for the attribute to be regarded as an ID value
   protected static int MAX_ID_VALUES = 100000; // maximum number of attribute values to be saved while checking for uniqueness

   TreeMap elementList; // alphabetical list of element types appearing in the document; each has the element name as a key and an ElementDetails object as the value

   Stack elementStack;  // stack of elements currently open; each entry is a StackEntry object


   // **** costruttore della classe DTDGenerator ****

   public DTDGenerator()
   {
      elementList = new TreeMap();
      elementStack = new Stack();
   }


   // **** avvia l'analisi del file Xml attualmente caricato ****

   public void run(String filename)
   {
```

```
    try
    {
        InputSource is = new InputSource(new File(filename).toURL().toString());
        XMLReader parser = SAXParserFactory.newInstance().newSAXParser().getXMLReader();
        parser.setContentHandler(this);
        parser.parse(is);
    }
    catch (java.io.FileNotFoundException nf)
    {
        System.err.println("File " + filename + " non trovato.");
    }
    catch (Exception err)
    {
        System.err.println("Eccezione rilevata durante la fase di parsing del file xml per la creazione del file DTD.");
        System.err.println(err.getMessage());
        err.printStackTrace();
        System.exit(2);
    }
}


/**
* Test whether a string is an XML name.
* TODO: This is currently an incomplete test, it treats all non-ASCII characters
* as being valid in names.
*/

private boolean isValidName(String s)
{
    if (!isValidNMTOKEN(s)) return false;
    int c = s.charAt(0);
    return ! ((c>=0x30 && c<=0x39) || c=='.' || c=='-' );
}


/**
* Test whether a string is an XML NMTOKEN.
* TODO: This is currently an incomplete test, it treats all non-ASCII characters
* as being valid in NMTOKENs.
*/

private boolean isValidNMTOKEN(String s)
{
    if (s.length()==0) return false;
    for (int i=0; i<s.length(); i++)
    {
        int c = s.charAt(i);
        if (!( (c>=0x41 && c<=0x5a) ||
               (c>=0x61 && c<=0x7a) ||
               (c>=0x30 && c<=0x39) ||
                c=='.' ||
                c=='_' ||
                c=='-' ||
                c==':' ||
                c>128 ))
            return false;
    }
    return true;
}
```

```
/**
* When the whole document has been analysed, construct the DTD
*/

public void printDTD(String nomeSalvaDTD)
{
	try // scrive su file l'albero relativo ai tag xml
	{
		FileWriter out = new FileWriter(new File(nomeSalvaDTD));

		// process the element types encountered, in turn

		Iterator e=elementList.keySet().iterator();
		while (e.hasNext())
		{
		  String elementname = (String) e.next();
		  ElementDetails ed = (ElementDetails) elementList.get(elementname);
		  TreeMap children = ed.children;
		  Set childKeys = children.keySet();

		  //EMPTY content
		  if (childKeys.size()==0 && !ed.hasCharacterContent)
		    out.write("<!ELEMENT " + elementname + " EMPTY >\n");

		  //CHARACTER content
		  if (childKeys.size()==0 && ed.hasCharacterContent)
		    out.write("<!ELEMENT " + elementname + " ( #PCDATA ) >\n");

		  //ELEMENT content
		  if (childKeys.size()>0 && !ed.hasCharacterContent)
		  {
		    out.write("<!ELEMENT " + elementname + " ( ");

		    if (ed.sequenced)
		    {
		      // all elements of this type have the same child elements
		      // in the same sequence, retained in the childseq vector

		      Enumeration c = ed.childseq.elements();
		      while (true)
		      {
		        ChildDetails ch = (ChildDetails)c.nextElement();
		        out.write(ch.name);
		        if (ch.repeatable && !ch.optional)
		          out.write("+");
		        if (ch.repeatable && ch.optional)
		          out.write("*");
		        if (ch.optional && !ch.repeatable)
		          out.write("?");
		        if (c.hasMoreElements())
		          out.write(", ");
		        else
		          break;
		      }
		      out.write(" ) >\n");
		    }
		    else
		    {
		      // the children don't always appear in the same sequence; so
```

```
      // list them alphabetically and allow them to be in any order

      Iterator c1 = childKeys.iterator();
      while (c1.hasNext())
      {
         out.write((String)c1.next());
         if (c1.hasNext()) out.write(" | ");
      }
      out.write(" )* >\n");
   }
};

//MIXED content
if (childKeys.size()>0 && ed.hasCharacterContent)
{
   out.write("<!ELEMENT " + elementname + " ( #PCDATA");
   Iterator c2 = childKeys.iterator();
   while (c2.hasNext())
   {
      out.write(" | " + (String)c2.next());
   }
   out.write(" )* >\n");
};

// Now examine the attributes encountered for this element type

TreeMap attlist = ed.attributes;
boolean doneID = false;      // to ensure we have at most one ID attribute per element
Iterator a=attlist.keySet().iterator();
while (a.hasNext())
{
   String attname = (String) a.next();
   AttributeDetails ad = (AttributeDetails) attlist.get(attname);

   // If the attribute is present on every instance of the element, treat it as required
   boolean required = (ad.occurrences==ed.occurrences);

   // If every value of the attribute is distinct,
   // and there are at least MIN_ID_VALUES, treat it as an ID
   // TODO: this may give the wrong answer, we should check whether the value sets of two
   // candidate-ID attributes overlap, in which case they can't both be IDs !!)
   boolean isid = ad.allNames &&          // ID values must be Names
                  (!doneID) &&          // Only allowed one ID attribute per element type
                  (ad.unique) &&
                  (ad.occurrences>=MIN_ID_VALUES);

   // if there is only one attribute value, and at least MIN_FIXED occurrences of it,
   // treat it as FIXED
   boolean isfixed = required && ad.values.size()==1 && ad.occurrences >= MIN_FIXED;

   // if the number of distinct values is small compared with the number of occurrences,
   // treat it as an enumeration
   boolean isenum = ad.allNMTOKENs &&     // Enumeration values must be NMTOKENs
                  (ad.occurrences>=MIN_ENUMERATION_INSTANCES) &&
                  (ad.values.size()<=ad.occurrences/MIN_ENUMERATION_RATIO) &&
                  (ad.values.size()<=MAX_ENUMERATION_VALUES);

   out.write("<!ATTLIST " + elementname + " " + attname + " ");
   String tokentype = (ad.allNMTOKENs ? "NMTOKEN" : "CDATA");
```

```
                    if (isid)
                    {
                       out.write("ID");
                       doneID = true;
                    }
                    else if (isfixed)
                    {
                       String val = (String) ad.values.first();
                       out.write(tokentype + " #FIXED \"" + escape(val) + "\" >\n");
                    }
                    else if (isenum)
                    {
                       out.write("( ");
                       Iterator v = ad.values.iterator();
                       while (v.hasNext())
                       {
                          out.write((String) v.next());
                          if (!v.hasNext()) break;
                          out.write(" | ");
                       };
                       out.write(" )");
                    }
                    else
                       out.write(tokentype);

                    if (!isfixed)
                    {
                       if (required)
                          out.write(" #REQUIRED >\n");
                       else
                          out.write(" #IMPLIED >\n");
                    }
                 };
                 out.write("\n");
              };
    out.close();
}
catch (Exception e)
{
          JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(null, "Informazione", "Eccezione rilevata nel salvataggio del file DTD: " + nomeSalvaDTD + ".
}
}


/**
* Escape special characters for display.
* @param ch The character array containing the string
* @param start The start position of the input string within the character array
* @param length The length of the input string within the character array
* @return The XML/HTML representation of the string<br>
* This static method converts a Unicode string to a string containing
* only ASCII characters, in which non-ASCII characters are represented
* by the usual XML/HTML escape conventions (for example, "&lt;" becomes "&amp;lt;").
* Note: if the input consists solely of ASCII or Latin-1 characters,
* the output will be equally valid in XML and HTML. Otherwise it will be valid
* only in XML.
* The escaped characters are written to the dest array starting at position 0; the
* number of positions used is returned as the result
*/
```

```
private static int escape(char ch[], int start, int length, char[] out)
{
   int o = 0;
   for (int i = start; i < start+length; i++)
   {
      if (ch[i]=='<') {("&lt;").getChars(0,4,out,o); o+=4;}
      else if (ch[i]=='>') {("&gt;").getChars(0,4,out,o); o+=4;}
      else if (ch[i]=='&') {("&amp;").getChars(0,5,out,o); o+=5;}
      else if (ch[i]=='\"') {("&#34;").getChars(0,5,out,o); o+=5;}
      else if (ch[i]=='\'') {("&#39;").getChars(0,5,out,o); o+=5;}
      else if (ch[i]<=0x7f) {out[o++]=ch[i];}
      else
      {
         String dec = "&#" + Integer.toString((int)ch[i]) + ';';
         dec.getChars(0, dec.length(), out, o);
         o+=dec.length();
      }
   }
   return o;
}


/**
* Escape special characters in a String value.
* @param in The input string
* @return The XML representation of the string<br>
* This static method converts a Unicode string to a string containing
* only ASCII characters, in which non-ASCII characters are represented
* by the usual XML/HTML escape conventions (for example, "&lt;" becomes
* "&amp;lt;").<br>
* Note: if the input consists solely of ASCII or Latin-1 characters,
* the output will be equally valid in XML and HTML. Otherwise it will be valid
* only in XML.
*/

private static String escape(String in)
{
   char[] dest = new char[in.length()*8];
   int newlen = escape(in.toCharArray(), 0, in.length(), dest);
   return new String(dest, 0, newlen);
}


/**
* Handle the start of an element. Record information about the position of this
* element relative to its parent, and about the attributes of the element.
*/

public void startElement (String uri, String localName, String name, Attributes attributes) throws SAXException
{
   StackEntry se = new StackEntry();

   // create an entry in the Element List, or locate the existing entry
   ElementDetails ed = (ElementDetails) elementList.get(name);
   if (ed==null)
   {
      ed = new ElementDetails(name);
      elementList.put(name,ed);
   };
```

```
// retain the associated element details object
se.elementDetails = ed;

// initialise sequence numbering of child element types
se.sequenceNumber = -1;

// count occurrences of this element type
ed.occurrences++;

// Handle the attributes accumulated for this element.
// Merge the new attribute list into the existing list for the element

for (int a=0; a<attributes.getLength(); a++)
{
   String attName = attributes.getQName(a);
   String val = attributes.getValue(a);

   AttributeDetails ad = (AttributeDetails) ed.attributes.get(attName);
   if (ad==null)
   {
     ad=new AttributeDetails(attName);
     ed.attributes.put(attName, ad);
   };

   if (!ad.values.contains(val))
   {
      // We haven't seen this attribute value before

      ad.values.add(val);

      // Check if attribute value is a valid name
      if (ad.allNames && !isValidName(val))
      {
         ad.allNames = false;
      }

      // Check if attribute value is a valid NMTOKEN
      if (ad.allNMTOKENs && !isValidNMTOKEN(val))
      {
         ad.allNMTOKENs = false;
      }

      // For economy, don't save the new value unless it's needed;
      // it's needed only if we're looking for ID values or enumerated values

      if (ad.unique && ad.allNames && ad.occurrences <= MAX_ID_VALUES)
      {
         ad.values.add(val);
      }
      else
        if (ad.values.size() <= MAX_ENUMERATION_VALUES)
         {
                        ad.values.add(val);
         }
   }
   else
   {
      // We've seen this attribute value before
      ad.unique = false;
   }
```

```
      ad.occurrences++;
   };

   // now keep track of the nesting and sequencing of child elements
   if (!elementStack.isEmpty())
   {
      StackEntry parent = (StackEntry)elementStack.peek();
      ElementDetails parentDetails = parent.elementDetails;
      int seq = parent.sequenceNumber;

      // for sequencing, we're interested in consecutive groups of the same child element type
      boolean isFirstInGroup = (parent.latestChild==null || (!parent.latestChild.equals(name)));
      if (isFirstInGroup)
      {
         seq++;
         parent.sequenceNumber++;
      }
      parent.latestChild = name;

      // if we've seen this child of this parent before, get the details
      TreeMap children = parentDetails.children;
      ChildDetails c = (ChildDetails)children.get(name);
      if (c == null)
      {
         // this is the first time we've seen this child belonging to this parent
         c = new ChildDetails();
         c.name = name;
         c.position = seq;
         c.repeatable = false;
         c.optional = false;
         children.put(name, c);
         parentDetails.childseq.addElement(c);

         // if the first time we see this child is not on the first instance of the parent,
         // then we allow it as an optional element
         if (parentDetails.occurrences!=1)
         {
            c.optional = true;
         }

      }
      else
      {
         // if it's the first occurrence of the parent element, and we've seen this
         // child before, and it's the first of a new group, then the child occurrences are
         // not consecutive
         if (parentDetails.occurrences==1 && isFirstInGroup)
         {
            parentDetails.sequenced = false;
         }

         // check whether the position of this group of children in this parent element is

         // the same as its position in previous instances of the parent.
         if (parentDetails.childseq.size()<=seq || !((ChildDetails)parentDetails.childseq.elementAt(seq)).name.equals(name))
         {
            parentDetails.sequenced = false;
         }
      }
```

```
      // if there's more than one child element, mark it as repeatable
      if (!isFirstInGroup)
      {
         c.repeatable = true;
      }
   }
   elementStack.push(se);
}


/**
* End of element. If sequenced, check that all expected children are accounted for.
*/

public void endElement (String uri, String localName, String name) throws SAXException
{
   // If the number of child element groups in this parent element is less than the
   // number in previous elements, then the absent children are marked as optional

   ElementDetails ed = (ElementDetails) elementList.get(name);
   if (ed.sequenced)
   {
      StackEntry se = (StackEntry)elementStack.peek();
      int seq = se.sequenceNumber;
      for (int i=seq+1; i<ed.childseq.size(); i++)
      {
         ((ChildDetails)ed.childseq.elementAt(i)).optional = true;
      }
   }
   elementStack.pop();
}


/**
* Handle character data.
* Make a note whether significant character data is found in the element
*/

public void characters (char ch[], int start, int length) throws SAXException
{
   ElementDetails ed = ((StackEntry)elementStack.peek()).elementDetails;
   if (!ed.hasCharacterContent)
   {
      for (int i=start; i<start+length; i++)
      {
         if ((int)ch[i] > 0x20)
         {
            ed.hasCharacterContent = true;
            break;
         }
      }
   }
}


/**
* ElementDetails is a data structure to keep information about element types
*/

private class ElementDetails
```

```
{
   String name;
   int occurrences;
   boolean hasCharacterContent;
   boolean sequenced;
   TreeMap children;
   Vector childseq;
   TreeMap attributes;

   public ElementDetails ( String name )
   {
      this.name = name;
      this.occurrences = 0;
      this.hasCharacterContent = false;
      this.sequenced = true;
      this.children = new TreeMap();
      this.childseq = new Vector();
      this.attributes = new TreeMap();
   }
}


/**
* ChildDetails records information about the presence of a child element within its
* parent element. If the parent element is sequenced, then the child elements always
* occur in sequence with the given frequency.
*/

private class ChildDetails
{
   String name;
   int position;
   boolean repeatable;
   boolean optional;
}


/**
* AttributeDetails is a data structure to keep information about attribute types
*/

private class AttributeDetails
{
   String name;          // name of the attribute
   int occurrences;      // number of occurrences of the attribute
   boolean unique;       // true if no duplicate values encountered
   TreeSet values;       // set of all distinct values encountered for this attribute
   boolean allNames;     // true if all the attribute values are valid names
   boolean allNMTOKENs;  // true if all the attribute values are valid NMTOKENs

   public AttributeDetails ( String name )
   {
      this.name = name;
      this.occurrences = 0;
      this.unique = true;
      this.values = new TreeSet();
      this.allNames = true;
      this.allNMTOKENs = true;
   }
}
```

```
    /**
    * StackEntry is a data structure we put on the stack for each nested element
    */

    private class StackEntry
    {
       ElementDetails elementDetails;
       int sequenceNumber;
       String latestChild;
    }

} // end of outer class DTDSAXGen
```

Listato FiltroFile.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · FiltroFile.java - Definizione classe per il filtro di file
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.io.*;

// **** definizione classe per impostare i filtri nella finestra dialog ****

class FiltroFile extends javax.swing.filechooser.FileFilter
{
	private String filtro_estensione = null;
	private String filtro_descrizione = null;

	// costruttore della classe

	public FiltroFile(String estensione, String descrizione)
	{
		filtro_estensione = "." + estensione.toLowerCase();
		filtro_descrizione = descrizione;
	}

	// restituisce la descrizione per il filtro

	public String getDescription()
	{
		return filtro_descrizione;
	}

	// se un determinato file e' accettato da questo filtro

	public boolean accept(File f)
	{
		if (f == null)		return false;
		if (f.isDirectory()) return true;
		return f.getName().toLowerCase().endsWith(filtro_estensione);
	}
}
```

Listato Guida.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · Guida.java - Definizione classe per la Guida in linea del programma
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.io.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;


// **** definizione classe per aprire la guida testuale che ricorda alcune informazioni utili ****

class Guida extends JDialog implements ActionListener
{
	private JButton pulsanteOk; // pulsanteOk per confermare il font scelto

	// **** costruttore della classe Guida ****

	// la JDialog appartiene ad un'altra finestra, il JFrame

	public Guida(JFrame parent) // al costruttore della classe che estende la JDialog passa il JFrame, finestra principale del programma
	{
		super(parent, "Guida testuale", true); // richiama la superclasse per creare la JDialog relativa alla Guida

		final Container contenitoreGuida = this.getContentPane(); // final = inizializzato e non modificabile il contentPane della JDialog
		contenitoreGuida.setLayout(new BorderLayout()); // impostazione del layout del ContentPane della JDialog

		JPanel Pannello = new JPanel(); // definisce un primo pannello
		pulsanteOk = new JButton("Ok");

		pulsanteOk.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent e)
			{
				dispose(); // chiude la JDialog con questo metodo di Java
			}
		});

		Pannello.add(pulsanteOk); // aggiunge il pulsanteOk al pannello
		contenitoreGuida.add("South", Pannello); // posiziona il Pannello (con il pulsante Ok) nella parte bassa della JDialog

		JPanel Pannello2 = new JPanel(); // definisce un secondo pannello
		Pannello2.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
		Pannello2.setLayout(new FlowLayout());
```

```
		//Pannello2.setLayout(new GridLayout(1,1));

		final JTextArea areaTestoGuida = new JTextArea();

		areaTestoGuida.setFont(new Font("Courier", Font.PLAIN, 12));
		areaTestoGuida.setEditable(false);
		JScrollPane areaTestoScorrimento = new JScrollPane(areaTestoGuida);

		//JViewport port = areaTestoScorrimento.getViewport();
		//port.add(areaTestoGuida);

		areaTestoScorrimento.setMinimumSize(new Dimension(520, 420)); // imposta la dimensione minima dell'area testo
		areaTestoScorrimento.setPreferredSize(new Dimension(520, 420)); // imposta la dimensione predefinita dell'area testo
		Pannello2.add(areaTestoScorrimento);

		contenitoreGuida.add("Center", Pannello2); // posiziona il Pannello2 (con l'areaTestoScorrimento) nella JDialog
		this.setSize(536, 440); // imposta la dimensione della JDialog

		try
		{
			File f = new File("./risorse/XMLStudio.guida"); // associa all'oggetto f il file selezionato relativo alla guida
			Reader in = new FileReader(f); // crea l'oggetto in per leggere il file
			areaTestoGuida.read(in, null); // riporta il contenuto dell'oggetto in su areaTestoGuida
		}
		catch(Exception e)
		{
			System.err.println("Errore di apertura del file.");
		 }

		pack(); // causa alla JDialog di essere ridimensionata secondo il layout e la dimensione dei suoi sottocomponenti
		Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
		this.setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // sposta la JDialog nella posizione centrale
		this.setVisible(true); // la JDialog è visibile
	}


	// **** risponde ad azioni dell'utente ed esegue del codice: in attesa di eventi dai pulsanti ****

	public void actionPerformed(ActionEvent evento) // rileva oggetti ActionEvent
	{
		Object oggettoScelto = evento.getSource(); // grazie ad oggettoScelto individua quale bottone ha causato l'evento

		if (oggettoScelto == pulsanteOk) // se viene premuto il pulsanteOK
		{
				dispose(); // chiude la JDialog dopo aver premuto il pulsanteCancella
		}
	}

}
```

Listato ImpostaFont.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · ImpostaFont.java - Definizione classe per la selezione del carattere
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;


// **** definizione classe per impostare i Font ****

class ImpostaFont extends JDialog implements ActionListener, ListSelectionListener
{

        GraphicsEnvironment ambiente = GraphicsEnvironment.getLocalGraphicsEnvironment(); // per la descrizione di oggetti Font in Java
        private String[] fontType = ambiente.getAvailableFontFamilyNames(); // restituisce string[] contenente l'elenco dei caratteri disponibili
        private String[] fontStyle = {"Grassetto","Corsivo","Normale"}; // font styles avialible
        private String[] fontSize = {"8","10","12","14","16","18","20","22","24","26","28","30"}; // dimensioni dei caratteri
        private String prev = "aAbByYzZ 123456789"; // oggetto stringa per il preview del font
        private JLabel preview = new JLabel(prev,JLabel.CENTER); // oggetto JLabel usato per la preview del font
        private Font fontScelto; // oggetto di tipo Font che contiene il font scelto dall'utente
        private JButton pulsanteOk; // pulsanteOk per confermare il font scelto
        private JButton pulsanteCancella; // pulsanteCancella per
        private JList listaFont; // lista per la visualizzazione dei font
        private JList listaStiliFont; // lista per la visualizzazione degli stili dei font
        private JList listaDimensioniFont; // lista per la visualizzazione delle dimensioni dei font


        // **** costruttore della classe ImpostaFont ****

        // il JDialog appartiene ad un'altra finestra, il JFrame

        public ImpostaFont(JFrame parent) // al costruttore della classe che estende la JDialog passa il JFrame, finestra principale del programma
        {
                // questo costruttore imposta il frame del font

                super(parent, "Impostazione del carattere", true); // richiama la superclasse per creare la JDialog relativa al font

                //this.setTitle("Impostazione del carattere"); // il titolo della JDialog è già impostato grazie al costrutto super

                pulsanteOk = new JButton("Ok"); // definisce il pulsante Ok
                pulsanteOk.addActionListener(this); // aggiunge al pulsante la possibilità di rilevare eventi
                pulsanteCancella = new JButton("Annulla"); // definisce il pulsante Annulla
                pulsanteCancella.addActionListener(this); // aggiunge al pulsante la possibilità di rilevare eventi
```

```
// definizione della Jlist relativa all'elenco dei font disponibili

listaFont = new JList(fontType); // non bisogna definire ActionListener per le liste
listaFont.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // imposta la selezione di un solo item nella Jlist
listaFont.setVisibleRowCount(10);
listaFont.setSelectedIndex(0);
listaFont.addListSelectionListener(this);

// definizione della Jlist relativa agli stili disponibili per i font

listaStiliFont = new JList(fontStyle);
listaStiliFont.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
listaStiliFont.setVisibleRowCount(10);
listaStiliFont.setSelectedIndex(0);

// definizione della Jlist relativa alla dimensione dei font

listaDimensioniFont = new JList(fontSize);
listaDimensioniFont.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
listaDimensioniFont.setVisibleRowCount(10);
listaDimensioniFont.setSelectedIndex(0);

// definizione di un JPanel per il posizionamento delle JList

JPanel pannelloListe = new JPanel(new GridLayout(1,5,3,15)); // righe, colonne, spazio orizzontale e verticale
pannelloListe.add(new JScrollPane(listaFont)); // aggiunge al JPanel i 3 controlli JList
pannelloListe.add(new JScrollPane(listaStiliFont));
pannelloListe.add(new JScrollPane(listaDimensioniFont));

// definizione di un JPanel per il posizionamento dei pulsanti

JPanel pannelloPulsanti = new JPanel();
pannelloPulsanti.add(pulsanteCancella);
pannelloPulsanti.add(pulsanteOk);

// aggiunta dei singoli componenti alla JDialog

getContentPane().setLayout(new BorderLayout(5,5)); // imposta il layout del Frame
getContentPane().add(preview,BorderLayout.NORTH); // posiziona la JLabel relativa alla preview
getContentPane().add(pannelloListe,BorderLayout.CENTER); // posiziona il JPanel pannelloListe sul Frame
getContentPane().add(pannelloPulsanti,BorderLayout.SOUTH); // posiziona il JPanel dei pulsanti sul Frame

// Un listener chiude la finestra quando l'utente preme il pulsante di chiusura della finestra

addWindowListener(new WindowAdapter() // classe esistente per agevolare la creazione di oggetti listener definendo solo metodi necessari
{

        // metodo che gestisce la chiusura della JDialog

public void windowClosing(WindowEvent e)
        {
                fontScelto = null;
                dispose(); // chiude la JDialog con questo metodo di Java
        }
});  // fine di addWindowListener

preview.setFont(new Font((String)listaFont.getSelectedValue(),Font.PLAIN,14));
pack(); // causa alla JDialog di essere ridimensionata secondo il layout e la dimensione dei suoi sottocomponenti
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
```

```
		setLocation((screenSize.width - getWidth()) / 2, (screenSize.height - getHeight()) / 2); // sposta la JDialog nella posizione centrale
		setVisible(true); // la JDialog è visibile
}


// **** metodo che restituisce il Font scelto ****

public Font returnFont()
{
		if(fontScelto == null) // se non è stato scelto alcun font, ritorna null
				return null;
		else
				return fontScelto;
}


// **** risponde ad azioni dell'utente ed esegue del codice: in attesa di eventi dai pulsanti ****

public void actionPerformed(ActionEvent evento) // rileva oggetti ActionEvent
{
		Object oggettoScelto = evento.getSource(); // grazie ad oggettoScelto individua quale bottone ha causato l'evento

		if (oggettoScelto == pulsanteOk) // se viene premuto il pulsanteOK
		{
				int valoreStile; // intero corrispondente allo stile selezionato

				String stileSelezionato = (String)listaStiliFont.getSelectedValue(); // inserisce nell'oggetto il valore selezionato dalla lista

				if (stileSelezionato.equals("Grassetto"))
						valoreStile = Font.BOLD;
				else
						if (stileSelezionato.equals("Corsivo"))
								valoreStile = Font.ITALIC;
						else
								if(stileSelezionato.equals("Normale"))
										valoreStile = Font.PLAIN;
				else // questo caso non si verifica
										valoreStile = Font.PLAIN;

				fontScelto = new Font((String)listaFont.getSelectedValue(), valoreStile, Integer.parseInt((String)listaDimensioniFont.getSelectedValu
				dispose(); // chiude la JDialog dopo aver premuto il pulsanteOK
		}
		else
				if(oggettoScelto == pulsanteCancella) // è stato premuto pulsanteCancella
				{
						fontScelto = null;
						dispose(); // chiude la JDialog dopo aver premuto il pulsanteCancella
				}
}


// **** richiamato ogni volta che cambia il valore selezionato dello stile carattere ****

public void valueChanged(ListSelectionEvent evento)
{
		int valoreStile; // intero corrispondente allo stile selezionato

		String stileSelezionato = (String)listaStiliFont.getSelectedValue(); // inserisce nell'oggetto il valore selezionato dalla lista

		//preview.setFont(new Font((String)listaFont.getSelectedValue(),Font.BOLD,12));
```

```
                if (stileSelezionato.equals("Grassetto"))
                        valoreStile = Font.BOLD;
                else
                        if (stileSelezionato.equals("Corsivo"))
                                valoreStile = Font.ITALIC;
                        else
                                if(stileSelezionato.equals("Normale"))
                                        valoreStile = Font.PLAIN;
                else // questo caso non si verifica
                                        valoreStile = Font.PLAIN;

                preview.setFont(new Font((String)listaFont.getSelectedValue(), valoreStile, Integer.parseInt((String)listaDimensioniFont.getSelectedValue())));
        }
}
```

Listato JOptionPaneEsteso.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · JOptionPaneEsteso.java - Definizione classe per realizzare JoptionPane
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import javax.swing.*;
import java.awt.Dimension;


// **** definizione classe per realizzare questo JOptionPane esteso ****

/* questo JOptionPane consente di specificare in testoMessaggio delle stringhe piu'
   lunghe rispetto a  quelle normalmente definibili nei normali JOptionPane */

public class JOptionPaneEsteso extends JDialog
{
        // **** costruttore della classe JOptionPaneEsteso ****

        public JOptionPaneEsteso(JFrame parent, String titolo, String testoMessaggio)
        {
                super(parent, "Impostazione del carattere", true); // richiama la superclasse per creare la JDialog

        JTextArea tmpAreaTesto = new JTextArea(testoMessaggio); // definisce una JTextArea che conterra' il nostro testo
        tmpAreaTesto.setLineWrap(true);
        tmpAreaTesto.setWrapStyleWord(true);
        tmpAreaTesto.setRows((testoMessaggio.length()/80)+1);
        tmpAreaTesto.setBackground(getContentPane().getBackground()); // lo sfondo della JTextArea deve essere analogo al JOptionPane

        JOptionPane pane = new JOptionPane(tmpAreaTesto, JOptionPane.WARNING_MESSAGE, JOptionPane.DEFAULT_OPTION );
        Dimension dimensionePane = new Dimension(400, 140); // imposta la dimensione del JOptionPaneEsteso
         pane.setPreferredSize(dimensionePane);
        JDialog dialog = pane.createDialog(getContentPane(), titolo);
        pack();
        dialog.setVisible(true);
  }
}
```

Listato PopupListener.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · PopupListener.java - Riceve eventi del mouse
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */

/* Consente di rilevare quando il mouse e' premuto o rilasciato, cliccato, ecc.
  Ridefizione dei metodi di interesse: mousePressed, mouseReleased */

// **** importa i packages utili ****

import java.awt.event.*;
import javax.swing.JPopupMenu;

// **** definizione classe per ricevere eventi del mouse ****

class PopupListener extends MouseAdapter
{
        JPopupMenu oggettoMenuJPopup = null;

        // **** costruttore della classe PopupListener ****

    public PopupListener(JPopupMenu parametroMenu)
    {
        oggettoMenuJPopup = parametroMenu;
    }

    // **** invocata quando il pulsante destro del mouse e' premuto su di un componente ****

    public void mousePressed(MouseEvent e)
    {
        mostraMenuPopup(e);
    }

    // **** invocata quando il pulsante del mouse e' rilasciato su di un componente

    public void mouseReleased(MouseEvent e)
    {
        mostraMenuPopup(e);
    }

    private void mostraMenuPopup(MouseEvent e)
    {
        if (e.isPopupTrigger())
        {
            oggettoMenuJPopup.show(e.getComponent(), e.getX(), e.getY());
        }
    }
}
```

Listato RenderAlbero.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · RenderAlbero.java - Definizione classe per realizzare il render dell'Albero
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.*;
import java.awt.event.*;
import javax.swing.JTree;
//import javax.swing.tree.DefaultMutableTreeNode;
//import javax.swing.event.TreeSelectionListener;
//import javax.swing.event.TreeSelectionEvent;
//import javax.swing.tree.TreeSelectionModel;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.ImageIcon;
//import javax.swing.ToolTipManager;
import org.w3c.dom.*;

/* Negli alberi JTree, se si desidera avere un controllo sulle icone (o sui tooltip) e' necessario creare
   una sottoclasse di nome "DefaultTreeCellRenderer" sovrapponendo un metodo getTreeCellRendererComponent.
   Restituisce una nuova istanza di DefaultTreeCellRenderer per mostrare un'appropriata icona grazie al
   metodo setIcon in base all'elemento del documento Xml. Icone e colori del testo sono determinati in base
   ai settaggi di UIManager.
   Quando un albero analizza e mostra ogni nodo, ne' il JTree e neppure il suo look-and-feel contengono
   informazioni relative alla visualizzazione del nodo. Invece l'albero utilizza il codice incluso in questa
   classe di rendering per mostrare ogni nodo. Per esempio, per mostrare una foglia dell'albero che ha un
   determinato valore stringa, l'albero interroga il Cell Renderer in modo che restituisca un componente che possa
   visualizzare un nodo foglia con quello specifico valore di stringa. Se il Cell Renderer e' un
   DefaultTreeCellRenderer, allora ritornera' una label che mostra l'iconda di default per le foglie seguita
   dalla stringa. Un Cell Renderer consente soltanto la visualizzazione, non gestisce eventi. */


// **** definizione classe per realizzare il render dell'albero ****

class RenderAlbero extends DefaultTreeCellRenderer
{
        // ImageIcon iconaFoglia;

        // **** costruttore della classe RenderAlbero ****

    public RenderAlbero()
    {
        // iconaFoglia = new ImageIcon("images/esempio.gif");
    }

    // configura il render basato sui componenti passati per mostrare icone personalizzate (ed eventualmente testo)
    public Component getTreeCellRendererComponent(JTree albero, Object valore, boolean sel, boolean espanso, boolean foglia, int riga, boolean haFocus)
```

```
	{
		// setLeafIcon(new ImageIcon("images/middle.gif"));
		Component componente = super.getTreeCellRendererComponent(albero, valore, sel, espanso, foglia, riga, haFocus);
		if (valore instanceof XmlNodo) // se value e' uguale al tipo XmlNodo che estende DefaultMutableTreeNode
		{
			// casting esplicito di value nel tipo XmlNodo in node
			Node nodoesaminato = ( (XmlNodo) valore).restituisciXmlNodo(); // estrae da valore il nodo esaminato
			if (nodoesaminato instanceof Element)
				setIcon(espanso ? openIcon : closedIcon);
			else
				setIcon(leafIcon);
			}
			return componente; // restituisce un component ovvero un oggetto grafico
	}
}
```

Listato StampaGrafica.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · StampaGrafica.java - Classe per la stampa del documento in modo grafico
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.*; // libreria grafica AWT
import java.io.StringReader; // uno stream di caratteri il cui input e' una stringa
import java.io.LineNumberReader; // tiene traccia dei numeri di linea: setLineNumber e getLineNumber imposta e restituisce il numero di linea
import java.io.EOFException; // segnala che la fine del file o dello stream e' stata raggiunta inaspettatamente durante l'input


// **** definizione classe per effettuare la stampa del documento utilizzando la metodologia grafica ****

// stampa del testo grazie all'uso di PrintJob; non considera il Word Wrap o i tab

public class StampaGrafica
{
	// costruttore della classe StampaGrafica

	public StampaGrafica(PrintJob jobStampa, Graphics oggettoGrafico, String testo, String nomeDocumento)
	{
		int numeroPagina = 1; // numero della pagina corrente da stampare; numero di pagine complessive
		int lineePagina = 0; // numero di linee della pagina corrente
		int lineeComplessive = 0; // numero di linee dell'itero documento
		int altezzaCorrente = 0; // altezza corrente del testo stampato nella pagina
		String lineaSuccessiva; // contiene una linea di testo letta dal parametro "testo"

		// nota: String non e' mutabile e quindi non muta durante la stampa
		if (!(oggettoGrafico instanceof PrintGraphics))
		{
			throw new IllegalArgumentException ("Errore di istanza PrintGraphics.");
		}
		StringReader readerStringa = new StringReader (testo); // oggetto stream di carattere il cui parametro in input e' stringa
		LineNumberReader readerNumeriLinea = new LineNumberReader (readerStringa); // crea un nuovo reader di numeri di linea
		int altezzaPagina = jobStampa.getPageDimension().height;

		// e' necessario impostare il font per poter avere l'output
		Font carattereHelvetica = new Font("carattereHelveticaetica", Font.PLAIN, 12);
		oggettoGrafico.setFont(carattereHelvetica); // imposta Helvetica come carattere di stampa
		FontMetrics oggettofm = oggettoGrafico.getFontMetrics(carattereHelvetica); // restituisce oggetto LineMetrics per lo specifico carattere
		int altezzaFont = oggettofm.getHeight(); // restituisce l'altezza standard di una linea di testo che usa questo carattere
		int descentFont = oggettofm.getDescent(); // restituisce il font descent per questo carattere
		try
		{
			System.out.println ("----> Stampa del documento " + nomeDocumento + " in corso <----");
		do // fai mentre vi sono righe da stampare nel documento
```

```
		{
			lineaSuccessiva = readerNumeriLinea.readLine(); // legge una linea di testo
			if (lineaSuccessiva != null) // se vi e' ancora del testo da stampare
			{
			if ((altezzaCorrente + altezzaFont) > altezzaPagina) // inizia una nuova pagina
			{
				System.out.println ("" + lineePagina + " linee stampate nella pagina " + numeroPagina);
				numeroPagina++; // incrementa il numero di pagina
				lineePagina = 0; // il numero di linee e' reinizializzato a zero
				altezzaCorrente = 0; // altezza corrente del testo stampato nella pagina
				oggettoGrafico.dispose(); // rilascia risorse legate al precedente oggettoGrafico
				oggettoGrafico = jobStampa.getGraphics(); // preleva il successivo oggettoGrafico
				if (oggettoGrafico != null) // se l'oggettoGrafico non e' nullo
				{
					oggettoGrafico.setFont(carattereHelvetica); // imposta questo carattere a oggettoGrafico
				}
			}
			altezzaCorrente += altezzaFont; // altezzaCorrente della pagina e' incrementato dall'altezza del Font attuale
			if (oggettoGrafico != null) // se l'oggettoGrafico non e' null allora
			{
				// scrive il testo incluso in lineaSuccessiva utilizzando l'attuale font e colore
				// la base del carattere piu' a sinistra ha coordinate x = 0 e y = altezzaCorrente - descentFont
				oggettoGrafico.drawString(lineaSuccessiva, 0, altezzaCorrente - descentFont);
				lineePagina++; // e' stata appena stampata una linea della pagina
				lineeComplessive++; // e' stata appena stampata una linea del documento
			}
			else
			{
				System.out.println ("oggettoGrafico e' vuoto: errore di stampa.");
			}
			}
		} while (lineaSuccessiva != null);
	} catch (EOFException eof)
	{
		System.out.println ("Errore nella lettura del documento da stampare.");
	}
	catch (Throwable e)
	{
			System.out.println ("Eccezione rilevata durante la stampa del documento: " + e);
			e.printStackTrace();
	}
	System.out.println ("" + lineePagina + " linee stampate nella pagina " + numeroPagina);
	System.out.println ("Numero complessivo di linee stampate: " + lineeComplessive);
	System.out.println ("Numero di pagine stampate: " + numeroPagina);
	}
}
```

Listato StoriaFile.java

```
/* ********************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · StoriaFile.java - Definizione classe per elencare i file aperti di recente
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  ******************************************************************************** */


// **** importa i packages utili ****
import java.io.*;


// **** definizione classe per memorizzare i cammini degli ultimi file aperti ****

public class StoriaFile
{
	//File filestoria = new File("risorse/XMLStudio.his");

	String primo = null, // path relativo all'ultimo file aperto nell'editor
	      secondo = null,
	      terzo = null,
	      quarto = null;

	// **** costruttore della classe StoriaFile ****

	public StoriaFile()
	{
		String linea; // linea del file letta

		try
		{
			BufferedReader in = new BufferedReader(new FileReader("./risorse/XMLStudio.his"));

			if ((primo = in.readLine()) != null) // carica i path relativi agli ultimi 4 file aperti nell'editor
			{
				if ((secondo = in.readLine()) != null)
				{
					if ((terzo = in.readLine()) != null)
					{
						if ((quarto = in.readLine()) != null);
					}
				}
			}

			/*
			System.out.println("Elenco file recentemente aperti");
			System.out.println("Primo: " + primo);
			System.out.println("Secondo: " + secondo);
			System.out.println("Terzo: " + terzo);
			System.out.println("Quarto: " + quarto);
			System.out.println("--------------------------------");*/
```

```
		}
		catch (Exception e)
		{
			System.err.println("Lettura del file XMLStudio.his fallita.");
		}
}


// **** metodo per salvare il path degli ultimi 4 file aperti ****

public void salvaStoriaFile(File ultimofile)
{
		boolean uguali;
		String temporaneo;

		if ((primo != null) && (primo.equals(ultimofile.getAbsolutePath()))) // il file appena aperto corrisponde il path di primo
		{
			uguali = true; // non e' necessario nessun aggiornamento dei path nel file
		}
		else
			if ((secondo != null) && (secondo.equals(ultimofile.getAbsolutePath()))) // l'ultimo file aperto corrisponde a secondo
			{
				temporaneo = primo; // realizza gli opportuni scambi per spostare secondo in cima della lista
				primo = secondo;
				secondo = temporaneo;
				uguali = false;
			}
			else
				if ((terzo != null) && (terzo.equals(ultimofile.getAbsolutePath()))) // l'ultimo file aperto corrisponde a terzo
				{
					temporaneo = secondo; // realizza scambi per spostare terzo in cima alla lista
					secondo = terzo;
					terzo = temporaneo;
					temporaneo = primo;
					primo = secondo;
					secondo = temporaneo;
					uguali = false;
				}
				else
					if ((quarto != null) && (quarto.equals(ultimofile.getAbsolutePath()))) // l'ultimo file aperto corrisponde a quar
					{
						temporaneo = terzo; // realizza scambi per spostare quarto in cima alla lista
						terzo = quarto;
						quarto = temporaneo;
						temporaneo = secondo;
						secondo = terzo;
						terzo = temporaneo;
						temporaneo = primo;
						primo = secondo;
						secondo = temporaneo;
						uguali = false;
					}
					else // l'ultimo file aperto non e' presente nell'elenco di file
					{
						//if ((primo != null) && (secondo != null) && (terzo != null) && (quarto != null))
						//{
							quarto = terzo; // realizza scambi per spostare il path di ultimo file in cima alla seque
							terzo = secondo;
							secondo = primo;
							primo = ultimofile.getAbsolutePath();
```

```
                                        //}
                                        uguali = false;
                                }

                /*System.out.println("Elenco file recentemente salvati");
                System.out.println("Salvato primo: " + primo);
                System.out.println("Salvato secondo: " + secondo);
                System.out.println("Salvato terzo: " + terzo);
                System.out.println("Salvato quarto: " + quarto);
                System.out.println("--------------------------------");*/

                if (uguali == false) // e' necessario salvare su file le modifiche effettuato
                {
                        try
                        {
                                FileWriter fw = new FileWriter("./risorse/XMLStudio.his");
                                BufferedWriter bw = new BufferedWriter(fw);
                                PrintWriter out = new PrintWriter(bw);

                                if (primo != null)
                                        out.println(primo); // primo contiene il cammino dell'ultimo file aperto
                                if (secondo != null)
                                        out.println(secondo);
                                if (terzo != null)
                                        out.println(terzo);
                                if (quarto != null)
                                        out.println(quarto);

                                out.flush();  // scrive gli ultimi dati rimasti nel buffer
                                out.close();
                                bw.close();
                                fw.close();
                        }
                        catch (Exception e)
                        {
                                System.err.println("Scrittura del file XMLStudio.his fallita." + e);
                        }
                }
        }
}
```

Listato TavolaAttributi.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · TavolaAttributi.java - Definizione classe per gestire attributi Xml
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import javax.swing.table.*;
//import javax.xml.parsers.*;
import org.w3c.dom.*;

import javax.swing.tree.*;
import javax.swing.event.*;
import org.w3c.dom.*;
import javax.swing.JTree;


/* Questa classe astratta fornisce un'implementazione di default per la maggior parte dei metodi
   nell'interfaccia TableModel in javax.swing.table. Si occupa di gestire i listener e fornisce alcuni strumenti
   per generare TableModelEvents e inviarli agli opportuni listeners. Per creare un concreto TableModel come
   sottoclasse di AbstractTableModel e' necessario implementare i 3 seguenti metodi:

   · public int getRowCount();
   · public int getColumnCount();
   · public Object getValueAt(int row, int column); */

// **** definizione classe per realizzare la gestione degli attributi nell'albero xml di parsing ****

class TavolaAttributi extends AbstractTableModel
{
        // l'interfaccia Node e' il principale tipo di dati per il documento DOM
        protected Node nodoCorrente; // istanza di nodo i cui attributi sono rappresentati nel modello.
        protected NamedNodeMap attributiCorrenti; /* oggetto che implementa l'interfaccia NamedNodeMap per rappresentare
                                                      collezioni di nodi accessibili con il nome - questa istanza rappresenta
                                                      tutti gli attributi dei nodi */

        // **** dato un nodo, in attributiCorrenti mette i suoi attributi ****

        public void assegnaNodo(Node nodo)
        {
                nodoCorrente = nodo;

                // attributiCorrenti = (node == null ? null : node.getAttributes());
                if (nodo == null)
                        attributiCorrenti = null; // se il nodo e' null allora non conterra' attributi
                else
                        attributiCorrenti = nodo.getAttributes(); // da nodo preleva i suoi attributi e li mette in un tipo NamedNodeMap
        }
```

```
// **** restituisce il nodo esaminato ****

public Node restituisciNodoCorrente()
{
	return nodoCorrente;
}


/* **** restituisce il numero di righe nel modello; una JTable usa questo metodo per determinare quante righe
	dovrebbe visualizzare; ritorna quindi il numero di attributi presenti in attributiCorrenti **** */

public int getRowCount()
{
	if (attributiCorrenti == null)
		return 0;
	return attributiCorrenti.getLength();
}


/* restituisce il numero di colonne del modello corrente; ua JTable lo usa per determinare il numero
  di colonne che dovrebbe creare e visualizzare per default */

public int getColumnCount()
{
	return 2;
}


/* **** dato un determinato indice di colonna restituisce il nome della colonna **** */

public String getColumnName(int numeroColonna)
{
	return numeroColonna == 0 ? "Attributo" : "Valore"; // la colonna 0 ha nome "Attributo" e la colonna 1 ha nome "Valore"
}


// **** restituisce il valore relativo ad una determinata cella identificata da un indice colonna e riga ****

public Object getValueAt(int numeroRiga, int numeroColonna)
{
	if (attributiCorrenti == null || numeroRiga < 0 || numeroRiga >= getRowCount())
		return "";
	Attr attributi = (Attr)attributiCorrenti.item(numeroRiga); // interfaccia attr che contiene gli attributi
	if (attributi == null) // se non vi sono attributi, restituisci un oggetto vuoto
		return "";
	switch (numeroColonna) // se vi sono attributi allora considera il numero della colonna
	{
		case 0: // se si desidera la colonna 0, restituisci il nome dell'attributo
			return attributi.getName();
		case 1: // se si desidera la colonna 1, restituisci il valore dell'attributo
			return attributi.getValue();
	}
	return "";
}


// **** restituisce vero se la cella identificata da una riga e da una colonna e' editabile ****
```

```
	public boolean isCellEditable(int numeroRiga, int numeroColonna)
	{
		return (numeroColonna == 1 ); // se il numero di colonna e' relativo al valore, allora la cella della JTable e' editabile
	}


	// **** metodo da utilizzare per l'inserimento manuale dei valori nella tavola JTable ****

	public void setValueAt(Object valoreAttributo, int numeroRiga, int numeroColonna)
	{
		if (numeroRiga < 0 || numeroRiga >= getRowCount()) // se numeroRiga non e' un valore di JTable valido
			return;
		if (nodoCorrente instanceof Element) // se nodoCorrente e' un nodo di tipo Element
		{
			String nome = getValueAt(numeroRiga, 0).toString(); // restituisce il nome dell'attributo da modificare
			((Element)nodoCorrente).setAttribute(nome, valoreAttributo.toString()); // reimposta il nome dell'attributo esistente associando un
		}
	}
}
```

Listato UndoableTextArea.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · UndoableTextArea.java - Estensione JTextArea per undo-redo
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


/* Tutte le classi interessate ad implementare le operazioni di tipo 'undo' (annulla) dovrebbero
   implementare le operazioni UndoableEditListener interface dal package javax.swing.event.
   La classe implementa un metodo pubblico void undoableEditHappened(UndoableEditEvent e), richiamato
   per notificare ai listerer che l'edit 'cancellabile', 'annullabile' (undo) è avvenuto. L'azione che
   viene svolta da questo metodo e' notificare il testo editato dall'UndoManager. L'UndoManager colleziona
   tutte le azioni di editing avvenute.

   · Il numero massimo di azioni di editing può essere impostato richiamando il metodo setLimit(int)
   della classe UndoManager: imposta il numero massimo di azioni di editing che l'UndoManager può contenere.

   · Per annullare un cambiamento e' sufficiente richiamare il metodo undo() di UndoManager.

   · Per ripristinare un cambiamento e' sufficiente richiamare il metodo redo() di UndoManager.

   Se UndoManager non puo' realizzare una operazione di undo, verra' generata un'eccezione CannotUndoException.
   Similarmente se UndoManager non puo' realizzare una operazione di ripristino (redo), verra' generata un'eccezione
   CannotRedoException. */


// **** importa i packages utili ****

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.undo.*; // package specifico che implementa le funzioni di undo-redo
import javax.swing.text.JTextComponent;



/* **** definizione classe per definire la JTextArea ****

   · Per realizzare le operazioni di undo-redo e' importante implementare la JTextArea con UndoableEditListener

   · Per utilizzare i metodi KeyPressed, KeyReleased, KeyTyped e' necessario implementare la JTextArea con KeyListener

   · Per utilizzare i metodi FocusGained e FocusLost e' necessario implementare la JTextArea con FocusListener.
     Questi metodi sono invocati quando un componente guadagna o perde il focus della tastiera.*/

class UndoableTextArea extends JTextArea implements UndoableEditListener, FocusListener//, KeyListener //, FocusListener
{
        public static final int limiteundo = 100; // imposta come limite di 'undo', 100 caratteri
```

```
public UndoManager oggettoUndoManager; // definizione dell'oggetto UndoManager

public Color coloreTestoEvidenziato;


// **** costruttore della classe UndoableTextArea ****

public UndoableTextArea()
{
	//super("testo"); // inserisce 'testo' all'interno dell'editor

	creaUndoManager(); // crea la TextArea dodata di proprieta' undo-redo
	addFocusListener(this); // per l'intercettazione di eventi Focus e il richiamo dei metodi focusGained e focusLost
}


// **** metodo che crea l'UndoManager ****

public void creaUndoManager()
{
	// aggiunge l'UndoableEditListener alla TextArea
	getDocument().addUndoableEditListener(this);

	// aggiunge il KeyListener necessario per keyPressed
	// this.addKeyListener(this);

	// aggiunge il focus listener per ricevere un evento di focus da questo
	// componente (JTextArea) quando acquisisce il focus relativo all'input
	// this.addFocusListener(this);

	oggettoUndoManager = new UndoManager();
	oggettoUndoManager.setLimit(limiteundo);
}


// **** metodo che rimuove l'UndoManager ****

public void rimuoviUndoManager()
{
	oggettoUndoManager.end(); // rimuove l'UndoManager
}


// **** metodo che rimuove i dati presenti nell'UndoManager ****

public void cancelladatiUndoManager()
{
	oggettoUndoManager.discardAllEdits(); // rimuove l'UndoManager
}


// **** undoableEditHappened e' richiamato non appena avviene una azione di edit su JTextArea ****

public void undoableEditHappened(UndoableEditEvent e)
{
	oggettoUndoManager.addEdit(e.getEdit()); // aggiunge l'azione di editing all'UndoManager
}


// **** metodo che realizza l'operazione di undo-annulla ****
```

```
public void annulla()
{
	try
	{
		oggettoUndoManager.undo(); // annulla l'ultimo cambiamento
	}
	catch(CannotUndoException cue)
	{
		Toolkit.getDefaultToolkit().beep();
	}
}


// **** metodo che realizza l'operazione di ripristina-redo ****

public void ripristina()
{
	try
	{
		oggettoUndoManager.redo(); // ripristina l'ultimo cambiamento
	}
	catch(CannotRedoException cue)
	{
		Toolkit.getDefaultToolkit().beep();
	}
}


// **** metodo invocato quando e' premuto un tasto ****

/*public void keyPressed(KeyEvent e)
{
	//identify the key pressed is the combination of
	//Controlkey and Z key
	if((e.getKeyCode() == KeyEvent.VK_Z) && (e.isControlDown()))
	{
		annulla();
	}

	//identify the key pressed is the combination of
	//Controlkey and Y key
	if((e.getKeyCode() == KeyEvent.VK_R) && (e.isControlDown()))
	{
		ripristina();
	}
}*/


/*// **** metodo invocato quando un tasto viene rilasciato ****

public void keyReleased(KeyEvent e)
{
}


// **** metodo invocato quando un tasto e' premuto e poi rilasciato ****

public void keyTyped(KeyEvent e)
{
```

```
        }*/


        // **** metodo invocato quando la JTextArea guadagna il focus della tastiera **** ooo

        public void focusGained(FocusEvent fe)
        {
        }


        // **** metodo invocato quando la JTextArea perde il focus della tastiera **** ooo

        public void focusLost(FocusEvent fe)
        {
                this.getCaret().setSelectionVisible(true);
        }
}
```

Listato XmlNodo.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · XmlNodo.java - Definizione classe che estende DefaultMutableTreeNode
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  *************************************************************************** */


// **** importa i packages utili ****

import javax.swing.tree.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;


// **** definizione classe per rappresentare i nodi Xml nel nostro viewer ****

/* questa classe estende DefaultMutableTreeNode per rappresentare i nodi Xml nel viewer
   che mostra l'albero del documento Xml parsificato. La principale personalizzazione
   della classe e' data dalla ridefinizione del metodo toString che restituisce una
   rappresentazione testuale del nodo in base al suo tipo elemento o testuale */

class XmlNodo extends DefaultMutableTreeNode
{
	// **** costruttore della classe XmlNodo ****

	public XmlNodo(Node node)
	{
		super(node); // l'oggetto node ha tutte le caratteristiche dei nodi DefaultMutableTreeNode
	}

	// **** restituisce da un oggetto XmlNodo il nodo di tipo Node ****

	public Node restituisciXmlNodo()
	{
		Object oggetto = getUserObject(); // restituisce l'oggetto Node memorizzato in questo nodo XmlNode
		if (oggetto instanceof Node)
			return (Node) oggetto; // casting di oggetto in Node
		return null; // altrimenti non restituisce nulla
	}

	// **** restituisce la stringa che individua il contenuto del nodo ****

	public String toString()
	{
		Node nodo = restituisciXmlNodo();
		if (nodo == null) // se l'oggetto e' vuoto
			return getUserObject().toString();
		StringBuffer sb = new StringBuffer();
		switch (nodo.getNodeType())
		{
```

```
                        case Node.ELEMENT_NODE: // se il nodo e' un elemento includilo tra parentesi graffe
                                sb.append('<');
                                sb.append(nodo.getNodeName());
                                sb.append('>');
                                break;
                        case Node.TEXT_NODE: // se il nodo e' testuale restituisci il suo valore
                                sb.append(nodo.getNodeValue());
                                break;
                }
                return sb.toString(); // restituisce la stringa che individua il contenuto del nodo
        }


        // **** aggiunge il nodo figlio al nodo padre ****

        public void aggiungiXmlNodo(XmlNodo nodoFiglio)
        throws Exception
        {
                Node nodoPadre = restituisciXmlNodo(); // restituisce l'oggetto di tipo node selezionato
                if (nodoPadre == null) // il nodo padre non dev'essere nullo
                        throw new Exception("Nodo xml corrotto");
                nodoPadre.appendChild(nodoFiglio.restituisciXmlNodo()); // appende il nodo figlio al padre
                add(nodoFiglio);
        }


        // **** rimuove un determinato nodo ****

        public void rimuovi() throws Exception
        {
                Node nodoAttuale = restituisciXmlNodo(); // restituisce l'oggetto di tipo node selezionato
                if (nodoAttuale == null) // il nodo non dev'essere nullo
                        throw new Exception("Nodo xml corrotto nella rimozione");
                Node nodoPadre = nodoAttuale.getParentNode(); // individua il nodo genitore del nodo attualmente selezionato
                if (nodoPadre == null) // se nodoPadre e' null allora si sta cercando di rimuovere la radice
                        throw new Exception("Non e' possibile rimuovere il nodo radice");
                TreeNode nodoGenitore = getParent(); // restituisce il nodo padre del nodo selezionato
                if (!(nodoGenitore instanceof DefaultMutableTreeNode))
                        throw new Exception("Non e' possibile rimuovere il nodo dell'albero");
                nodoPadre.removeChild(nodoAttuale);
                ((DefaultMutableTreeNode)nodoGenitore).remove(this);
        }

        // **** modifica il tag di un determinato nodo ****

//      public void cambia() throws Exception
//      {
//              Node nodoAttuale = restituisciXmlNodo(); // restituisce l'oggetto di tipo node selezionato
//              if (nodoAttuale == null) // il nodo non dev'essere nullo
//                      throw new Exception("Nodo xml corrotto nella rimozione");
//              //nodoAttuale.setPrefix("nuovoTagNodo");
//              //nodoAttuale.setNodeValue("nuovoTagNodo");
//              System.out.println("sono qui:3 " + nodoAttuale.getNodeName() + "/" + nodoAttuale.getNodeValue());
//      }
}
```

Listato XMLRoutine.java

```
/* ****************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · XMLRoutine.java - Definizione classe per la scrittura del file xml
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

  **************************************************************************** */


// **** importa i packages utili ****

import java.io.*;
import org.w3c.dom.*;


// **** classe per la scrittura del file xml ****

class XMLRoutine
{
	static int spaziTab = 0;

	// **** dato il documento xml in formato document lo scrive su file attraverso la classe Writer ****

	public static void writeDocumento(Document documento, Writer out) throws Exception
	{
		out.write("<?xml version=\"1.0\" encoding=\"iso-8859-1\" standalone=\"yes\"?>\n");
		write(documento.getDocumentElement(), out); // getoDocumentElement restituisce diretto accesso ai nodi che sono figli della radice
	}

	// **** dato il documento xml in formato document lo scrive su file attraverso la classe Writer ****

	public static void write(Document documento, Writer out) throws Exception
	{
		write(documento.getDocumentElement(), out); // getoDocumentElement restituisce diretto accesso ai nodi che sono figli della radice
	}

	// **** scrive un singolo nodo su file ****

	public static void write(Node nodo, Writer out) throws Exception
	{
		if (nodo == null || out == null)
			return;

		int tipo = nodo.getNodeType(); // restituisce il tipo del nodo in esame
		switch (tipo)
		{
			case Node.DOCUMENT_NODE: // il nodo identifica a sua volta un documento xml
				write(((Document)nodo).getDocumentElement(), out); // richiama ricorsivamente il metodo write
				out.flush();
				break;

			case Node.ELEMENT_NODE: // il nodo e' un elemento e puo' avere uno o piu' attributi
```

```
                for (int i = 0; i<spaziTab; i++)
                        out.write('\t');
                spaziTab =  spaziTab + 1;

                out.write('<');
                out.write(nodo.getNodeName()); // scrive il tag del nodo
                NamedNodeMap attributi = nodo.getAttributes();
                for (int k = 0; k < attributi.getLength(); k++ ) // nell'eventualita' che il nodo abbia attributi
                {
                        Node attributo = attributi.item(k);
                        out.write(' ');
                        out.write(attributo.getNodeName()); // scrive il nome dell'attributo
                        out.write("=\"");
                        out.write(attributo.getNodeValue());
                        out.write('\"');
                }
                out.write('>');
                out.write('\n');
                break;

        case Node.ENTITY_REFERENCE_NODE: // il nodo e' di tipo entità reference
                out.write('&');
                out.write(nodo.getNodeName());
                out.write(';');
                break;

        case Node.CDATA_SECTION_NODE: // Il nodo rappresenta una sezione CDATA
                out.write("<![CDATA[");
                out.write(nodo.getNodeValue());
                out.write("]]>");
                break;

        case Node.TEXT_NODE: // il nodo e' testuale

                for (int i = 0; i<spaziTab; i++)
                        out.write('\t');

                out.write(nodo.getNodeValue());

                out.write('\n');
                break;

        case Node.PROCESSING_INSTRUCTION_NODE: // il nodo ha informazioni specifiche al processore
                out.write("<?");
                out.write(nodo.getNodeName());
                String data = nodo.getNodeValue();
                if ( data != null && data.length() > 0 )
                {
                        out.write(' ');
                        out.write(data);
                }
                out.write("?>");
                break;

        default:
                out.write("<TYPE=" + tipo);
                out.write(nodo.getNodeName());
                out.write("?>");
                break;
```

```
                }

                NodeList figli = nodo.getChildNodes(); // in figli vengono posti i nodi figli del nodo
                if (figli != null) // se vi sono dei nodi figli, richiama ricorsivamente il metodo write
                {
                        for ( int k = 0; k < figli.getLength(); k++ ) // per tutti i nodi figli
                        {
                                write(figli.item(k), out); // richiama il metodo write passando il k-esimo nodo figlio
                        }
                }

                if (nodo.getNodeType() == Node.ELEMENT_NODE) // il nodo e' di tipo elemento e lo scrive su disco
                {
                        spaziTab =  spaziTab - 1;
                        for (int i = 0; i<spaziTab; i++)
                                out.write('\t');

                        out.write("</");
                        out.write(nodo.getNodeName());
                        out.write('>');

                        out.write('\n');
                }
                out.flush();
        }
}
```

Listato XMLStudio.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · XMLStudio.java - Programma principale
    (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

   *  ================================================= *
   *  This program contains code from the book "Swing"    *
   *  2nd Edition by Matthew Robinson and Pavel Vorobiev  *
   *  http://www.spindoczine.com/sbe                      *
   *  ================================================= *

   This program is free software; you can redistribute it and/or
   modify it under the terms of the GNU General Public License
   as published by the Free Software Foundation; either version 2
   of the License, or (at your option) any later version.

   This program is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with this program; if not, write to the Free Software
   Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

   http://www.gnu.org/copyleft/gpl.html

  *************************************************************************** */


// **** importa i packages utili ****
import java.io.*; // classi per la gestione dell'input-output
import java.awt.*; // libreria grafica AWT
import java.awt.event.*; // classi di eventi legati alle librerie AWT
import java.awt.Cursor; // per modificare il puntatore del mouse
import javax.swing.*; // libreria grafica Swing
import javax.swing.event.*; // classi di eventi legati alle librerie Swing
import javax.swing.tree.*; // classi per l'uso degli alberi in ambiente Swing
import java.util.Properties; // classi per la definizione di oggetto proprieta' per le impostazioni predefinite
import javax.help.*; // classi che consentono l'uso di JavaHelp
import java.net.URL; // classi che definiscono gli oggetti di tipo URL
import javax.xml.parsers.*; // classi usate per processare documenti Xml
import org.w3c.dom.*; // serie di interfacce che definiscono il DOM (Document Object Model)
import org.xml.sax.SAXException;  // eccezione utilizzata per la verifica di documento xml ben formato


// **** definizione classe ****

public class XMLStudio extends JFrame implements ActionListener, WindowListener, CaretListener
{
   // dichiarazioni generali
```

```
private Dimension screenSize; // oggetto contenente le dimensioni dello schermo
public JLabel stato; // riporta lo stato corrente sulla StatusBar
private JLabel colonnerighe; // riporta l'attuale riga e colonna del cursore nell'editor

// impostazioni oggetti proprietà dell'editor
private Font carattereDefault; // oggetto contenente il Font di default impostato nelle proprieta'
private Color coloreTestoDefault; // oggetto contenente il colore di default del testo
private Color coloreSfondoDefault; // oggetto contenente il colore di default dello sfondo del testo
private Boolean wordWrapDefault; // variabile globale che determina se ad areaTesto e' applicato o meno il word-wrap
private String directoryCorrente; // oggetto stringa contenente la directory attuale di lavoro
private String fileXmlTagCorrente; // oggetto stringa contenente il nome del file dei tag da caricare per default all'avvio
private String finestraX, finestraY, divisoreOrizzontale, divisoreVerticale; // dimensioni del JFrame dell'editor
private String lookandfeelCorrente; // oggetto stringa contenente il lookandfeel attualmente visualizzato
private String memorizzaLinea = null; // memorizza il numero di riga del cursore sul documento
private int riga; // memorizza il valore relativo alla riga raggiunto dal cursore
private File fileCorrente = new File("Vuoto"); // file salvato e attualmente aperto
private boolean fileGiaSalvato; // vero se fileCorrente e' gia' stato salvato
private String parola; // stringa ricercata in Trova, Trova successivo, Sostituisci
private StoriaFile ultimiFile; // memorizza in questo oggetto i cammini degli ultimi file aperti
HelpSet hs; // definisce l'oggetto HelpSet per l'uso di JavaHelp
HelpBroker hb; // definisce l'oggetto HelpBroker per l'uso di JavaHelp
public UndoableTextArea areaTesto; // JTextArea areaTesto che contiene il documento testuale;
public JPopupMenu popupAreaTesto; // implementa i menu' popup relativi all'area testo
private JSplitPane splitPaneCentrale, splitPaneCentrale2; // contenitori dei JTree, JTable, ecc.
private XmlTag oggettoXmlTag; // oggetto contenente il JTree e JTable relativi agli attributi
protected Document documentoXml; // interfaccia che rappresenta l'intero documento Xml
protected JTree alberoXml; // definisce il JTree che permette la visualizzazione dei dati secondo gerarchia
protected DefaultTreeModel modelloAlbero; // implementa l'interfaccia TreeModel
protected JTable tavolaAttributiXml; // definisce la JTable per gli attributi dell'albero di parsing
protected TavolaAttributi modellotavolaAttributiXml; // definisce un'estensione di AbstractTableModel per implementare la JTable

JButton nuovo, apri, salva, stampa, taglia, copia, incolla, annulla, ripristina, aggiornaxml, benformatoxml, // pulsanti della toolbar
    aggiungitagxml, memolinea, vaimemolinea, creaxml, trova, trovasuccessivo, sostituisci, helpcontesto;

public JMenuItem jmi_salva; // per l'abilitazione/disabilitazione dell'icona relativa al salvataggio del file
public JMenuItem jmi_aggiunginodotag, jmi_modificanodotag, jmi_eliminanodotag, // menu' pubblici in quanto sono abilitati/disabilitati da XmlTag
                                        jmi_aggiungiattributonodotag, jmi_modificaattributonodotag, jmi_eliminaattributonodotag;

JCheckBoxMenuItem jmi_wordwrap; // per la selezione dell'opzione "A capo linea" nell'area testo

        // **** inizializza il look e feel dell'editor ****

        private void initLookAndFeel(String look)
        {
          String lookAndFeel = null; // definizione oggetto per l'impostazione del look e feel

          if (look != null)
          {
                if (look.equals("Metal")) // stile 'Metal' di look e feel
                {
                        lookAndFeel = UIManager.getCrossPlatformLookAndFeelClassName();
                }
                else
                if (look.equals("System")) // stile 'System' di look e feel
                {
                        lookAndFeel = UIManager.getSystemLookAndFeelClassName();
                }
                else
                if (look.equals("Mac")) // stile 'Mac' di look e feel
                {
```

```
                        lookAndFeel = "com.sun.java.swing.plaf.mac.MacLookAndFeel";
                    }
                    else
                    if (look.equals("Windows")) // stile 'Windows' di look e feel
                    {
                        lookAndFeel = "com.sun.java.swing.plaf.windows.WindowsLookAndFeel";
                    }
                    else
                    if (look.equals("Motif")) // stile 'Motif' di look e feel
                    {
                        lookAndFeel = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
                    }

                try
                {
                    UIManager.setLookAndFeel(lookAndFeel); // imposta il look e feel dell'interfaccia grafica

                    //SwingUtilities.updateComponentTreeUI(contenitoreComponenti); //servirebbe ad aggiornare a run-time il lookefeel ma necessita di Frame
                    SwingUtilities.updateComponentTreeUI(getContentPane());
                }
                catch (ClassNotFoundException e)
                {
                  System.err.println("Non trovo la classe relativa al look e feel specificato: " + lookAndFeel);
                  System.err.println("Uso del look e feel di default in questa sessione.");
                }
                catch (UnsupportedLookAndFeelException e)
                {
                  System.err.println("Non e' possibile usare il look e feel specificato (" + lookAndFeel + ") su questo computer.");
                  System.err.println("Uso del look e feel di default in questa sessione.");
                }
                catch (Exception e)
                {
                  System.err.println("Il look e feel specificato (" + lookAndFeel + "), non e' disponibile.");
                  System.err.println("Uso del look e feel di default in questa sessione.");
                  e.printStackTrace();
                }
        }
}


 // **** creaJavaHelp definisce gli oggetti per usare l'HelpSet ****

        private void creaJavaHelp()
        {
                // il codice seguente mostra una tipica istanzazione dell'HelpSet

                ClassLoader loader = XMLStudio.class.getClassLoader(); // usata per identificare classi richieste da chi consulta l'HelpSet
                try
                {
                URL url = HelpSet.findHelpSet(loader, "./help/XMLStudio.hs"); // url del principale file di HelpSet
                hs = new HelpSet(loader, url); // per creare un'istanza di HelpSet e' necessario fornire 2 parametri

                // codice per definire l'interfaccia HelpBroker che descrive un componente usato per gestire l'iterazione con HelpSet

                hb = hs.createHelpBroker(); // crea l'oggetto HelpBroker
                hb.enableHelpKey(getRootPane(), "top", hs); // specifichiamo il valore ID dell'argomento di apertura dell'help
                }
                catch (Exception ex)
                {
                        ex.printStackTrace();
```

```
                }
        }


        // **** imposta le proprieta' di default nell'editor ****

        private void eseguiImpostazioniDefault(Properties p)
        {
                // carica le impostazioni di default in variabili

                carattereDefault = new Font(p.getProperty("font"),Integer.parseInt(p.getProperty("tipofont")),Integer.parseInt(p.getProperty("dimensionefont"))
                coloreTestoDefault = new Color(Integer.parseInt(p.getProperty("colorefont")));
                coloreSfondoDefault = new Color(Integer.parseInt(p.getProperty("coloresfondo")));

                finestraX = p.getProperty("finestrax");
                finestraY = p.getProperty("finestray");
                setBounds(0, 0, Integer.parseInt(finestraX), Integer.parseInt(finestraY)); // inizializzazione dimensione dell'editor

                divisoreOrizzontale = p.getProperty("divisorex");
                divisoreVerticale = p.getProperty("divisorey");

                directoryCorrente = p.getProperty("directory");
                File verifica = new File(directoryCorrente); // verifica se la directory correntemente salvata nelle impostazioni esiste realmente
                if (verifica.isDirectory() == false) // imposta come directory corrente la directory del programma
                {
                        System.out.println("- Si consiglia di impostare la Directory di lavoro relativa ai documenti testuali sui quali lavorare, nel menu' Impost
                        directoryCorrente = ".";
                }

                fileXmlTagCorrente = p.getProperty("filexmltag");
                verifica = new File(fileXmlTagCorrente); // verifica se il file di tag xml salvato nelle impostazioni esiste realmente
                if (verifica.isFile() == false) // imposta nessun file come file di tag xml
                {
                        System.out.println("- Si consiglia di impostare il File tag di default (.xml) precaricato all'avvio di XMLStudio, nel menu' Impostazioni\n'
                        fileXmlTagCorrente = null;
                }

                lookandfeelCorrente = p.getProperty("look");

                String wordWrapTemp = p.getProperty("wordwrap");

                if (wordWrapTemp.equals("true"))
                        wordWrapDefault = new Boolean("TRUE");
                else
                        wordWrapDefault = new Boolean("FALSE");
        }


// **** carica le impostazioni di default dell'editor ****

private void carica_ImpostazioniDefault()
{
        try
        {
                InputStream is = new FileInputStream("./risorse/XMLStudio.default"); // creazione di oggetto is per la lettura
                BufferedInputStream bis = new BufferedInputStream(is); // creazione di oggetto bis per la lettura con buffer
                Properties proprieta_XMLStudioEditor = new Properties(); // creazione di oggetto proprieta' per l'editor
                proprieta_XMLStudioEditor.load(bis); // carica le proprieta' dall'input stream
                eseguiImpostazioniDefault(proprieta_XMLStudioEditor);
                bis.close();
```

```
	}

	catch(FileNotFoundException e)
	{
		System.err.println("File impostazioni default XMLStudio.default non trovato.");
	}

	catch(IOException e)
	{
		System.err.println("Lettura del file impostazioni default XMLStudio.default fallita.");
	}
}


	// **** salva le impostazioni di default ****

	private void salva_ImpostazioniDefault()
	{
		// salva tutte le preferenze in un oggetto di tipo Properties

		try
		{
			OutputStream os = new FileOutputStream("./risorse/XMLStudio.default");
			BufferedOutputStream bos = new BufferedOutputStream(os);
			Properties proprieta_XMLStudioEditor = new Properties();

			proprieta_XMLStudioEditor.setProperty("font", carattereDefault.getName());
			proprieta_XMLStudioEditor.setProperty("tipofont", "" + carattereDefault.getStyle());
			proprieta_XMLStudioEditor.setProperty("dimensionefont", "" + carattereDefault.getSize());
			proprieta_XMLStudioEditor.setProperty("colorefont", "" + coloreTestoDefault.getRGB());
			proprieta_XMLStudioEditor.setProperty("coloresfondo", "" + coloreSfondoDefault.getRGB());
			proprieta_XMLStudioEditor.setProperty("directory", directoryCorrente);
			proprieta_XMLStudioEditor.setProperty("filexmltag", fileXmlTagCorrente);
			proprieta_XMLStudioEditor.setProperty("look", lookandfeelCorrente);
			proprieta_XMLStudioEditor.setProperty("finestrax", finestraX);
			proprieta_XMLStudioEditor.setProperty("finestray", finestraY);
			proprieta_XMLStudioEditor.setProperty("divisorex", divisoreOrizzontale);
			proprieta_XMLStudioEditor.setProperty("divisorey", divisoreVerticale);
			proprieta_XMLStudioEditor.setProperty("wordwrap", wordWrapDefault.toString());

			proprieta_XMLStudioEditor.store(bos,"File proprieta' di default di XMLStudio."); // memorizza le proprieta' sul file

			bos.close();

			stato.setText("Aggiornamento del file XMLStudio.default avvenuto con successo.");
		}
		catch(Exception e)
		{
			System.err.println("Salvataggio del file XMLStudio.default fallito.");
			stato.setText("Salvataggio del file XMLStudio.default fallito.");
			return;
		}
	}


	// **** carica un file di impostazioni predefinito ****

	private void ripristinaImpostazioniDefault()
	{
		int n;
```

```
		byte buffer[] = new byte[256]; // crea un buffer costituito da 256 byte

		try
		{		// carica le impostazioni predefinite dal file XMLStudio.predefinito

				FileInputStream in = new FileInputStream("./risorse/XMLStudio.predefinito"); // creazione di oggetto in per la lettura
				FileOutputStream out = new FileOutputStream("./risorse/XMLStudio.default"); // creazione di oggetto out per la scrittura

				while ((n = in.read(buffer)) > -1) // legge un byte di dati dall'inputstream
					out.write(buffer, 0, n); // scrive n byte di dati sull'outputstream a partire dall'offset 0

				out.close(); // chiudi il file output stream e rilascia le risorse associate a questo stream
				in.close(); // chiudi il file input stream e rilascia le risorse associate a questo stream


				// riavvia l'editor per rendere effettive le nuove impostazioni

				azioneRiavviaEditor();
		}

		catch(FileNotFoundException e)
		{
				System.err.println("File impostazioni default XMLStudio.predefinito non trovato.");
		}

		catch(IOException e)
		{
				System.err.println("Lettura del file impostazioni default XMLStudio.predefinito fallita.");
		}
}


// **** crea i menu della finestra dell'editor ****

private JMenuBar crea_MenuBar()
{
		JMenuItem jmi_nuovo, jmi_apri, jmi_chiudi, jmi_salvacome, jmi_uscita, // definizione oggetti menu File
					jmi_stampawindows, jmi_stampalinux, jmi_stampaunix, jmi_stampa, // definizione oggetti per stampa menu File
					jmi_ultimo1, jmi_ultimo2, jmi_ultimo3, jmi_ultimo4, // definizione oggetti menu File relativi a ultimi file aperti
					jmi_memlinea, jmi_vaiamemlinea, // definizione oggetti menu
			jmi_annulla, jmi_ripristina, jmi_taglia, jmi_copia, jmi_incolla, jmi_selezionatutto, // definizione oggetti menu Modifica
			jmi_trova, jmi_trovatag, jmi_trovasuccessivo, jmi_sostituisci, jmi_linea, // definizione oggetti menu Ricerca
			jmi_creaxml, jmi_benformatoxml, jmi_creaalberoxml, jmi_aggiungitagxml, jmi_eliminatagxml, jmi_selezionatagxml, // definizione operazioni
			jmi_modificaxml, jmi_modificaxml1, jmi_modificaxml2, jmi_modificaxml3, jmi_modificaxml4, jmi_modificaxml5, // modifica documento xml
			jmi_ripristinacaratterispeciali, jmi_creadocumentovalido, jmi_creadtd, jmi_css,// definizione oggetti menu per modifiche al documento XMl
			jmi_carattere, jmi_coloretesto, jmi_coloresfondo, jmi_directory, jmi_filexmltag, jmi_dimensionefinestra, jmi_dimensionipredefinite, // defin
			jmi_nuovotag, jmi_apritag, jmi_salvatag, jmi_salvanometag, jmi_cancellaSelezioneJTable, // definizione oggetti menu Tag
			jmi_javahelp, jmi_javahelpcontesto, jmi_guida, jmi_info; // definizione oggetti menu Aiuto

		JMenuBar menuBar = new JMenuBar(); // definizione oggetto per la barra dei menu

		JRadioButtonMenuItem jmi_aspetto1, jmi_aspetto2, jmi_aspetto3, jmi_aspetto4, jmi_aspetto5;

  // definizione menu fileMenu

		JMenu fileMenu = new JMenu("File");

		jmi_nuovo = new JMenuItem("Nuovo");
		jmi_nuovo.addActionListener(this);
		jmi_nuovo.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N, ActionEvent.CTRL_MASK));
```

```
jmi_nuovo.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/nuovo.gif")));
fileMenu.add(jmi_nuovo);

jmi_apri = new JMenuItem("Apri..");
jmi_apri.addActionListener(this);
jmi_apri.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_A, ActionEvent.CTRL_MASK));
jmi_apri.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/apri.gif")));
fileMenu.add(jmi_apri);

jmi_chiudi = new JMenuItem("Chiudi");
jmi_chiudi.addActionListener(this);
jmi_chiudi.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D, ActionEvent.CTRL_MASK));
fileMenu.add(jmi_chiudi);

fileMenu.addSeparator();

jmi_salva = new JMenuItem("Salva");
jmi_salva.addActionListener(this);
jmi_salva.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S, ActionEvent.CTRL_MASK));
jmi_salva.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/salva.gif")));
fileMenu.add(jmi_salva);

jmi_salvacome = new JMenuItem("Salva come...");
jmi_salvacome.addActionListener(this);
fileMenu.add(jmi_salvacome);

fileMenu.addSeparator();

jmi_stampawindows = new JMenuItem("Stampa (Windows)");
jmi_stampawindows.addActionListener(this);
fileMenu.add(jmi_stampawindows);

jmi_stampalinux = new JMenuItem("Stampa (Linux)");
jmi_stampalinux.addActionListener(this);
fileMenu.add(jmi_stampalinux);

jmi_stampaunix = new JMenuItem("Stampa (Unix)");
jmi_stampaunix.addActionListener(this);
fileMenu.add(jmi_stampaunix);

jmi_stampa = new JMenuItem("Stampa (grafica)");
jmi_stampa.addActionListener(this);
jmi_stampa.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/stampa.gif")));
fileMenu.add(jmi_stampa);

fileMenu.addSeparator();

jmi_uscita = new JMenuItem("Uscita");
jmi_uscita.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, ActionEvent.ALT_MASK));
jmi_uscita.addActionListener(this);
fileMenu.add(jmi_uscita);

fileMenu.addSeparator();

if (ultimiFile.primo != null) // la condizione non e' messa in and altrimenti il programma va in crash se la riga del file e' nulla
{
        if ((new File(ultimiFile.primo)).exists()) // il file realtivo al path del cammino caricato e' esistente
        {
                jmi_ultimo1 = new JMenuItem(ultimiFile.primo);
                jmi_ultimo1.addActionListener(this);
```

```
                        fileMenu.add(jmi_ultimo1);
                }
                else
                        ultimiFile.primo = null; // ""
}

if (ultimiFile.secondo != null)
{
                if ((new File(ultimiFile.secondo)).exists()) // il file realtivo al path del cammino caricato e' esistente
                {
                        jmi_ultimo2 = new JMenuItem(ultimiFile.secondo);
                        jmi_ultimo2.addActionListener(this);
                        fileMenu.add(jmi_ultimo2);
                }
                else
                        ultimiFile.secondo = null;
}

if (ultimiFile.terzo != null)
{
                if ((new File(ultimiFile.terzo)).exists()) // il file realtivo al path del cammino caricato e' esistente
                {
                        jmi_ultimo3 = new JMenuItem(ultimiFile.terzo);
                        jmi_ultimo3.addActionListener(this);
                        fileMenu.add(jmi_ultimo3);
                }
                else
                        ultimiFile.terzo = null;
}

if (ultimiFile.quarto != null)
{
                if ((new File(ultimiFile.quarto)).exists()) // il file realtivo al path del cammino caricato e' esistente
                {
                        jmi_ultimo4 = new JMenuItem(ultimiFile.quarto);
                        jmi_ultimo4.addActionListener(this);
                        fileMenu.add(jmi_ultimo4);
                }
                else
                        ultimiFile.quarto = null;
}

// definizione menu modificaMenu

JMenu modificaMenu = new JMenu("Modifica");

jmi_annulla = new JMenuItem("Annulla");
jmi_annulla.addActionListener(this);
jmi_annulla.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Z, ActionEvent.CTRL_MASK));
jmi_annulla.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/annulla.gif")));
modificaMenu.add(jmi_annulla);

jmi_ripristina = new JMenuItem("Ripristina");
jmi_ripristina.addActionListener(this);
jmi_ripristina.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_R, ActionEvent.CTRL_MASK));
jmi_ripristina.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/ripristina.gif")));
modificaMenu.add(jmi_ripristina);

modificaMenu.addSeparator();
```

```
jmi_taglia = new JMenuItem("Taglia");
jmi_taglia.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X, ActionEvent.CTRL_MASK));
jmi_taglia.addActionListener(this);
jmi_taglia.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/taglia.gif")));
modificaMenu.add(jmi_taglia);

jmi_copia = new JMenuItem("Copia");
jmi_copia.addActionListener(this);
jmi_copia.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, ActionEvent.CTRL_MASK));
jmi_copia.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/copia.gif")));
modificaMenu.add(jmi_copia);

jmi_incolla = new JMenuItem("Incolla");
jmi_incolla.addActionListener(this);
jmi_incolla.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, ActionEvent.CTRL_MASK));
jmi_incolla.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/incolla.gif")));
modificaMenu.add(jmi_incolla);

modificaMenu.addSeparator();

jmi_selezionatutto = new JMenuItem("Seleziona tutto");
jmi_selezionatutto.addActionListener(this);
//jmi_selezionatutto.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, ActionEvent.CTRL_MASK));
modificaMenu.add(jmi_selezionatutto);

// definizione menu ricercaMenu

JMenu ricercaMenu = new JMenu("Ricerca");

jmi_trova = new JMenuItem("Trova");
jmi_trova.addActionListener(this);
jmi_trova.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_T, ActionEvent.CTRL_MASK));
jmi_trova.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/trova.gif")));
ricercaMenu.add(jmi_trova);

jmi_trovatag = new JMenuItem("Trova tag");
jmi_trovatag.addActionListener(this);
jmi_trovatag.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F, ActionEvent.CTRL_MASK));
jmi_trovatag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/trova.gif")));
ricercaMenu.add(jmi_trovatag);

jmi_trovasuccessivo = new JMenuItem("Trova successivo");
jmi_trovasuccessivo.addActionListener(this);
jmi_trovasuccessivo.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_U, ActionEvent.CTRL_MASK));
jmi_trovasuccessivo.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/trovas.gif")));
ricercaMenu.add(jmi_trovasuccessivo);

jmi_sostituisci = new JMenuItem("Sostituisci");
jmi_sostituisci.addActionListener(this);
jmi_sostituisci.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_D, ActionEvent.CTRL_MASK));
jmi_sostituisci.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/sostituisci.gif")));
ricercaMenu.add(jmi_sostituisci);

ricercaMenu.addSeparator();

jmi_linea = new JMenuItem("Vai a linea");
jmi_linea.addActionListener(this);
jmi_linea.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_L, ActionEvent.CTRL_MASK));
ricercaMenu.add(jmi_linea);
```

```
ricercaMenu.addSeparator();

jmi_memlinea = new JMenuItem("Memorizza numero linea");
jmi_memlinea.addActionListener(this);
jmi_memlinea.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_E, ActionEvent.CTRL_MASK));
jmi_memlinea.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/memlinea.gif")));
ricercaMenu.add(jmi_memlinea);

jmi_vaiamemlinea = new JMenuItem("Vai a numero linea");
jmi_vaiamemlinea.addActionListener(this);
jmi_vaiamemlinea.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_M, ActionEvent.CTRL_MASK));
jmi_vaiamemlinea.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/vailinea.gif")));
ricercaMenu.add(jmi_vaiamemlinea);

// definizione menu xmlMenu

JMenu xmlMenu = new JMenu("Xml");

jmi_creaxml = new JMenuItem("Crea documento xml");
jmi_creaxml.addActionListener(this);
jmi_creaxml.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_G, ActionEvent.CTRL_MASK));
jmi_creaxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
xmlMenu.add(jmi_creaxml);

jmi_benformatoxml = new JMenuItem("Verifica documento xml ben formato");
jmi_benformatoxml.addActionListener(this);
jmi_benformatoxml.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, ActionEvent.CTRL_MASK));
jmi_benformatoxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/benformato.gif")));
xmlMenu.add(jmi_benformatoxml);

jmi_creaalberoxml = new JMenuItem("Aggiorna albero xml");
jmi_creaalberoxml.addActionListener(this);
jmi_creaalberoxml.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_J, ActionEvent.CTRL_MASK));
jmi_creaalberoxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/aggiorna.gif")));
xmlMenu.add(jmi_creaalberoxml);

jmi_aggiungitagxml = new JMenuItem("Aggiungi tag xml");
jmi_aggiungitagxml.addActionListener(this);
jmi_aggiungitagxml.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_K, ActionEvent.CTRL_MASK));
jmi_aggiungitagxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
xmlMenu.add(jmi_aggiungitagxml);

jmi_selezionatagxml = new JMenuItem("Seleziona paragrafo tag xml");
jmi_selezionatagxml.addActionListener(this);
jmi_selezionatagxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/selblocco.gif")));
xmlMenu.add(jmi_selezionatagxml);

jmi_eliminatagxml = new JMenuItem("Elimina tag xml");
jmi_eliminatagxml.addActionListener(this);
jmi_eliminatagxml.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/canctag.gif")));
xmlMenu.add(jmi_eliminatagxml);

xmlMenu.addSeparator();

jmi_modificaxml = new JMenu("Modifica per documento ben formato");
jmi_modificaxml.addActionListener(this);
xmlMenu.add(jmi_modificaxml);

jmi_modificaxmll = new JMenuItem("Sostituisci &");
jmi_modificaxmll.addActionListener(this);
```

```
jmi_modificaxml.add(jmi_modificaxml1);

jmi_modificaxml2 = new JMenuItem("Sostituisci <");
jmi_modificaxml2.addActionListener(this);
jmi_modificaxml.add(jmi_modificaxml2);

jmi_modificaxml3 = new JMenuItem("Sostituisci >");
jmi_modificaxml3.addActionListener(this);
jmi_modificaxml.add(jmi_modificaxml3);

jmi_modificaxml4 = new JMenuItem("Sostituisci '");
jmi_modificaxml4.addActionListener(this);
jmi_modificaxml.add(jmi_modificaxml4);

jmi_modificaxml5 = new JMenuItem("Sostituisci \"");
jmi_modificaxml5.addActionListener(this);
jmi_modificaxml.add(jmi_modificaxml5);

xmlMenu.addSeparator();

jmi_ripristinacaratterispeciali = new JMenuItem("Ripristina caratteri speciali");
jmi_ripristinacaratterispeciali.addActionListener(this);
xmlMenu.add(jmi_ripristinacaratterispeciali);

xmlMenu.addSeparator();

jmi_creadtd = new JMenuItem("Crea dtd da documento xml");
jmi_creadtd.addActionListener(this);
xmlMenu.add(jmi_creadtd);

jmi_creadocumentovalido = new JMenuItem("Crea dtd e documento xml valido");
jmi_creadocumentovalido.addActionListener(this);
xmlMenu.add(jmi_creadocumentovalido);

jmi_css = new JMenuItem("Crea css associato al file xml");
jmi_css.addActionListener(this);
xmlMenu.add(jmi_css);

// definizione menu tagMenu

JMenu tagMenu = new JMenu("Tag");

jmi_nuovotag = new JMenuItem("Crea nuovo file tag (.xml)");
jmi_nuovotag.addActionListener(this);
jmi_nuovotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/nuovo.gif")));
tagMenu.add(jmi_nuovotag);

jmi_apritag = new JMenuItem("Apre file tag (.xml)");
jmi_apritag.addActionListener(this);
jmi_apritag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/apri.gif")));
tagMenu.add(jmi_apritag);

jmi_salvatag = new JMenuItem("Salva file tag (.xml)");
jmi_salvatag.addActionListener(this);
jmi_salvatag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/salva.gif")));
tagMenu.add(jmi_salvatag);

jmi_salvanometag = new JMenuItem("Salva nome file tag (.xml)");
jmi_salvanometag.addActionListener(this);
jmi_salvanometag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/salvanome.gif")));
```

```
tagMenu.add(jmi_salvanometag);

tagMenu.addSeparator();

jmi_aggiunginodotag = new JMenuItem("Aggiungi nuovo nodo tag xml");
jmi_aggiunginodotag.addActionListener(this);
jmi_aggiunginodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
tagMenu.add(jmi_aggiunginodotag);

jmi_modificanodotag = new JMenuItem("Modifica nodo tag xml");
jmi_modificanodotag.addActionListener(this);
jmi_modificanodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
tagMenu.add(jmi_modificanodotag);

jmi_eliminanodotag = new JMenuItem("Elimina nodo tag xml");
jmi_eliminanodotag.addActionListener(this);
jmi_eliminanodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancella.gif")));
tagMenu.add(jmi_eliminanodotag);

tagMenu.addSeparator();

jmi_aggiungiattributonodotag = new JMenuItem("Aggiungi attributo nodo tag xml");
jmi_aggiungiattributonodotag.addActionListener(this);
jmi_aggiungiattributonodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
tagMenu.add(jmi_aggiungiattributonodotag);

jmi_modificaattributonodotag = new JMenuItem("Modifica attributo nodo tag xml");
jmi_modificaattributonodotag.addActionListener(this);
jmi_modificaattributonodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
tagMenu.add(jmi_modificaattributonodotag);

jmi_eliminaattributonodotag = new JMenuItem("Elimina attributo nodo tag xml");
jmi_eliminaattributonodotag.addActionListener(this);
jmi_eliminaattributonodotag.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancella.gif")));
tagMenu.add(jmi_eliminaattributonodotag);

tagMenu.addSeparator();

jmi_cancellaSelezioneJTable = new JMenuItem("Deseleziona celle attributi e valori");
jmi_cancellaSelezioneJTable.addActionListener(this);
jmi_cancellaSelezioneJTable.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancellatable.gif")));
tagMenu.add(jmi_cancellaSelezioneJTable);

// definizione menu impostazioniMenu

JMenu impostazioniMenu = new JMenu("Impostazioni");

jmi_wordwrap = new JCheckBoxMenuItem("A capo automatico");
jmi_wordwrap.addActionListener(this);
impostazioniMenu.add(jmi_wordwrap);

jmi_carattere = new JMenuItem("Tipo carattere");
jmi_carattere.addActionListener(this);
impostazioniMenu.add(jmi_carattere);

jmi_coloretesto = new JMenuItem("Colore testo");
jmi_coloretesto.addActionListener(this);
impostazioniMenu.add(jmi_coloretesto);

jmi_coloresfondo = new JMenuItem("Colore sfondo");
```

```
jmi_coloresfondo.addActionListener(this);
impostazioniMenu.add(jmi_coloresfondo);

jmi_directory = new JMenuItem("Directory di lavoro");
jmi_directory.addActionListener(this);
impostazioniMenu.add(jmi_directory);

jmi_filexmltag = new JMenuItem("File tag di default (.xml)");
jmi_filexmltag.addActionListener(this);
impostazioniMenu.add(jmi_filexmltag);

jmi_dimensionefinestra = new JMenuItem("Dimensione finestra");
jmi_dimensionefinestra.addActionListener(this);
impostazioniMenu.add(jmi_dimensionefinestra);

// definisce un nuovo menu' e il codice per sottomenu' di definizione dell'aspetto della finestra

JMenu jmi_aspetto_menu = new JMenu("Aspetto finestra");
jmi_aspetto_menu.addActionListener(this);
impostazioniMenu.add(jmi_aspetto_menu);

jmi_aspetto1 = new JRadioButtonMenuItem("Metal");
jmi_aspetto1.addActionListener(this);
jmi_aspetto_menu.add(jmi_aspetto1);

jmi_aspetto2 = new JRadioButtonMenuItem("System");
jmi_aspetto2.addActionListener(this);
jmi_aspetto_menu.add(jmi_aspetto2);

jmi_aspetto3 = new JRadioButtonMenuItem("Motif");
jmi_aspetto3.addActionListener(this);
jmi_aspetto_menu.add(jmi_aspetto3);

jmi_aspetto4 = new JRadioButtonMenuItem("Windows");
jmi_aspetto4.addActionListener(this);
jmi_aspetto_menu.add(jmi_aspetto4);

jmi_aspetto5 = new JRadioButtonMenuItem("Mac");
jmi_aspetto5.addActionListener(this);
jmi_aspetto_menu.add(jmi_aspetto5);

ButtonGroup group = new ButtonGroup();
group.add(jmi_aspetto1);
group.add(jmi_aspetto2);
group.add(jmi_aspetto3);
group.add(jmi_aspetto4);
group.add(jmi_aspetto5);

if (lookandfeelCorrente.equals("Metal"))
        jmi_aspetto1.setSelected(true);

if (lookandfeelCorrente.equals("System"))
        jmi_aspetto2.setSelected(true);

if (lookandfeelCorrente.equals("Motif"))
        jmi_aspetto3.setSelected(true);

if (lookandfeelCorrente.equals("Windows"))
        jmi_aspetto4.setSelected(true);
```

```
		if (lookandfeelCorrente.equals("Mac"))
			jmi_aspetto5.setSelected(true);

		impostazioniMenu.addSeparator();

		jmi_dimensionipredefinite = new JMenuItem("Ripristina impostazioni predefinite");
		jmi_dimensionipredefinite.addActionListener(this);
		impostazioniMenu.add(jmi_dimensionipredefinite);

		// definizione menu helpMenu

		JMenu helpMenu = new JMenu("?");

		jmi_javahelpcontesto = new JMenuItem("Guida contestuale");
		//jmi_javahelpcontesto.addActionListener(this);
		jmi_javahelpcontesto.addActionListener(new CSH.DisplayHelpAfterTracking(hb));
		jmi_javahelpcontesto.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/helpcntx.gif")));
		helpMenu.add(jmi_javahelpcontesto);

		jmi_javahelp = new JMenuItem("Guida del programma");
		//jmi_javahelp.addActionListener(this);
		helpMenu.add(jmi_javahelp);
		CSH.setHelpIDString(jmi_javahelp, "top");
		jmi_javahelp.addActionListener(new CSH.DisplayHelpFromSource(hb));

		jmi_guida = new JMenuItem("Guida testuale");
		jmi_guida.addActionListener(this);
		helpMenu.add(jmi_guida);

		jmi_info = new JMenuItem("Informazioni sul programma");
		jmi_info.addActionListener(this);
		helpMenu.add(jmi_info);

		// aggiunge i singoli menu alla barra dei menu

		menuBar.add(fileMenu);
		menuBar.add(modificaMenu);
		menuBar.add(ricercaMenu);
		menuBar.add(xmlMenu);
		menuBar.add(tagMenu);
		menuBar.add(impostazioniMenu);
		menuBar.add(helpMenu);

		return menuBar; // ritorna la barra dei menu dopo tutti gli aggiornamenti
}


// **** crea la ToolBar dell'editor ****

private JToolBar crea_Barra()
{
		JToolBar barraComandi = new JToolBar("Barra comandi");
		CSH.setHelpIDString(barraComandi, "oggettoJToolBar");

		nuovo = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/nuovo.gif")));
		barraComandi.add(nuovo);
		nuovo.addActionListener(this);
		nuovo.setToolTipText("Nuovo file");

		apri = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/apri.gif")));
```

```
barraComandi.add(apri);
apri.addActionListener(this);
apri.setToolTipText("Apri il file");

salva = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/salva.gif")));
barraComandi.add(salva);
salva.addActionListener(this);
salva.setToolTipText("Salva il file");

stampa = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/stampa.gif")));
barraComandi.add(stampa);
stampa.addActionListener(this);
stampa.setToolTipText("Stampa il file (modalita' grafica)");

barraComandi.addSeparator();

annulla = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/annulla.gif")));
barraComandi.add(annulla);
annulla.addActionListener(this);
annulla.setToolTipText("Annulla l'ultima operazione");

ripristina = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/ripristina.gif")));
barraComandi.add(ripristina);
ripristina.addActionListener(this);
ripristina.setToolTipText("Ripristina l'ultima operazione");

barraComandi.addSeparator();

taglia = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/taglia.gif")));
barraComandi.add(taglia);
taglia.addActionListener(this);
taglia.setToolTipText("Taglia il contenuto selezionato");

copia = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/copia.gif")));
barraComandi.add(copia);
copia.addActionListener(this);
copia.setToolTipText("Copia il contenuto selezionato");

incolla = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/incolla.gif")));
barraComandi.add(incolla);
incolla.addActionListener(this);
incolla.setToolTipText("Incolla il contenuto selezionato");

barraComandi.addSeparator();

memolinea = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/memlinea.gif")));
barraComandi.add(memolinea);
memolinea.addActionListener(this);
memolinea.setToolTipText("Memorizza il numero di linea del cursore");

vaimemolinea = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/vailinea.gif")));
barraComandi.add(vaimemolinea);
vaimemolinea.addActionListener(this);
vaimemolinea.setToolTipText("Sposta il cursore al numero di linea memorizzato");

barraComandi.addSeparator();

trova = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/trova.gif")));
barraComandi.add(trova);
trova.addActionListener(this);
```

```
            trova.setToolTipText("Trova il testo nel documento");

            trovasuccessivo = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/trovas.gif")));
            barraComandi.add(trovasuccessivo);
            trovasuccessivo.addActionListener(this);
            trovasuccessivo.setToolTipText("Ripete la ricerca del testo nel documento");

            sostituisci = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/sostituisci.gif")));
            barraComandi.add(sostituisci);
            sostituisci.addActionListener(this);
            sostituisci.setToolTipText("Trova il testo nel documento e lo sostituisce");

            barraComandi.addSeparator();

            creaxml = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
            barraComandi.add(creaxml);
            creaxml.addActionListener(this);
            creaxml.setToolTipText("Crea il documento xml");

            benformatoxml = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/benformato.gif")));
            barraComandi.add(benformatoxml);
            benformatoxml.addActionListener(this);
            benformatoxml.setToolTipText("Salva il file xml e verifica che sia ben formato");

            aggiornaxml = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/aggiorna.gif")));
            barraComandi.add(aggiornaxml);
            aggiornaxml.addActionListener(this);
            aggiornaxml.setToolTipText("Salva il file xml, verifica che sia ben formato e aggiorna l'albero xml relativo al documento");

            aggiungitagxml = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
            barraComandi.add(aggiungitagxml);
            aggiungitagxml.addActionListener(this);
            aggiungitagxml.setToolTipText("Aggiunge al documento il tag xml relativo al nodo selezionato");

            barraComandi.addSeparator();

            helpcontesto = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/helpcntx.gif")));
            barraComandi.add(helpcontesto);
            helpcontesto.addActionListener(new CSH.DisplayHelpAfterTracking(hb));
            helpcontesto.setToolTipText("Apre la guida in linea relativa all'oggetto selezionato");

            return barraComandi;
}


// **** creazione della barra di stato dell'editor ****

private JPanel crea_StatusBar()
{
            stato = new JLabel("Pronto."); // label che evidenzia lo stato e le operazioni del programma
            colonnerighe = new JLabel("Riga: " + 0 + " Colonna: " + 0 + "  "); // label per i numeri di colonna e riga
            JPanel pannelloLabel = new JPanel(new BorderLayout(3,3)); // pannello per contenere le due label
            pannelloLabel.add(stato, BorderLayout.WEST); // posizionamento della label di stato
            pannelloLabel.add(colonnerighe, BorderLayout.EAST); // posizionamento della label colonnerighe

            return pannelloLabel;
}


// **** creazione pannello laterale dell'editor ****
```

```
private JPanel crea_PannelloAlberoXml()
{
	// crea un nodo generico che non ha ne' padre, ne' figli, ma che consente di avere figli
	DefaultMutableTreeNode nodoRadice = new DefaultMutableTreeNode("Albero xml vuoto");
	modelloAlbero = new DefaultTreeModel(nodoRadice); // crea un albero, passandovi la radice, in cui ogni nodo ha dei figli
	// restituisce un'istanza di JTree che mostra il nodo radice e l'albero e' creato usando uno specifico modello dati
	alberoXml = new JTree(modelloAlbero);
	// imposta la selezione dell'albero a SINGLE_TREE_SELECTION: consente di selezionare solo un nodo dell'albero per volta
	alberoXml.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
	alberoXml.setShowsRootHandles(true); // gli handles sono mostrati
	alberoXml.setEditable(false); // imposta l'albero JTree come non editabile
	CSH.setHelpIDString(alberoXml, "oggettoAlberoXml");

	 RenderAlbero render = new RenderAlbero(); // definisce un nuovo oggetto render di tipo RenderAlbero
	 alberoXml.setCellRenderer(render); // applica all'albero JTree il render con le associazioni di icone e tooltip relative

	JScrollPane scorreAlbero = new JScrollPane(); // viene creato uno scrollpane per l'albero JTree che visualizza il documento Xml
	scorreAlbero.getViewport().add(alberoXml); // l'albero e' posizionato in uno scrollpane
	scorreAlbero.setToolTipText("Rappresentazione ad albero del documento xml");

	/* modellotavolaAttributiXml e' un oggetto di tipo TavolaAttributi con campi nodoCorrente e
	  attributiCorrenti ed implementa TableModel, interfaccia usata da JTable per interrogare
	  un modello di dati ti tipo tabulare. Sono sufficienti due righe di codice per permettere
	  alla JTable di visualizzare qualsiasi modello dati che implementi TableModel */

	modellotavolaAttributiXml = new TavolaAttributi(); // nuovo oggetto TavolaAttributi che estende AbstractTableModel
	tavolaAttributiXml = new JTable(modellotavolaAttributiXml); // assegna alla JTable l'interfaccia specifica appena creata

	JScrollPane scorreTavola = new JScrollPane(tavolaAttributiXml);
	scorreTavola.getViewport().setBackground(tavolaAttributiXml.getBackground());

	JSplitPane splitAlbero = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, scorreAlbero, scorreTavola);
	splitAlbero.setDividerLocation((Integer.parseInt(finestraX))-250);
	//splitAlbero.setDividerSize(30);
	splitAlbero.setContinuousLayout(true);
	splitAlbero.setOneTouchExpandable(true); // fornisce simbolo per rapida esapansione-chiusura splitPane
	getContentPane().add(splitAlbero, BorderLayout.CENTER);

	JPanel pannelloAlberoXml = new JPanel(new BorderLayout());
	pannelloAlberoXml.add(splitAlbero, BorderLayout.CENTER);

	/* Se si e' interessati nel sapere quando la selezione cambia, bisogna implementare l'interfaccia
	  TreeSelectionListener ed aggiungere l'istanza usando il metodo addTreeSelectionListener.
	  valueChanged sara' invocato (una volta sola) quando cambia la selezione ovvero quando l'utente
	  clicca due volte consecutive sullo stesso nodo */

	// e' un listener che e' modificato quando cambia la selezione in un TreeSelectionModel*/
	TreeSelectionListener listenerAlbero = new TreeSelectionListener() // per rilevare quando l'utente seleziona un nodo nell'albero
	{
		public void valueChanged(TreeSelectionEvent e) // richiamato ogni qualvolta cambia la selezione di un valore
		{
			Node nodoselezionato = restituisciNodoSelezionato();

			// aggiorna l'albero con il nodo attualmente selezionato
			aggiornaAttributiNodoInTavola(nodoselezionato); // il valore null e' accettato
		}
	};
	alberoXml.addTreeSelectionListener(listenerAlbero); // assegna all'albero Xml il listener definito per eventi TreeSelection
```

```
		return (pannelloAlberoXml);
}


// **** carica l'attuale nodo selezionato e lo restituisce come istanza di XmlNodo ****

public XmlNodo restituisciNodoSelezionatoAlbero()
{
		TreePath path = alberoXml.getSelectionPath(); // restituisce il path del nodo selezionato grazie a getSelectionPath
		if (path == null) // restituisce null se il path ottenuto e' null
			return null;
		Object oggetto = path.getLastPathComponent(); // in oggetto restituisce l'ultimo componente che ha determinato questo path
		if (!(oggetto instanceof XmlNodo)) // se l'oggetto non e' istanza di XmlNodo ritorna null
			return null;
		return (XmlNodo)oggetto; // se l'oggetto che ha determinato l'ultimo path e' di tipo XmlNodo, restituiscilo con un casting
}


// **** restituisce l'attuale nodo selezionato richiamando restituisciNodoSelezionatoAlbero ****

public Node restituisciNodoSelezionato()
{
		XmlNodo nodoAlberoXml = restituisciNodoSelezionatoAlbero(); // in nodoAlberoXml viene messo il nodo attualmente selezionato
		if (nodoAlberoXml == null)
			return null;
		return nodoAlberoXml.restituisciXmlNodo(); // restituisce da un oggetto XmlNodo il nodo di tipo Node
}


// **** aggiorna l'albero xml con il nodo attualmente selezionato ****

public void aggiornaAttributiNodoInTavola(Node nodo)
{
		/* modellotavolaAttributiXml e' un oggetto di tipo TavolaAttributi con campi nodoCorrente e
		  attributiCorrenti ed implementa TableModel, interfaccia usata da JTable per interrogare
		  un modello di dati ti tipo tabulare */

		modellotavolaAttributiXml.assegnaNodo(nodo);

		/* TableModelEvent e' usato per notificare i listener che il modello della tavola e' cambiato
		  Tutte le righe contenenti dati nella tavola sono cambiate, i listener devono perdere ogni stato
		  basato sulle precedenti righe e richiedere al TableModel il nuovo numero di linee e di valori */

		tavolaAttributiXml.tableChanged(new TableModelEvent(modellotavolaAttributiXml));
}


// **** metodo invocato per creare i menu' Popup che appaiono premendo il pulsante destro del mouse sulla JTextArea ****

private void creaMenuPopup()
{
  JMenuItem menuItem; // definisce un oggetto di tipo JMenuItem per la definizione degli item dei menu' popup

  popupAreaTesto = new JPopupMenu();

  menuItem = new JMenuItem("Tag testo");
  menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_K, ActionEvent.CTRL_MASK));
  menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
  menuItem.addActionListener(this);
  menuItem.setActionCommand("tag testo popup");
```

```
        popupAreaTesto.add(menuItem);

        menuItem = new JMenuItem("Seleziona paragrafo tag");
        menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/selblocco.gif")));
        menuItem.addActionListener(this);
        menuItem.setActionCommand("seleziona tag testo popup");
        popupAreaTesto.add(menuItem);

        menuItem = new JMenuItem("Elimina tag testo");
        menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/canctag.gif")));
        menuItem.addActionListener(this);
        menuItem.setActionCommand("elimina tag testo popup");
        popupAreaTesto.add(menuItem);

        popupAreaTesto.addSeparator();

        menuItem = new JMenuItem("Taglia");
        menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_X, ActionEvent.CTRL_MASK));
        menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/taglia.gif")));
        menuItem.addActionListener(this);
        menuItem.setActionCommand("taglia testo popup");
        popupAreaTesto.add(menuItem);

        menuItem = new JMenuItem("Copia");
        menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, ActionEvent.CTRL_MASK));
        menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/copia.gif")));
        menuItem.addActionListener(this);
        menuItem.setActionCommand("copia testo popup");
        popupAreaTesto.add(menuItem);

        menuItem = new JMenuItem("Incolla");
        menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, ActionEvent.CTRL_MASK));
        menuItem.setIcon(new ImageIcon(ClassLoader.getSystemResource("./risorse/incolla.gif")));
        menuItem.addActionListener(this);
        menuItem.setActionCommand("incolla testo popup");
        popupAreaTesto.add(menuItem);

        popupAreaTesto.setOpaque(true);
        popupAreaTesto.setLightWeightPopupEnabled(true);

       // aggiunge al componente JTextArea il listener per fare comparire il menu' popup
      MouseListener popupListener = new PopupListener(popupAreaTesto);
      areaTesto.addMouseListener(popupListener);
    }


    // **** imposta le proprieta' dell'area di testo dell'editor ****

    private void impostaProprietaAreaTesto(JTextArea areaTesto)
    {
                // sets the preferences of each new and opened file.
                //areaTesto.setSelectedTextColor(Color.green);
                //areaTesto.setSelectionColor(Color.yellow);

                areaTesto.setBackground(coloreSfondoDefault);
                areaTesto.setForeground(coloreTestoDefault);
                areaTesto.setFont(carattereDefault);

                areaTesto.setLineWrap(wordWrapDefault.booleanValue()); // imposta se e' attivo o meno il word-wrap nell'area testo
                areaTesto.setWrapStyleWord(wordWrapDefault.booleanValue()); // imposta lo stile di word-wrapping
```

```
            jmi_wordwrap.setState(wordWrapDefault.booleanValue()); // modifica la selezione nel menu'
}


// **** creazione area di testo dell'editor ****

private JScrollPane crea_AreaTesto() // JScrollPane
{
            //areaTesto = new JTextArea(); // definizione dell'oggetto area di testo
            areaTesto = new UndoableTextArea();
            areaTesto.addCaretListener(this); // rileva i movimenti del cursore per il calcolo di riga e colonna
            impostaProprietaAreaTesto(areaTesto);
            JScrollPane areaScorrimento = new JScrollPane(areaTesto); // vista scrollabile per JTextArea
            creaMenuPopup(); // crea i menu' popup relativi all'areaTesto
            CSH.setHelpIDString(areaScorrimento, "oggettoJTextArea");

            return areaScorrimento;
}


// **** rimuove il carattere  che e' presente caricando i file ****

private void rimuoviCarattere()
{
            boolean modificato = false;

            String testo = areaTesto.getText();
            int posizioneCarattere = testo.indexOf("carattere non stampabile in Microsoft Word") ;
            while ( posizioneCarattere >= 0 )
            {
                        testo = testo.substring(0, posizioneCarattere ) + testo.substring(posizioneCarattere + "|".length() ) ;
                        posizioneCarattere = testo.indexOf("carattere non stampabile in Microsoft Word", posizioneCarattere);
                        modificato = true;
            }
            if (modificato == true)
            {
                          areaTesto.setText(testo);
                          areaTesto.setCaretPosition(0);
            }
}


// **** creazione split pane centrale dell'editor ****

private JSplitPane crea_SplitPaneCentrale()
{
            splitPaneCentrale = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
            splitPaneCentrale.setContinuousLayout(true);
            splitPaneCentrale.setOneTouchExpandable(true); // fornisce simbolo per rapida esapansione-chiusura splitPane
            splitPaneCentrale.setDividerLocation(Integer.parseInt(divisoreVerticale)); // impostazione posizione delimitatore verticale
            splitPaneCentrale.setLeftComponent(crea_AreaTesto());
            splitPaneCentrale.setRightComponent(crea_PannelloAlberoXml());

            splitPaneCentrale2 = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
            splitPaneCentrale2.setContinuousLayout(true);
            splitPaneCentrale2.setOneTouchExpandable(true); // fornisce simbolo per rapida esapansione-chiusura splitPane
            splitPaneCentrale2.setDividerLocation(Integer.parseInt(divisoreOrizzontale)); // impostazione posizione delimitatore orizzontale
            splitPaneCentrale2.setTopComponent(splitPaneCentrale);
            // splitPaneCentrale2.setBottomComponent(crea_PannelloAlberoXml()); linea di prova
            oggettoXmlTag = new XmlTag(this); // crea l'oggetto
```

```
            CSH.setHelpIDString(oggettoXmlTag, "oggettoXmlTag");

            splitPaneCentrale2.setBottomComponent(oggettoXmlTag);

            return splitPaneCentrale2;
}


// **** dichiarazione di un costruttore per la classe ****

private XMLStudio() // costruttore per la classe XMLStudio relativa all'editor
{
            // inizializzazioni dell'editor
            creaJavaHelp();

            carica_ImpostazioniDefault(); // carica le proprieta' di default dell'editor scelte dall'utente

            initLookAndFeel(lookandfeelCorrente); // richiama il metodo per l'impostazione del look e feel dell'interfaccia grafica

            screenSize = Toolkit.getDefaultToolkit().getScreenSize(); // rilevazione dimensioni dello schermo
            //this.setSize(screenSize); imposta la dimensione della finestra dell'applicazione a tutto schermo

            //GraphicsDevice device = GraphicsEnvironment.getLocalGraphicsEnvironment().getDefaultScreenDevice(); // altro metodo per impostare a tu
            //device.setFullScreenWindow(this);

            Dimension size = this.getSize(); // centra il JFrame al centro dello schermo
            screenSize.height = screenSize.height/2;
            screenSize.width = screenSize.width/2;
            size.height = size.height/2;
            size.width = size.width/2;
            this.setLocation(screenSize.height - size.height, screenSize.width - size.width);

            //GraphicsDevice device = GraphicsEnvironment.getLocalGraphicsEnvironment().getDefaultScreenDevice();
            //device.setFullScreenWindow(this);

            ultimiFile = new StoriaFile(); // crea un oggetto di tipo StoriaFile

            setTitle("XMLStudio"); // impostazione del titolo
            setJMenuBar(crea_MenuBar()); // creazione della barra dei menu

            Container contenitoreComponenti = getContentPane(); // ritorna l'oggetto ContentPane per il frame corrente
            contenitoreComponenti.setLayout(new BorderLayout()); // impostazione del layout del ContentPane del Frame

            contenitoreComponenti.add(crea_Barra(), BorderLayout.NORTH); // creazione e posizionamento ToolBar dell'editor
            contenitoreComponenti.add(crea_SplitPaneCentrale(), BorderLayout.CENTER); // inserimento dell'area di testo
            contenitoreComponenti.add(crea_StatusBar(), BorderLayout.SOUTH); // creazione e posizionamento StatusBar dell'editor

            // stato.setText("Dimensione schermo " + screenSize.toString());
            // stato.setText(carattereDefault.toString());

            this.setIconImage(Toolkit.getDefaultToolkit().createImage(getClass ().getResource("./risorse/icona.gif")));
            azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file

            addWindowListener(this);  // per la gestione dell'evento WindowClosing della finestra dell'editor
}


// **** il metodo windowClosing gestisce la chiusura della finestra ****

public void windowClosing(WindowEvent event)
```

```
	{
		azioneUscita();
	}


// **** sette metodi relativi alla finestra dell'editor, implementabili se necessario ****

	public void windowIconified(WindowEvent event)
	{
	}

	public void windowDeiconified(WindowEvent event)
	{
	}

	public void windowClosed(WindowEvent event)
	{
	}

	public void windowDeactivated(WindowEvent event)
	{
	}

	public void windowActivated(WindowEvent event)
	{
	}

	public void windowOpened(WindowEvent event)
	{
	}


	// **** rileva il movimento del cursore su areaTesto ****

	public void caretUpdate(CaretEvent evento)
	{
		try
		{
			areaTesto.requestFocus(); // richiama il focus sulla JTextArea areaTesto - disatt
			int colonna = evento.getDot() - areaTesto.getLineStartOffset(areaTesto.getLineOfOffset(evento.getDot())) + 1;
			riga = areaTesto.getLineOfOffset(evento.getDot()) + 1;
			colonnerighe.setText("Riga: " + riga + " Colonna: " + colonna + "  ");
			stato.setText("Pronto."); // indica l'avvenuto caricamento
			azioneSalvaPulsante(true); // abilita il pulsante per il salvataggio del file
			areaTesto.requestFocus(); // richiama il focus sulla JTextArea areaTesto
		}
		catch(Exception e)
		{
			System.err.println("Errore di rilevamento movimento cursore.");
		}
	}


	// **** metodi invocati quando avviene un action event ****

	public void actionPerformed(ActionEvent e)
	{
		String evento = e.getActionCommand();

		if (evento.equals("Nuovo") || e.getSource() == nuovo)
```

```
                azioneNuovoFile();
if (evento.equals("Apri...") || e.getSource() == apri)
                azioneApriFile();
if (evento.equals("Chiudi"))
                azioneChiudiFile();
if (evento.equals("Salva") || e.getSource() == salva)
                azioneSalvaFile();
if (evento.equals("Salva come..."))
                azioneSalvaFileNome();
if (evento.equals("Stampa (Windows)"))
                azioneStampaWindows();
if (evento.equals("Stampa (Linux)"))
                azioneStampaLinux();
if (evento.equals("Stampa (Unix)"))
                azioneStampaUnix();
if (evento.equals("Stampa (grafica)") || e.getSource() == stampa)
                azioneStampaGrafica();

if (evento.equals("Uscita"))
                azioneUscita();

if (evento.equals(ultimiFile.primo))
                azioneApriUltimoFile(ultimiFile.primo);
if (evento.equals(ultimiFile.secondo))
                azioneApriUltimoFile(ultimiFile.secondo);
if (evento.equals(ultimiFile.terzo))
                azioneApriUltimoFile(ultimiFile.terzo);
if (evento.equals(ultimiFile.quarto))
                azioneApriUltimoFile(ultimiFile.quarto);

if (evento.equals("Annulla") || e.getSource() == annulla)
                azioneAnnulla();
if (evento.equals("Ripristina") || e.getSource() == ripristina)
                azioneRipristina();
if (evento.equals("Taglia") || e.getSource() == taglia)
                azioneTaglia();
if (evento.equals("Copia") || e.getSource() == copia)
                azioneCopia();
if (evento.equals("Incolla") || e.getSource() == incolla)
                azioneIncolla();
if (evento.equals("Seleziona tutto"))
                azioneSelezionaTutto();

if (evento.equals("Trova") || e.getSource() == trova)
                azioneTrova();
if (evento.equals("Trova tag"))
                azioneTrovaTag();
if (evento.equals("Trova successivo") || e.getSource() == trovasuccessivo)
                azioneTrovaSuccessivo();
if (evento.equals("Sostituisci") || e.getSource() == sostituisci)
                azioneSostituisci();
if (evento.equals("Vai a linea"))
                azioneVaiALinea();
if (evento.equals("Memorizza numero linea") || e.getSource() == memolinea)
                azioneMemorizzaLinea();
if (evento.equals("Vai a numero linea") || e.getSource() == vaimemolinea)
                azioneVaiAMemorizzaLinea();

if (evento.equals("Crea documento xml") || e.getSource() == creaxml)
                azioneCreaXml();
```

```
if (evento.equals("Verifica documento xml ben formato") || e.getSource() == benformatoxml)
		azioneBenFormatoXml();
if (evento.equals("Aggiorna albero xml") || e.getSource() == aggiornaxml)
		azioneCreaAlberoXml();
if (evento.equals("Aggiungi tag xml") || e.getSource() == aggiungitagxml)
		azioneAggiungiTagXml();
if (evento.equals("Elimina tag xml"))
		azioneRimuoviTagXml();
if (evento.equals("Seleziona paragrafo tag xml"))
		azioneSelezionaTagXml();
if (evento.equals("Sostituisci &"))
		azioneSostituisciE();
if (evento.equals("Sostituisci <"))
		azioneSostituisciMinore();
if (evento.equals("Sostituisci >"))
		azioneSostituisciMaggiore();
if (evento.equals("Sostituisci '"))
		azioneSostituisciApostrofo();
if (evento.equals("Sostituisci \""))
		azioneSostituisciQuota();
if (evento.equals("Ripristina caratteri speciali"))
		azioneRipristinaCaratteriSpeciali();
if (evento.equals("Crea dtd da documento xml"))
		azioneCreaDtd();
if (evento.equals("Crea dtd e documento xml valido"))
    azioneCreaDocumentoXmlValido();
if (evento.equals("Crea css associato al file xml"))
		azioneCreaCss();

if (evento.equals("Crea nuovo file tag (.xml)"))
		oggettoXmlTag.azioneNuovoDocumentoXmlTag();
if (evento.equals("Apre file tag (.xml)"))
		oggettoXmlTag.azioneApriDocumentoXmlTag();
if (evento.equals("Deseleziona celle attributi e valori"))
		oggettoXmlTag.azioneCancellaSelezioneJTable();
if (evento.equals("Salva file tag (.xml)"))
		oggettoXmlTag.azioneSalvaDocumentoXmlTag(false);
if (evento.equals("Salva nome file tag (.xml)"))
		oggettoXmlTag.azioneSalvaDocumentoXmlTag(true);
if (evento.equals("Aggiungi nuovo nodo tag xml"))
		oggettoXmlTag.azioneAggiungiNodoXmlTag();
if (evento.equals("Modifica nodo tag xml"))
		oggettoXmlTag.azioneEditaNodoXmlTag();
if (evento.equals("Elimina nodo tag xml"))
		oggettoXmlTag.azioneCancellaNodo();
if (evento.equals("Aggiungi attributo nodo tag xml"))
		oggettoXmlTag.azioneAggiungiAttributoNodo();
if (evento.equals("Modifica attributo nodo tag xml"))
		oggettoXmlTag.azioneModificaAttributoNodo();
if (evento.equals("Elimina attributo nodo tag xml"))
		oggettoXmlTag.azioneEliminaAttributoNodo();

if (evento.equals("A capo automatico"))
		azioneWordWrap();
if (evento.equals("Tipo carattere"))
		azioneTipoCarattere();
if (evento.equals("Colore sfondo"))
		azioneTipoColoreSfondo();
if (evento.equals("Colore testo"))
		azioneTipoColoreTesto();
```

```
		if (evento.equals("Directory di lavoro"))
			azioneImpostaDirectory();
		if (evento.equals("File tag di default (.xml)"))
			azioneImpostaFileXmlTag();

		if (evento.equals("Dimensione finestra"))
			azioneDimensioneFinestra();
		if (evento.equals("Ripristina impostazioni predefinite"))
			ripristinaImpostazioniDefault();

		if (evento.equals("Metal"))
			azioneImpostaLookMetal();
		if (evento.equals("System"))
			azioneImpostaLookSystem();
		if (evento.equals("Windows"))
			azioneImpostaLookWindows();
		if (evento.equals("Motif"))
			azioneImpostaLookMotif();
		if (evento.equals("Mac"))
			azioneImpostaLookMac();

		if (evento.equals("Guida testuale"))
			azioneGuida();
		if (evento.equals("Informazioni sul programma"))
			azioneAboutBox();

		// gestione eventi relativi ai menu' popup
		if (evento.equals("tag testo popup"))
			azioneAggiungiTagXml();
		if (evento.equals("elimina tag testo popup"))
			azioneRimuoviTagXml();
		if (evento.equals("seleziona tag testo popup"))
			azioneSelezionaTagXml();
		if (evento.equals("taglia testo popup"))
			azioneTaglia();
		if (evento.equals("copia testo popup"))
			azioneCopia();
		if (evento.equals("incolla testo popup"))
			azioneIncolla();
}


// **** metodo che modifica il cursore dell'applicazione in clessidra ****

public void cursoreClessidra()
{
		getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // modifica il cursore in clessidra
		areaTesto.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // modifica il cursore in clessidra
		splitPaneCentrale.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // modifica il cursore in clessidra
		splitPaneCentrale2.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // modifica il cursore in clessidra
		oggettoXmlTag.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // modifica il cursore in clessidra
}


// **** metodo che modifica il cursore dell'applicazione in normale ****

public void cursoreNormale()
{
		getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // ripristina il cursore normale
		areaTesto.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // ripristina il cursore normale
```

```
            splitPaneCentrale.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // modifica il cursore in normale
            splitPaneCentrale2.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // modifica il cursore in normale
            oggettoXmlTag.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // modifica il cursore in normale
}


// **** metodi legati all'esecuzione di eventi da parte dell'utente ****

// **** azioneNuovoFile e' richiamato per creare un nuovo file vuoto ****

private void azioneNuovoFile()
{
            azioneRicordaSalvaFile();
            areaTesto.setVisible(true);
            areaTesto.setText("");
            setTitle("XMLStudio"); // imposta il nuovo titolo nell'editor
            fileCorrente = new File("Vuoto");
            stato.setText("Nuovo file.");
            areaTesto.cancelladatiUndoManager(); // rimuove i dati presenti nell'UndoManager
            azioneCreaAlberoXmlVuoto();  // reinizializza il viewer dell'albero Xml
            memorizzaLinea = null; // azzera il numero di linea sul quale spostarsi se richiesto
            azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
}


// **** azioneChiudiFile e' richiamato per chiudere il file correntemente in uso ****

private void azioneChiudiFile()
{
            azioneRicordaSalvaFile();
            areaTesto.setText(null); // cancella il testo contenuto nell'areaTesto
            setTitle("XMLStudio"); // imposta il nuovo titolo nell'editor
            stato.setText("Pronto.");
            areaTesto.setVisible(false); // disattiva l'areaTesto (la rende 'grigia')
            areaTesto.cancelladatiUndoManager(); // rimuove i dati presenti nell'UndoManager
            fileCorrente = new File("Vuoto");
            azioneCreaAlberoXmlVuoto(); // reinizializza il viewer dell'albero Xml
            memorizzaLinea = null; // azzera il numero di linea sul quale spostarsi se richiesto
            azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
}


// **** azioneApriFile e' richiamato per aprire il file ****

private void azioneApriFile()
{
            Thread runner = new Thread() // utilizza le thread per consentire la modifica del cursore in clessidra
            {
            public void run()
                        {
                                    azioneRicordaSalvaFile();
                                    cursoreClessidra();
                                    areaTesto.rimuoviUndoManager(); // cessa l'esecuzione dell'UndoManager

                                    JFileChooser fileChooser = new JFileChooser(directoryCorrente); // crea l'oggetto per l'apertura del file sulla directory

                                    fileChooser.setApproveButtonText("Apri"); // imposta il valore del bottone di esecuzione
                                    fileChooser.setDialogTitle("Apri file txt/xml"); // imposta il titolo della finestra Dialog

                                    // imposta ed aggiunge i filtri opportuni
```

```
                        fileChooser.resetChoosableFileFilters(); // reinizializza l'oggetto fileChooser
                        FiltroFile txtFiltro = new FiltroFile("txt", "File documenti *.TXT"); // imposta il filtro per i file .txt
                        FiltroFile xmlFiltro = new FiltroFile("xml", "File xml *.XML"); // imposta il filtro per i file .xml
                        fileChooser.addChoosableFileFilter(txtFiltro); // aggiunge il filtro alla finestra di apertura
                        fileChooser.addChoosableFileFilter(xmlFiltro); // aggiunge il filtro alla finestra di apertura
                        fileChooser.setFileFilter(txtFiltro); // imposta il filtro txtFiltro come filtro di default

                        int ritornoValore = fileChooser.showOpenDialog(XMLStudio.this); // mostra la finestra dialog per l'apertura del file

                        if (ritornoValore == JFileChooser.APPROVE_OPTION)
                        {
                        try
                        {
                                File f = fileChooser.getSelectedFile(); // associa all'oggetto f il file selezionato
                                if (f == null || !f.isFile()) // se f e' nullo o non e' un file allora esci dal metodo
                                        return;
                                fileCorrente = f; // in fileCorrente salva i dati relativi al file appena salvato per successivo uso
                                Reader in = new FileReader(f); // crea l'oggetto in per leggere il file
                                areaTesto.read(in, null); // riporta il contenuto dell'oggetto in su areaTesto
                                rimuoviCarattere(); // rimuove un carattere che non permette l'elaborazione
                                areaTesto.setVisible(true); // attiva l'areaTesto (la rende 'grigia')
                                setTitle("XMLStudio - " + f.getName()); // imposta il nuovo titolo nell'editor
                                stato.setText("File " + f.getName() + " caricato."); // indica l'avvenuto caricamento
                                areaTesto.creaUndoManager(); // reinizializza l'UndoManager
                                fileGiaSalvato = false; // il fileCorrente appena aperto non e' ancora stato salvato

                                        String nomefile = fileCorrente.getName();
                                        String estensione = nomefile.substring(nomefile.length()-4);
                                        if (estensione.equals(".txt") == true) // se il file e' testuale proponi la trasformazione in xml
                                        {
                                                ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera crea
                                                if (ritornoValore == JOptionPane.YES_OPTION)
                                                        azioneCreaXml();
                                                else
                                                        ultimiFile.salvaStoriaFile(fileCorrente); // il file caricato e' testuale ma nor
                                                        setJMenuBar(crea_MenuBar()); // creazione della barra dei menu
                                        }
                                        else
                                        {
                                                ultimiFile.salvaStoriaFile(fileCorrente); // il file caricato non ha estensione .txt
                                                setJMenuBar(crea_MenuBar()); // creazione della barra dei menu
                                        }
                                        azioneCreaAlberoXmlVuoto(); // reinizializza il viewer dell'albero Xml
                                        memorizzaLinea = null; // azzera il numero di linea sul quale spostarsi se richiesto
                                        azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
                        }
                        catch(Exception e)
                        {
                                stato.setText("Errore di apertura del file " + fileCorrente.getName() + " " + e);
                                System.err.println("Errore di apertura del file " + fileCorrente.getName() + " " + e);
                                cursoreNormale();
                        }
                        finally
                        {
                                        cursoreNormale();
                                }
                        }
                        cursoreNormale();
                }
};
```

```
		runner.start();
		//cursoreNormale();
}


// **** azioneApriUltimoFile e' richiamato per aprire il file più in alto nella history del menu File dell'editor ****

private void azioneApriUltimoFile(final String stringaultimofile)
{
		Thread runner = new Thread() // utilizza le thread per consentire la modifica del cursore in clessidra
		{
		public void run()
			{
				azioneRicordaSalvaFile();
				cursoreClessidra();
				areaTesto.rimuoviUndoManager(); // cessa l'esecuzione dell'UndoManager
				try
				{
					File f = new File(stringaultimofile); // associa all'oggetto f il file selezionato
					if (f.exists())
					{
						fileCorrente = f; // in fileCorrente salva i dati relativi al file appena salvato per successivo uso
						Reader in = new FileReader(f); // crea l'oggetto in per leggere il file
						areaTesto.read(in, null); // riporta il contenuto dell'oggetto in su areaTesto
						rimuoviCarattere();  // rimuove un carattere che non permette l'elaborazione
						areaTesto.setVisible(true); // attiva l'areaTesto (la rende 'grigia')
						setTitle("XMLStudio - " + f.getName()); // imposta il nuovo titolo nell'editor
						stato.setText("File " + f.getName() + " caricato."); // indica l'avvenuto caricamento
						areaTesto.creaUndoManager(); // reinizializza l'UndoManager
						fileGiaSalvato = false; // il fileCorrente appena aperto non e' ancora stato salvato
						ultimiFile.salvaStoriaFile(f); // aggiorna l'elenco dei file aperti di recente
						setJMenuBar(crea_MenuBar()); // creazione della barra dei menu
						azioneCreaAlberoXmlVuoto(); // reinizializza il viewer dell'albero Xml
						memorizzaLinea = null; // azzera il numero di linea sul quale spostarsi se richiesto
						azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
					}
					else
					{
						stato.setText("Probabilmente e' stato cancellato un file relativo ad ultimi file aperti.");
					}
				}
				catch(Exception e)
				{
				stato.setText("Errore di apertura dell'ultimo file." + e);
				System.err.println("Errore di apertura dell'ultimo file." + e);
			}
				finally
				{
					cursoreNormale();
				}
			}
		};
		runner.start();
}


// **** azioneSalvaFileNome è richiamato per salvare il file con un nome specifico ****

private void azioneSalvaFileNome()
{
```

```
            boolean salvareFile = false;

            JFileChooser fileChooser = new JFileChooser(directoryCorrente); // crea l'oggetto per l'apertura del file sulla directory corrente
            // imposta la corrente directory di lavoro, esempio /Tesi/Tesi XMLStudio oppure "."

            fileChooser.setApproveButtonText("Salva"); // imposta il valore del bottone di esecuzione
            fileChooser.setDialogTitle("Salva file xml"); // imposta il titolo della finestra Dialog

            // imposta ed aggiunge i filtri opportuni
            fileChooser.resetChoosableFileFilters(); // reinizializza l'oggetto fileChooser
            FiltroFile txtFiltro = new FiltroFile("txt", "File documenti *.TXT"); // imposta il filtro per i file .txt
            FiltroFile xmlFiltro = new FiltroFile("xml", "File xml *.XML"); // imposta il filtro per i file .xml
            fileChooser.addChoosableFileFilter(txtFiltro); // aggiunge il filtro alla finestra di apertura
            fileChooser.addChoosableFileFilter(xmlFiltro); // aggiunge il filtro alla finestra di apertura
            fileChooser.setFileFilter(xmlFiltro); // imposta il filtro xmlFiltro come filtro di default

int ritornoValore = fileChooser.showSaveDialog(getContentPane()); // mostra la finestra dialog per il salvataggio del file

if (ritornoValore == JFileChooser.APPROVE_OPTION)
{
                        try
                        {
                                File f = fileChooser.getSelectedFile(); // associa all'oggetto f il file selezionato
                                if (f.exists()) // il file selezionato esiste gia' nella directory corrente
                                {
                                    ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera sovrascrivere il file " + f.getName() +
                                        if (ritornoValore == JOptionPane.YES_OPTION)
                                                salvareFile = true; // e' stato selezionato di sovrascrivere il file
                                if (ritornoValore == JOptionPane.NO_OPTION)
                                                salvareFile = false; // e' stato selezionato di non sovrascrivere il file
                                }
                        else
                        {
                                salvareFile = true; // il file non esiste nella directory corrente e viene salvato
                        }

                        if (salvareFile == true) // gestisce il salvataggio del file
                        {
                                fileCorrente = f; // in fileCorrente salva i dati relativi al fine appena salvato per successivo uso
                                fileGiaSalvato = true; // il fileCorrente appena aperto e' stato salvato
                                Writer out = new FileWriter(f); // crea l'oggetto out per salvare il file
                                areaTesto.write(out); // riporta il contenuto di areaTesto sull'oggetto out
                                setTitle("XMLStudio - " + f.getName()); // imposta il nuovo titolo nell'editor
                                stato.setText("File " + f.getName() + " salvato."); // indica l'avvenuto caricamento
                                ultimiFile.salvaStoriaFile(fileCorrente); // il file caricato non ha estensione .txt
                                        setJMenuBar(crea_MenuBar()); // creazione della barra dei menu
                                        azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
                        }
                        else // non salva il file
                        {
                                stato.setText("File " + f.getName() + " non salvato."); // indica l'avvenuto caricamento
                        }
            }
            catch(Exception e)
            {
                        stato.setText("Errore di salvataggio del file " + fileCorrente.getName() + ".");
                        System.err.println("Errore di salvataggio del file " + fileCorrente.getName() + ".");
            }
            }
}
```

```
// **** metodo che abilita e disabilita l'icona per il salvataggio dei file ****

private void azioneSalvaPulsante(boolean abilitato)
{
	jmi_salva.setEnabled(abilitato);
	salva.setEnabled(abilitato);
}


// **** azioneSalvaFile è richiamato per salvare il file ****

private void azioneSalvaFile()
{
	if (fileGiaSalvato == false) // il fileCorrente appena aperto non e' ancora stato salvato
	{
		azioneSalvaFileNome(); // al file non e' ancora stato assegnato un nome
	}
	else
	{
		if (fileCorrente.isFile() == true)
		{
			try
			{
				Writer out = new FileWriter(fileCorrente); // crea l'oggetto out per salvare il file
				areaTesto.write(out); // riporta il contenuto di areaTesto sull'oggetto out
				setTitle("XMLStudio - " + fileCorrente.getName()); // imposta il nuovo titolo nell'editor
				stato.setText("File " + fileCorrente.getName() + " salvato."); // indica l'avvenuto caricamento
				azioneSalvaPulsante(false); // disabilita il pulsante per il salvataggio del file
			}
			catch(Exception e)
			{
				stato.setText("Errore di salvataggio del file " + fileCorrente.getName() + ".");
				System.err.println("Errore di salvataggio del file " + fileCorrente.getName() + ".");
			}
		}
		else
		{
			azioneSalvaFileNome(); // al file non e' ancora stato assegnato un nome
		}
	}
}


// **** chiede all'utente se desidera salvare il file corrente aperto e visualizzato nella JTextArea ****

private void azioneRicordaSalvaFile()
{
	if ((areaTesto.isVisible() == true) && (!fileCorrente.getName().equals("Vuoto")))
	{
		int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera salvare il file corrente " + fileCorrente.getName(
		if (ritornoValore == JOptionPane.YES_OPTION)
			azioneSalvaFile(); // salva il file
	}
}


// **** azioneStampaWindows consente di stampare il documento visibile nella JTextArea in ambiente Windows ****
```

```
private void azioneStampaWindows()
{
	// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
	if (!areaTesto.getText().equals("")) // se vi e' del testo in areaTesto
	{
		try
		{
			stato.setText("Stampa del documento in ambiente Windows.");
			PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("lpt1"))); //lpt1 in Windows consente di inviare il te
			out.print(areaTesto.getText());
			out.flush();
			out.close();
		}
		catch ( IOException e)
		{
			stato.setText("Errore di stampa su carta - Stampa Windows.");
			System.err.println("Errore di stampa su carta - Stampa Windows.");
		}
	}
	else
	{
		JOptionPane.showInternalMessageDialog(getContentPane(), "Non vi e' testo da stampare.", "Informazione", JOptionPane.INFORMATI
	}
	// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
}


// **** azioneStampaWindows consente di stampare il documento visibile nella JTextArea in ambiente Linux ****

private void azioneStampaLinux()
{
	// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
	if (!areaTesto.getText().equals("")) // se vi e' del testo in areaTesto
	{
		try
		{
			stato.setText("Stampa del documento in ambiente Linux.");
			PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("lp"))); //lpt1 in Windows consente di inviare il tes
			out.print(areaTesto.getText());
			out.flush();
			out.close();
		}
		catch ( IOException e)
		{
			stato.setText("Errore di stampa su carta - Stampa Linux.");
			System.err.println("Errore di stampa su carta - Stampa Linux.");
		}
	}
	else
	{
		JOptionPane.showInternalMessageDialog(getContentPane(), "Non vi e' testo da stampare.", "Informazione", JOptionPane.INFORMATI
	}
	// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
}


// **** azioneStampaWindows consente di stampare il documento visibile nella JTextArea in ambiente Unix ****

private void azioneStampaUnix()
{
```

```
		// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
		if (!areaTesto.getText().equals("")) // se vi e' del testo in areaTesto
		{
			try
			{
				stato.setText("Stampa del documento in ambiente Unix.");
				PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("/dev/lp"))); //lpt1 in Windows consente di inviarı
				out.print(areaTesto.getText());
				out.flush();
				out.close();
			}
			catch ( IOException e)
			{
				stato.setText("Errore di stampa su carta - Stampa Unix.");
				System.err.println("Errore di stampa su carta - Stampa Unix.");
			}
		}
		else
		{
			JOptionPane.showInternalMessageDialog(getContentPane(), "Non vi e' testo da stampare.", "Informazione", JOptionPane.INFORMATIO
		}
		// getContentPane().setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR)); // dovrebbe cambiare il cursorse in clessidra
}


// **** azioneStampaGrafica invia alla stampante il documento contenuto nella JTextArea ****

private void azioneStampaGrafica()
{
		if (!areaTesto.getText().equals("")) // se vi e' del testo in areaTesto
		{
			stato.setText("Stampa del documento in modalita' grafica.");
			PrintJob jobStampa = getToolkit().getPrintJob(this, "Stampa documento", null); // attiva un PrintJob per la corrente istanza di java.a
			if (jobStampa != null)
			{
				Graphics oggettoGrafico = jobStampa.getGraphics(); // richiede al PrintJob un oggetto grafico per stampare la pagina
				if (oggettoGrafico != null)
				{
					String testo = areaTesto.getText(); // in testo viene inserito il testo della JTextArea
					StampaGrafica oggettoStampaGrafica = new StampaGrafica(jobStampa, oggettoGrafico, testo, fileCorrente.g
					oggettoGrafico.dispose(); // rilascio di tutte le risorse che usavano questo contesto grafico
				}
			jobStampa.end(); // termina il PrintJob
			}
		}
		else
		{
			JOptionPane.showInternalMessageDialog(getContentPane(), "Non vi e' testo da stampare.", "Informazione", JOptionPane.INFORMATIO
		}
}


// **** azioneUscita è richiamato quando si esce dal programma ****

private void azioneUscita()
{
		// avvisa l'utente se si desidera salvare il file corrente aperto nell'editor
		azioneRicordaSalvaFile();

		// avvisa l'utente se si desidera salvare il file corrente relativo ai tag xml
```

```
                oggettoXmlTag.azioneRicordaSalva();

                System.out.println("Grazie per aver usato XMLStudio.");
                System.exit(0);
}


// **** azioneAnnulla, azioneRipristina realizzano le specifiche funzionalità di undo e redo dell'editor ****

private void azioneAnnulla()
{
                areaTesto.annulla();
}


// **** azioneAnnulla, azioneRipristina realizzano le specifiche funzionalità di undo e redo dell'editor ****

private void azioneRipristina()
{
                areaTesto.ripristina();
}


// **** azioneTaglia, azioneCopia, azioneIncolla realizzano le specifiche funzionalità nell'editor ****

private void azioneTaglia()
{
                areaTesto.cut();
}


// **** copia in memoria una porzione di testo selezionata nell'areaTesto ****

private void azioneCopia()
{
                areaTesto.copy();
}


// **** incolla una porzione di testo dalla memoria ****

private void azioneIncolla()
{
                areaTesto.paste();
}


// **** seleziona tutto il testo incluso in areaTesto ****

private void azioneSelezionaTutto()
{
                areaTesto.selectAll();
                areaTesto.requestFocus();
                stato.setText("Testo interamente selezionato.");
}


// **** azioneTrovaStringa individua del testo (in genere un carattere &, >, <, ', " all'interno del documento xml ****

private boolean azioneTrovaStringa(String parola) // gli passa un carattere che rende il documento invalido e non ben formato
```

```
{
	int posizione, // posizione della parola individuata
	posizioneiniziale, // posizione della fine della stringa <documento> dalla quale bisogna iniziare a cercare
	posizionefinale; // posizione dell'inizio della stringa </documento> prima della quale bisogna cercare

	if (parola != null) // se la stringa non e' nulla
	{
		String testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
		posizioneiniziale = testoCompleto.indexOf("<documento>") + 5; // ricerca in testoCompleto la stringa parola a partire da posizioneIn
		posizionefinale = testoCompleto.indexOf("</documento>");

		posizione = testoCompleto.indexOf(parola, posizioneiniziale + 1); // ricerca in testoCompleto la stringa parola a partire da salvaPosiz

		if ((posizione < posizionefinale -1) && (posizione != -1))// || (posizione == posizioneiniziale) || (posizione == posizionefinale))// restit
		{
			return true; // non ha trovato un carattere &, >, <, ', " o altra stringa nel documento xml
		}
		else
		{
			return false; // ha trovato un carattere &, >, <, ', " o altra stringa nel documento xml
		}
	}
	return false;
}


// **** azioneTrovaCaratteriNonValidi individua nel testo xml dei caratteri che non validi ****

private String azioneTrovaCaratteriNonValidi()
{
	String stringacompleta = null, // in stringacompleta verra' incluso l'eventuale sequenza di caratteri che rendono non valido e non ben formato il
		stringacaratterinonvalidi = "";

	if (azioneTrovaStringa("&"))
		stringacaratterinonvalidi = "& ";
	if (azioneTrovaStringa("<"))
		stringacaratterinonvalidi = stringacaratterinonvalidi + "< ";
	if (azioneTrovaStringa(">"))
		stringacaratterinonvalidi = stringacaratterinonvalidi + "> ";
	if (azioneTrovaStringa("'"))
		stringacaratterinonvalidi = stringacaratterinonvalidi + "' ";
	if (azioneTrovaStringa("\""))
		stringacaratterinonvalidi = stringacaratterinonvalidi + "\" ";
	if (stringacaratterinonvalidi.equals(null) == false)
		stringacompleta = "I caratteri non validi sono: " + stringacaratterinonvalidi;

	return stringacompleta;
}


// **** azioneTrova individua del testo all'interno dell'editor ****

private void azioneTrova()
{
	CreaDialog cd = new CreaDialog(this, "Trova", true, "Trova", "Ok"); // crea una finestra Dialog
	parola = cd.getString(); // in parola viene inserito la stringa digitata

	if (parola != null)// se la stringa non e' nulla
	{
		parola = cd.getString().toLowerCase(); // in parola viene inserito la stringa digitata
```

```
                String testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto

                int posizione = testoCompleto.indexOf(parola); // restituisce la posizione in cui trova in testoCompleto la prima occorrenza di parola

                if (posizione == -1) // restituisce -1 se non individua la stringa parola in testoCompleto
                {
                        Toolkit tk = Toolkit.getDefaultToolkit();
                        tk.beep(); // emette un suono beep
                        stato.setText("Vocabolo " + parola + " non trovato: superata la fine del file.");
                }
                else
                {
                        areaTesto.select(posizione, posizione + parola.length()); // seleziona il testo individuato tra le specifiche locazioni
                        areaTesto.requestFocus();
                        stato.setText("Parola " + parola + " individuata.");
                }
        }
}


// **** azioneTrova individua un tag di testo all'interno dell'editor ****

private void azioneTrovaTag()
{
        CreaDialog cd = new CreaDialog(this, "Trova tag", true, "Trova tag", "Ok"); // crea una finestra Dialog
        parola = cd.getString(); // in parola viene inserito la stringa digitata

        if (parola != null)// se la stringa non e' nulla
        {
                parola = "<" + cd.getString().toLowerCase(); // in parola viene inserito la stringa digitata
                String testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto

                int posizione = testoCompleto.indexOf(parola); // restituisce la posizione in cui trova in testoCompleto la prima occorrenza di parola

                if (posizione == -1) // restituisce -1 se non individua la stringa parola in testoCompleto
                {
                        Toolkit tk = Toolkit.getDefaultToolkit();
                        tk.beep(); // emette un suono beep
                        stato.setText("Vocabolo " + parola + " non trovato: superata la fine del file.");
                }
                else
                {
                        areaTesto.select(posizione, posizione + parola.length()); // seleziona il testo individuato tra le specifiche locazioni
                        areaTesto.requestFocus();
                        stato.setText("Parola " + parola + " individuata.");
                }
        }
}


// **** azioneTrovaSuccessivo ripete la ricerca della parola introdotta in Trova ****

private void azioneTrovaSuccessivo()
{
        if (parola == null)
        {
                azioneTrova();
                return;
        }
```

```
		String testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
		int posizioneInizio = areaTesto.getSelectionEnd(); // in posizioneInizio inserisce il valore finale evidenziato dalla precedente ricerca
		int posizione = testoCompleto.indexOf(parola, posizioneInizio); // ricerca in testoCompleto la stringa parola a partire da posizioneInizio

		if (posizione == -1) // restituisce -1 se non individua la stringa parola in testoCompleto
		{
			Toolkit tk = Toolkit.getDefaultToolkit();
			tk.beep(); // emette un suono beep
			stato.setText("Vocabolo " + parola + " non trovato: superata la fine del file.");
		}
		else
		{
			areaTesto.select(posizione, posizione + parola.length()); // seleziona il testo individuato tra le specifiche locazioni
			areaTesto.requestFocus();
			stato.setText("Parola " + parola + " individuata.");
		}
}


// **** azioneSostituisciE trova e sostituisce le occorrenze di & nel documento xml per renderlo ben formato ****

private void azioneSostituisciE()
{
		Object[] options = {"Manuale", "Automatica"};
		int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JOptionPane.I
		if (ritornoValore == JOptionPane.YES_OPTION)
		{
			azioneSostituisciManuale("&", "&amp;"); // sostituzione manuale
		}
		else
		{
			azioneSostituisciAutomatico("&", "&amp;"); // sostituzione automatica
		}
}


// **** azioneSostituisciMinore trova e sostituisce le occorrenze di < nel documento xml per renderlo ben formato ****

private void azioneSostituisciMinore()
{
		Object[] options = {"Manuale", "Automatica"};
		int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JOptionPane.I
		if (ritornoValore == JOptionPane.YES_OPTION)
		{
			azioneSostituisciManuale("<", "&lt;"); // sostituzione manuale
		}
		else
		{
			azioneSostituisciAutomatico("<", "&lt;"); // sostituzione automatica
		}
}


// **** azioneSostituisciMaggiore trova e sostituisce le occorrenze di > nel documento xml per renderlo ben formato ****

private void azioneSostituisciMaggiore()
{
		Object[] options = {"Manuale", "Automatica"};
		int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JOptionPane.I
		if (ritornoValore == JOptionPane.YES_OPTION)
```

```
		{
			azioneSostituisciManuale(">", "&gt;"); // sostituzione manuale
		}
		else
		{
			azioneSostituisciAutomatico(">", "&gt;"); // sostituzione automatica
		}
}


// **** azioneSostituisciApostrofo trova e sostituisce le occorrenze di ' nel documento xml per renderlo ben formato ****

private void azioneSostituisciApostrofo()
{
		Object[] options = {"Manuale", "Automatica"};
		int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JOptionPane.I
		if (ritornoValore == JOptionPane.YES_OPTION)
		{
			azioneSostituisciManuale("'", "&apos;"); // sostituzione manuale
		}
		else
		{
			azioneSostituisciAutomatico("'", "&apos;"); // sostituzione automatica
		}
}


// **** azioneSostituisciQuota trova e sostituisce le occorrenze di " nel documento xml per renderlo ben formato ****

private void azioneSostituisciQuota()
{
		Object[] options = {"Manuale", "Automatica"};
		int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JOptionPane.I
		if (ritornoValore == JOptionPane.YES_OPTION)
		{
			azioneSostituisciManuale("\"", "&quot;"); // sostituzione manuale
		}
		else
		{
			azioneSostituisciAutomatico("\"", "&quot;"); // sostituzione automatica
		}
}


// **** i caratteri speciali &, <, >, ', " vengono ripristinati (dalla sequenza di escape) ****

private void azioneRipristinaCaratteriSpeciali()
{
		try
		{
			azioneSostituisciAutomatico("&amp;", "&"); // sostituzione automatica
			azioneSostituisciAutomatico("&lt;", "<"); // sostituzione automatica
			azioneSostituisciAutomatico("&gt;", ">"); // sostituzione automatica
			azioneSostituisciAutomatico("&apos;", "'"); // sostituzione automatica
			azioneSostituisciAutomatico("&quot;", "\""); // sostituzione automatica
			stato.setText("Caratteri speciali ripristinati: documento xml non piu' ben formato.");
		}
		catch (Exception e)
		{
			stato.setText("Eccezione rilevata nel ripristino caratteri speciali");
```

```
		}
}


// **** azioneSostituisciAutomatico trova e sostituisce del testo all'interno dell'editor ****

private void azioneSostituisciAutomatico(String parola, String parolaSostituisci)
{
		String testoCompleto;
		int posizione, salvaposizione, posizionefinale = -1;
		boolean flagxml;

		cursoreClessidra();
		String nomefile = fileCorrente.getName();
		String estensione = nomefile.substring(nomefile.length()-4);
		if (estensione.equals(".xml") == true) // l'estensione del file corrente e' .txt
		{
			flagxml = true;
		}
		else
		{
			flagxml = false;
		}

		if ((parola.equals("<") == true) || (parola.equals(">") == true)) // caso speciale a causa dei tag <documento> e </documento>
		{
			testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
			if (parola.equals("<"))
				posizione = testoCompleto.indexOf("<documento>"); // ricerca in testoCompleto la stringa parola a partire da posizioneIn
			else
				posizione = testoCompleto.indexOf("<documento>") + 11; // ricerca in testoCompleto la stringa parola a partire da posizio
			salvaposizione = posizione;
			posizione = testoCompleto.indexOf(parola, salvaposizione + 1); // ricerca in testoCompleto la stringa parola a partire da salvaPosizio
			salvaposizione = posizione;
		}
		else
		{
			testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
			posizione = testoCompleto.indexOf(parola); // ricerca in testoCompleto la stringa parola a partire da posizioneInizio
			salvaposizione = posizione;
		}

		if ((parola != null) && (parolaSostituisci != null)) // se le stringhe non sono nulle
		{
			do
			{
				if (posizione == -1) // restituisce -1 se non individua la stringa parola in testoCompleto
				{
					Toolkit tk = Toolkit.getDefaultToolkit();
					tk.beep(); // emette un suono beep
					stato.setText("Vocabolo " + parola + " non trovato: superata la fine del file.");
				}
				else
				{
					areaTesto.select(posizione, posizione + parola.length()); // seleziona il testo individuato tra le specifiche loca
					areaTesto.requestFocus();

					//areaTesto.replaceRange(parolaSostituisci, posizione, posizione + parola.length()); // seleziona il testo indi
					areaTesto.replaceSelection(parolaSostituisci);
					areaTesto.requestFocus();
```

```
                                        stato.setText("Parola " + parola + " sostituita con " + parolaSostituisci + ".");
                                        //posizione = testoCompleto.indexOf(parola); // ricerca in testoCompleto la stringa parola a partire da posiz
                                        testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
                                        posizione = testoCompleto.indexOf(parola, salvaposizione + 1); // ricerca in testoCompleto la stringa parola a
                                        salvaposizione = posizione;

                                        if (flagxml) // se il file e' xml deve terminare cercando la stringa </documento> altrimenti deve terminare co
                                        {
                                                posizionefinale = testoCompleto.indexOf("</documento>"); // ricerca in testoCompleto la stringa
                                        }
                                        else
                                        {
                                                posizionefinale = testoCompleto.length();
                                        }
                                }
                        } while ((posizione != -1) && (posizione < posizionefinale));
                }
                cursoreNormale();
}


// **** azioneSostituisciManuale trova e sostituisce una per una le parole nel testo ****

private void azioneSostituisciManuale(String parola, String parolaSostituisci)
{
                boolean esci = false;

                String testoCompleto;
                int posizione, salvaposizione, posizionefinale = -1 ;
                boolean flagxml;

                String nomefile = fileCorrente.getName();
                String estensione = nomefile.substring(nomefile.length()-4);
                if (estensione.equals(".xml") == true) // l'estensione del file corrente e' .txt
                {
                        flagxml = true;
                }
                else
                {
                        flagxml = false;
                }

                if ((parola.equals("<") == true) || (parola.equals(">") == true)) // caso speciale a causa dei tag <documento> e </documento>
                {
                        testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
                        if (parola.equals("<"))
                                posizione = testoCompleto.indexOf("<documento>"); // ricerca in testoCompleto la stringa parola a partire da posizioneIn
                        else
                                posizione = testoCompleto.indexOf("<documento>") + 11; // ricerca in testoCompleto la stringa parola a partire da posizio
                        salvaposizione = posizione;
                        posizione = testoCompleto.indexOf(parola, salvaposizione + 1); // ricerca in testoCompleto la stringa parola a partire da salvaPosizio
                        salvaposizione = posizione;
                }
                else
                {
                        testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di areaTesto
                        posizione = testoCompleto.indexOf(parola); // ricerca in testoCompleto la stringa parola a partire da posizioneInizio
                        salvaposizione = posizione;
                }
```

```
		if ((parola != null) && (parolaSostituisci != null)) // se le stringhe non sono nulle
		{
			do
			{
				if (posizione == -1) // restituisce -1 se non individua la stringa parola in testoCompleto
				{
					Toolkit tk = Toolkit.getDefaultToolkit();
					tk.beep(); // emette un suono beep
					stato.setText("Vocabolo " + parola + " non trovato: superata la fine del file.");
				}
				else
				{
					areaTesto.select(posizione, posizione + parola.length()); // seleziona il testo individuato tra le specifiche loca
					areaTesto.requestFocus();

					int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera sostituire " + parola + " c
					if (ritornoValore == JOptionPane.YES_OPTION)
					{
						//areaTesto.replaceRange(parolaSostituisci, posizione, posizione + parola.length()); // seleziona
						areaTesto.replaceSelection(parolaSostituisci);
						areaTesto.requestFocus();
						stato.setText("Parola " + parola + " sostituita con " + parolaSostituisci + ".");
						//posizione = testoCompleto.indexOf(parola); // ricerca in testoCompleto la stringa parola a par
						testoCompleto = areaTesto.getText().toLowerCase(); // in testoCompleto inserisce il testo di are
						posizione = testoCompleto.indexOf(parola, salvaposizione + 1); // ricerca in testoCompleto la strir
						salvaposizione = posizione;
					}
					if (ritornoValore == JOptionPane.NO_OPTION)
					{
						stato.setText("Occorrenza non sostituita.");
						posizione = testoCompleto.indexOf(parola, areaTesto.getSelectionEnd()); // ricerca in testoComp
						salvaposizione = posizione;
					}
					if (ritornoValore == JOptionPane.CANCEL_OPTION)
						esci = true;
					if (flagxml) // se il file e' xml deve terminare cercando la stringa </documento> altrimenti deve terminare co
						posizionefinale = testoCompleto.indexOf("</documento>"); // ricerca in testoCompleto la stringa
					else
						posizionefinale = testoCompleto.length();
				}
			} while ((posizione != -1) && (esci == false) && (posizione < posizionefinale));
		}
}


// **** azioneSostituisci trova e sostituisce del testo all'interno dell'editor ****

private void azioneSostituisci()
{
		CreaDialog cd = new CreaDialog(this, "Sostituisci", true, "Trova       ", "Sostituisci", "Ok", "Annulla"); // crea una finestra Dialog
		parola = cd.getString();
		String parolaSostituisci = cd.getString2();

		if ((parola != null) && (parolaSostituisci != null)) // se la stringa non e' nulla
		{
			parola = cd.getString().toLowerCase(); // in parola viene inserito la stringa digitata da cercare
			parolaSostituisci = cd.getString2().toLowerCase(); // in parolaSostituisci viene inserita la stringa digitata da sostituire

			Object[] options = {"Manuale", "Automatica"};
			int ritornoValore =  JOptionPane.showOptionDialog(getContentPane(), "Scegliere un'opzione, sostituzione di tipo:", "Informazione", JO
```

```
                        if (ritornoValore == JOptionPane.YES_OPTION)
                        {
                                azioneSostituisciManuale(parola, parolaSostituisci); // sostituzione manuale delle parole
                        }
                        else
                        {
                                azioneSostituisciAutomatico(parola, parolaSostituisci); // sostituzione automatica delle parole
                        }
                }
}


// **** posiziona il cursore ad un numero di riga del documento identificato da linea ****

private void vaiALinea(String linea)
{
        if(linea != null) // se la stringa non e' nulla
        {
                try
                {
                        int valoreLinea = Integer.parseInt(linea); // conversione di linea in valore intero
                        if(valoreLinea >= 1)
                        {
                                areaTesto.select(areaTesto.getLineStartOffset(valoreLinea - 1), areaTesto.getLineEndOffset(valoreLinea - 1) -
                                areaTesto.requestFocus(); // posiziona il focus su questo oggetto
                                stato.setText("Cursore spostato alla linea " + linea + ".");
                        }
                        else
                                stato.setText(linea + " non e' un numero di linea valido.");
                }
                catch(Exception e)
                {
                        stato.setText(linea + " non e' un numero di linea valido.");
                }

        }
}


// **** azioneVaiALinea sposta il cursore alla linea specificata ****

private void azioneVaiALinea()
{
        CreaDialog cd = new CreaDialog(this, "Vai alla linea numero", true, "1 - " + areaTesto.getLineCount(), "Vai"); // crea una finestra Dialog
        String linea = cd.getString(); // in linea viene inserito la stringa linea
        if(linea != null) // se la stringa non e' nulla
        {
                vaiALinea(linea);
        }
        cd.dispose(); // chiude la finestra Dialog
}


// **** azioneMemorizzaLinea salva in memorizzaLinea il numero di linea del cursore nel documento attivo ****

private void azioneMemorizzaLinea()
{
        memorizzaLinea = String.valueOf(riga); // memorizza il valore di riga relativo alla posizione del cursore
        if(memorizzaLinea != null) // se la stringa non e' nulla
        {
```

```
                stato.setText("Memorizzato numero linea " + memorizzaLinea + ".");
        }
}


// **** azioneVaiAMemorizzaLinea sposta il cursore al numero di linea salvato in memorizzaLinea ****

private void azioneVaiAMemorizzaLinea()
{
        if(memorizzaLinea != null) // se la stringa non e' nulla
        {
                vaiALinea(memorizzaLinea);
        }
        else
        {
                JOptionPane.showInternalMessageDialog(getContentPane(), "E' prima necessario memorizzare un numero di linea al quale spostarsi
        }
}


// **** metodo usato da azioneCreaXml per aggiungere le intestazioni xml al file testuale aperto ****

private void aggiungiIntestazioniXml()
{
        areaTesto.insert("<?xml version=\"1.0\" encoding=\"iso-8859-1\" standalone=\"yes\"?>\n<documento>\n", 0);
        areaTesto.append("</documento>");
        stato.setText("File Xml creato, salvare il file per proseguire.");
        azioneSalvaFileNome(); // salva il file corrente assegnandogli un nome
        ultimiFile.salvaStoriaFile(fileCorrente); // aggiorna l'elenco dei file aperti di recente
        setJMenuBar(crea_MenuBar()); // creazione della barra dei menu
}


// **** crea il file xml aggiungendo le intestazioni opportune ****

private void azioneCreaXml()
{
        if (fileCorrente.exists())
        {
                // se il file caricato e' .txt, aggiungi i tag xml all'inizio e alla fine del documento
                String nomefile = fileCorrente.getName();
                String estensione = nomefile.substring(nomefile.length()-4);
                if (estensione.equals(".txt") == true) // l'estensione del file corrente e' .txt
                {
                        aggiungiIntestazioniXml();
                }
                if (estensione.equals(".xml") == true) // l'estensione del file corrente e' .xml
                {
                        //JOptionPane.showInternalMessageDialog(getContentPane(), "Il file aperto e' gia' un file Xml.", "Informazione", JOptionF
                        int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Il file aperto e' gia' xml. Si desidera comunque ag
                        if (ritornoValore == JOptionPane.YES_OPTION)
                        {
                                aggiungiIntestazioniXml();
                        }
                }
        }
        else
        {
                // JOptionPane.showInternalMessageDialog(getContentPane(), "E' prima necessario aprire un documento testuale.", "Informazione"
                int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "E' prima necessario aprire un documento testuale. Si deside
```

```
                        if (ritornoValore == JOptionPane.YES_OPTION)
                        {
                                aggiungiIntestazioniXml();
                        }
                }
}


// **** azioneImpostaDirectory è richiamato per impostare la directory corrente di lavoro ****

private void azioneImpostaDirectory()
{
                int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera impostare la directory del programma anche come directory
                if (ritornoValore == JOptionPane.YES_OPTION)
                {
                        directoryCorrente = ".";
                }
                else
                {
                        directoryCorrente = JOptionPane.showInputDialog("Inserire un nuovo path di directory dei documenti");
                        if (directoryCorrente == null) directoryCorrente = ".";
                }
                salva_ImpostazioniDefault();
                stato.setText("Impostazione del nuovo path di lavoro: " + directoryCorrente);
}


// **** azioneImpostaFileXmlTag è richiamato per impostare il file xml che contiene i tag da aggiungere al documento ****

private void azioneImpostaFileXmlTag()
{
                int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera impostare XMLStudio.xml come file di default per i tag?", "Do
                if (ritornoValore == JOptionPane.YES_OPTION)
                {
                        fileXmlTagCorrente = "./tag/XMLStudio.xml";
                }
                else
                {
                        fileXmlTagCorrente = JOptionPane.showInputDialog("Inserire un nuovo file (xml) di default per i tag");
                        if (fileXmlTagCorrente == null)
                        {
                                fileXmlTagCorrente = "./tag/XMLStudio.xml";
                        }
                        else
                        {
                                try
                                {
                                        File f = new File(fileXmlTagCorrente); // associa all'oggetto f il file selezionato
                                        if (f.exists())
                                        {
                                                String nomefile = f.getName();
                                                String estensione = nomefile.substring(nomefile.length()-4);
                                                if (estensione.equals(".xml") == false) // se il file non e' xml segnala che deve esserlo
                                                {
                                                        JOptionPane.showInternalMessageDialog(getContentPane(), "Il file di default per i tag
                                                        fileXmlTagCorrente = "./tag/XMLStudio.xml";
                                                }
                                        }
                                        else
                                        {
```

```
                                                JOptionPane.showInternalMessageDialog(getContentPane(), "Attenzione, il file digitato e' inesisten
                                                fileXmlTagCorrente = "./tag/XMLStudio.xml";
                                        }
                                }
                                catch (Exception e)
                                {
                                        //JOptionPane.showInternalMessageDialog(getContentPane(), "Il file di default per i tag deve essere xml. Rile
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Eccezione rilevata: " +
                                }
                        }
}
salva_ImpostazioniDefault();
stato.setText("Impostazione del file di default per tag: " + fileXmlTagCorrente);
}


// **** azioneDimensioneFinestra e' richiamato per memorizzare le nuove dimensioni della finestra dell'editor ****

private void azioneDimensioneFinestra()
{
        CreaDialog cd = new CreaDialog(this, "Imposta la dimensione finestra", true, "X (800)", "Y (600)", "Pannello orizzontale (350)", "Pannello vertic

        finestraX = cd.getString();
        finestraY = cd.getString2();
        divisoreOrizzontale = cd.getString3();
        divisoreVerticale = cd.getString4();

        if ((finestraX != null) && (finestraY != null) && (divisoreOrizzontale != null) && (divisoreVerticale != null)) // se le stringhe  non sono nulle
        {
                salva_ImpostazioniDefault();
                stato.setText("Nuove dimensioni finestra selezionate: " + finestraX + " " + finestraY + " " + divisoreOrizzontale + " " + divisoreVertica

                azioneRiavviaEditor();
        }
}


// **** azioneTipoColoreSfondo e' richiamato per modificare il colore dello sfondo dell'editor ****

private void azioneTipoColoreSfondo()
{
        coloreSfondoDefault = JColorChooser.showDialog(this, "Colore sfondo", Color.white);
        impostaProprietaAreaTesto(areaTesto);
        salva_ImpostazioniDefault();
        stato.setText("Impostazione del nuovo colore dello sfondo.");
}


// **** azioneTipoColoreTesto e' richiamato per modificare il colore dello sfondo dell'editor ****

private void azioneTipoColoreTesto()
{
        coloreTestoDefault = JColorChooser.showDialog(this, "Colore sfondo", Color.black);
        impostaProprietaAreaTesto(areaTesto);
        salva_ImpostazioniDefault();
        stato.setText("Impostazione del nuovo colore del testo.");
}


// **** azioneTipoCarattere e' richiamato per cambiare il tipo di carattere nell'editor ****
```

```
private void azioneTipoCarattere()
{
	stato.setText("Scelta del nuovo carattere di default.");
	ImpostaFont scegliFont = new ImpostaFont(this); // crea una dialog Window passando il frame corrente come owner
	if (scegliFont.returnFont() !=null)
	{
		areaTesto.setFont(scegliFont.returnFont());
		carattereDefault = scegliFont.returnFont();
		stato.setText("Nuovo carattere di default selezionato: " + carattereDefault.toString());
		salva_ImpostazioniDefault();
	}
	else
		stato.setText("Non e' stato selezionato alcun font.");
}


// **** imposta nell'area testo l'attivazione o meno della funzione di "A capo automatico" (detto Word-Wrap) ****

private void azioneWordWrap()
{
	if (wordWrapDefault.booleanValue() == true) // modifica l'impostazione del word-wrap nell'area testo
	{
		wordWrapDefault = new Boolean("False");
	}
	else
	{
		wordWrapDefault = new Boolean("True");
	}
	areaTesto.setLineWrap(wordWrapDefault.booleanValue()); // imposta se e' attivo o meno il word-wrap nell'area testo
	areaTesto.setWrapStyleWord(wordWrapDefault.booleanValue()); // imposta lo stile di word-wrapping
	jmi_wordwrap.setState(wordWrapDefault.booleanValue()); // modifica la selezione nel menu'
	stato.setText("Impostata proprieta' a capo automatico.");
	salva_ImpostazioniDefault();
}


// **** azioneImpostaLookMetal e' richiamato per cambiare il lookandfeel della finestra in "Metal" ****

private void azioneImpostaLookMetal()
{
	lookandfeelCorrente = "Metal";
	initLookAndFeel(lookandfeelCorrente);
	salva_ImpostazioniDefault();
	stato.setText("Nuovo lookandfeel impostato: Metal, riavviare l'applicazione.");
	azioneRiavviaEditor();
}


// **** azioneImpostaLookWindows e' richiamato per cambiare il lookandfeel della finestra in "Windows" ****

private void azioneImpostaLookWindows()
{
	lookandfeelCorrente = "Windows";
	initLookAndFeel(lookandfeelCorrente);
	salva_ImpostazioniDefault();
	stato.setText("Nuovo lookandfeel impostato: Windows, riavviare l'applicazione.");
	azioneRiavviaEditor();
}
```

```
// **** azioneImpostaLookSystem e' richiamato per cambiare il lookandfeel della finestra in "System" ****

private void azioneImpostaLookSystem()
{
		lookandfeelCorrente = "System";
		initLookAndFeel(lookandfeelCorrente);
		salva_ImpostazioniDefault();
		stato.setText("Nuovo lookandfeel impostato: System, riavviare l'applicazione.");
		azioneRiavviaEditor();
}


// **** azioneImpostaLookMotif e' richiamato per cambiare il lookandfeel della finestra in "Motif" ****

private void azioneImpostaLookMotif()
{
		lookandfeelCorrente = "Motif";
		initLookAndFeel(lookandfeelCorrente);
		salva_ImpostazioniDefault();
		stato.setText("Nuovo lookandfeel impostato: Motif, riavviare l'applicazione.");
		azioneRiavviaEditor();
}


// **** azioneImpostaLookMac e' richiamato per cambiare il lookandfeel della finestra in "Mac" ****

private void azioneImpostaLookMac()
{
		lookandfeelCorrente = "Mac";
		initLookAndFeel(lookandfeelCorrente);
		salva_ImpostazioniDefault();
		stato.setText("Nuovo lookandfeel impostato: Mac, riavviare l'applicazione.");
		azioneRiavviaEditor();
}


// **** azioneGuida mostra una breve guida testuale con informazioni relative all'installazione ****

private void azioneGuida()
{
		Guida finestraguida = new Guida(this); // crea una dialog Window passando il frame corrente come owner
}


// **** azioneAboutBox e' richiamato per mostrare la finestra "About" relativa al programma ****

private void azioneAboutBox()
{
		AboutBox finestraAboutBox = new AboutBox(this);
}


// **** azioneRiavviaEditor distrugge l'attuale oggetto editor e ne crea uno nuovo con le  impostazioni correnti ****

private void azioneRiavviaEditor()
{
		int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera riavviare l'editor per rendere effettive le nuove impostazioni'
		if (ritornoValore == JOptionPane.YES_OPTION)
		{
```

```
                dispose();
                XMLStudio XMLStudio = new XMLStudio();
                XMLStudio.setVisible(true);
        }
}


// **** segnala se il documento xml attivo e' o meno ben formato secondo le specifiche xml  ****

private boolean azioneBenFormatoXml()
{
        boolean benformato = false;

        if (fileCorrente.exists())
        {
                cursoreClessidra();
                // se il file caricato non e' .xml, segnala che non puoi verificare che il documento e' ben formato
                String nomefile = fileCorrente.getName();
                String estensione = nomefile.substring(nomefile.length()-4);
                if (estensione.equals(".xml") == false) // l'estensione del file corrente non e' .xml
                {
                        JOptionPane.showInternalMessageDialog(getContentPane(), "Il file aperto non e' .xml, per cui  non e' possibile verificare c
                }
                else
                {
                        try
                        {
                                int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera sovrascrivere il file " + fil
                                if (ritornoValore == JOptionPane.YES_OPTION)
                                        fileGiaSalvato = true; // e' stato selezionato di sovrascrivere il file
                                if (ritornoValore == JOptionPane.NO_OPTION)
                                        fileGiaSalvato = false; // e' stato selezionato di non sovrascrivere il file

                                azioneSalvaFile(); // prima di creare l'albero xml e' necessario salvare il file

                                // DocumentBuilderFactory definisce API che consentono ad applicazioni di ottenere un parser che produce d
                                DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance(); // crea una nuova ista
                                // docBuilderFactory.setValidating(false); // estensione di XMLStudio: per verificare anche la validita', impos
                                // DocumentBuilder e' usato per ottenere un'istanza di documento DOM da un documento XML
                                DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder(); // crea una nuova istanza di Docun

                                try
                                {
                                        documentoXml = docBuilder.parse(fileCorrente); // parsifica il contenuto di un dato file come un d
                                        JOptionPane.showInternalMessageDialog(getContentPane(), "Il documento " + fileCorrente.getNa
                                        stato.setText("Il documento " + fileCorrente.getName() + " e' ben formato.");
                                        benformato = true;
                                }
                                catch(SAXException e)
                                {
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Il documen
                                        // JOptionPane.showInternalMessageDialog(getContentPane(), "Il documento" + fileCorrente.get
                                        stato.setText("Il documento " + fileCorrente.getName() + " non e' ben formato.");
                                }
                        }
                        catch (Exception e)
                        {
                                JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Eccezione rilevata nell
                                // e.printStackTrace();
                        }
```

```
                        }
                        cursoreNormale();
                }
                else // se fileCorrente non esiste
                {
                        JOptionPane.showInternalMessageDialog(getContentPane(), "Per verificare che un documento xml sia ben formato, e' prima necessa
                }
                return benformato; // se non ha gia' restituito return in altri sottomenu
}


// **** crea o aggiorna l'albero che rappresenta l'attuale documento Xml ****

private void azioneCreaAlberoXmlVuoto()
{
                // System.out.println("azioneCreaAlberoXml: " + fileCorrente.getName());
                try
                {
                        DefaultMutableTreeNode nodoRadice = new DefaultMutableTreeNode("Albero xml vuoto");
                        modelloAlbero = new DefaultTreeModel(nodoRadice); // crea un albero, passandovi la radice, in cui ogni nodo ha dei figli
                        alberoXml.setModel(modelloAlbero);
                        alberoXml.treeDidChange(); // impostato quando il JTree e' cambiato a tal punto da necessitare il ridimensionamento
                        espandiAlbero(alberoXml); // espande l'albero JTree legato al documento Xml affinche' sia visibile nel viewer
                        aggiornaAttributiNodoInTavola(null); // imposta a null gli attributi nella tabella dei nodi
                        // stato.setText("Albero relativo al documento " + fileCorrente.getName() + " non e' aggiornato.");
                }
                catch (Exception e)
                {
                        // System.out.println("Eccezione rilevata nella creazione dell'albero: " + e);
                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Eccezione rilevata nella creazione dell'albero: "
                        // e.printStackTrace();
                }
}


// **** crea o aggiorna l'albero che rappresenta l'attuale documento Xml ****

private void azioneCreaAlberoXml()
{
                //System.out.println("azioneCreaAlberoXml: " + fileCorrente.getName());
                if (fileCorrente.exists())
                {
                        // se il file caricato non e' .xml, segnala che non puoi creare l'albero xml
                        String nomefile = fileCorrente.getName();
                        String estensione = nomefile.substring(nomefile.length()-4);
                        if (estensione.equals(".xml") == false) // l'estensione del file corrente non e' .xml

                        {
                                JOptionPane.showInternalMessageDialog(getContentPane(), "Il file aperto non e' .xml, per cui  non e' possibile creare l'alt
                        }
                        else
                        {
                                try
                                {
                                        int ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera sovrascrivere il file " + fil
                                        if (ritornoValore == JOptionPane.YES_OPTION)
                                                fileGiaSalvato = true; // e' stato selezionato di sovrascrivere il file
                                        if (ritornoValore == JOptionPane.NO_OPTION)
                                                fileGiaSalvato = false; // e' stato selezionato di non sovrascrivere il file
```

```
                                        azioneSalvaFile(); // prima di creare l'albero xml e' necessario salvare il file

                                        // DocumentBuilderFactory definisce API che consentono ad applicazioni di ottenere un parser che produce d
                                        DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance(); // crea una nuova ista
                                        // DocumentBuilder e' usato per ottenere un'istanza di documento DOM da un documento XML
                                        DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder(); // crea una nuova istanza di Docun

                                        try
                                        {
                                                cursoreClessidra();
                                                documentoXml = docBuilder.parse(fileCorrente); // parsifica il contenuto di un dato file come un d
                                                // System.out.println(fileCorrente.getName() + " e' xml ben formato.");

                                                Element nodoRadice = documentoXml.getDocumentElement();  // individua il nodo radice dal docu
                                                nodoRadice.normalize(); // assicura che la vista del documento DOM sia la stessa come se fosse

                                                DefaultMutableTreeNode topAlbero = creaAlberoNodi(nodoRadice); // passando nodoRadice crea

                                                modelloAlbero.setRoot(topAlbero); // imposta topAlbero come radice
                                                alberoXml.treeDidChange(); // impostato quando il JTree e' cambiato a tal punto da necessitare il
                                                espandiAlbero(alberoXml); // espande l'albero JTree legato al documento Xml affinche' sia visibile
                                                aggiornaAttributiNodoInTavola(null); // imposta a null gli attributi nella tabella dei nodi

                                                stato.setText("Creato l'albero relativo al documento ben formato " + fileCorrente.getName());
                                                cursoreNormale();
                                        }
                                        catch(SAXException e)
                                        {
                                                // System.out.println(fileCorrente.getName() + " non e' xml ben formato.");
                                                // JOptionPane.showInternalMessageDialog(getContentPane(), "Il documento " + fileCorrente.get
                                                JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Il documen
                                                stato.setText("Il documento " + fileCorrente.getName() + " non e' xml ben formato.");
                                                cursoreNormale();
                                        }
                                }
                                catch (Exception e)
                                {
                                        // System.out.println("Eccezione rilevata nella creazione dell'albero: " + e);
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Eccezione rilevata nella
                                        cursoreNormale();
                                        // e.printStackTrace();
                                }
                        }
                }
                else // se fileCorrente non esiste
                {
                        JOptionPane.showInternalMessageDialog(getContentPane(), "Per creare l'albero xml e' prima necessario caricare un file .xml.", "Inf
                }
}


// **** dato il nodo radice crea l'oggetto alberoNodi di tipo XmlNodo che estende DefaultMutableTreeNode ****

private DefaultMutableTreeNode creaAlberoNodi(Node nodoRadice) // dato un nodo crea la gerarchia di nodi ad alber
{
        if (!possoVisualizzareNodo(nodoRadice)) // se non puo' visualizzare la radice, ritorna null
        {
                stato.setText("Analisi del file xml: non posso visualizzare la radice.");
                return null;
        }
```

```
		XmlNodo alberoNodi = new XmlNodo(nodoRadice); // crea un oggetto treeNode di tipo XmlNodo per rappresentare la radice

		NodeList listaNodi = nodoRadice.getChildNodes(); // listaNodi e' una collezione ordinata di nodi: rappresenta tutti i nodi figli del nodo radice

		// per ogni nodo figlio viene creato un albero di nodi richiamando ricorsivamente creaAlberoNodi
		for (int k = 0; k < listaNodi.getLength(); k++) // getLength restituisce il numero di nodi nella lista
		{
			Node nodo = listaNodi.item(k); // ritorna il k-esimo nodo selezionato
			DefaultMutableTreeNode nodoFiglio = creaAlberoNodi(nodo); // chiamata ricorsiva del metodo passandogli il nodo e assumendolo cor
			if (nodoFiglio != null)
				alberoNodi.add(nodoFiglio); // aggiunge all'oggetto treeNode il nodo figlio esaminato
		}
		return alberoNodi; // alla fine restituisci l'oggetto treeNode creato
}


// **** metodo responsabile di espandere ogni nodo a partire dalla radice affinche' sia visibile l'albero nel viewer ****

private static void espandiAlbero(JTree alberoJTree) // riceve come parametro alberoXml da espandere
{
  // oggettoRadice e' nodo dell'albero in un JTree
  // getModel restituisce il modello di tipo TreeModel usato per i dati
  // getRoot restituisce il nodo radice
  TreeNode nodoRadice = (TreeNode)alberoJTree.getModel().getRoot(); // individua il nodo radice di questo albero
  TreePath path = new TreePath(nodoRadice); // individua il path della radice di questo albero
  for (int k = 0; k < nodoRadice.getChildCount(); k++) // per ogni nodo figlio della radice fai
  {
		TreeNode nodoFiglio = (TreeNode)nodoRadice.getChildAt(k); // individua il k-esimo nodo figlio della radice
		espandiAlbero(alberoJTree, path, nodoFiglio); // espandi l'albero dati il Jtree, il path della radice e il nodoFiglio
  }
}


// **** metodo responsabile di espandere ogni nodo affinche' sia visibile l'albero nel viewer ****

private static void espandiAlbero(JTree alberoJTree, TreePath path, TreeNode nodo)
{
		if (path == null || nodo == null) // se il path del nodo radice e' nullo o il nodo radice e' nullo
			return;                                                  // esci dal metodo
		alberoJTree.expandPath(path); // dato il path, assicura che il nodo identificato dal path specificato sia espanso e visibile
		TreePath nuovoPath = path.pathByAddingChild(nodo); // restituisce un nuovo path contenente tutti gli elementi del nodo padre e del nodo figlio
		for (int k = 0; k < nodo.getChildCount(); k++) // per ogni nodo figlio di nodo (che e' gia' un nodo interno dell'albero) fai
		{
			TreeNode nodoFiglio = (TreeNode)nodo.getChildAt(k); // individua il k-esimo nodo figlio del nodo considerato
			if (nodoFiglio != null) // se nodoFiglio individuato non e' null allora espandi quella porzione dell'albero
			{
				espandiAlbero(alberoJTree, nuovoPath, nodoFiglio); // alberoJTree, il path del nodo padre, il nodo figlio
			}
		}
}


// **** verifica se il nodo (radice) e' un elemento o un nodo testuale ****

private boolean possoVisualizzareNodo(Node nodocorrente) // se il nodo non e' elemento o testuale non viene visualizzato nell'albero
{
		switch (nodocorrente.getNodeType())  // estrapola il tipo del nodo attualmente esaminato
		{
			case Node.ELEMENT_NODE: // il nodo e' di tipo elemento nel file xml
```

```
                        return true;
                case Node.TEXT_NODE: // il nodo e' di tipo testuale
                        String testonodo = nodocorrente.getNodeValue().trim(); // essendo il nodo testuale, preleva il suo valore ovvero del test
                        return !(testonodo.equals("") || testonodo.equals("\n") || testonodo.equals("\r\n")); // ritorna true o false dopo i confr
        }
        return false;
}


// **** restituisce il valore della variabile fileXmlTagCorrente ****

public String getfileXmlTagCorrente()
{
        return fileXmlTagCorrente; // restituisce il valore della variabile fileXmlTagCorrente
}


// **** metodo che consente l'inserimento dei tag xml all'interno del documento attivo ****

private void azioneAggiungiTagXml()
{
        Node nodoSelezionato; // nodo attualmente selezionato
        String testoNodo = null, // contiene il testo del nodo selezionato
            testoAttributi = null, // contiene la sequenza di attributi relativi al nodo selezionato
            testoTagXml1 = null, // contiene il testo relativo al tag comprendente anche gli attributi
            testoTagXml2 = null, // contiene il testo relativo alla chiusura del tag
            testoTagXmlCompleto = null; // contiene la selezione, i due tag xml ed eventuali attributi

        try
        {
                nodoSelezionato = oggettoXmlTag.restituisciNodoSelezionato(); // in nodoselezionato viene posto il nodo selezionato nell'oggetto Xml

                switch (nodoSelezionato.getNodeType())  // estrapola il tipo del nodo dal nodo attualmente esaminato
                {
                        case Node.ELEMENT_NODE: // il nodo e' di tipo elemento nel file xml
                        {
                                testoNodo = nodoSelezionato.getNodeName();

                                int[] rigaSelezionata = oggettoXmlTag.tavolaAttributiXmlTag.getSelectedRows(); // aggiunge in testoAttributi
                                for (int i=0; i<rigaSelezionata.length; i++)
                                {
                                        if (testoAttributi == null)
                                                testoAttributi = " " + (oggettoXmlTag.tavolaAttributiXmlTag.getValueAt(rigaSelezionata
                                        else
                                                testoAttributi = testoAttributi + " " + (oggettoXmlTag.tavolaAttributiXmlTag.getValueAt
                                }

                                if (testoAttributi == null)
                                {
                                        testoTagXml1 = "<" + testoNodo + ">";
                                        testoTagXml2 = "</" + testoNodo + ">";
                                }
                                else
                                {
                                        testoTagXml1 = "<" + testoNodo + testoAttributi + ">";
                                        testoTagXml2 = "</" + testoNodo + ">";
                                }
                                break;
                        }
                        case Node.TEXT_NODE: // il nodo e' di tipo testuale
```

```
                                {
                                        testoNodo = nodoSelezionato.getNodeValue(); // essendo il nodo testuale, preleva il suo valore ovvero del tes
                                        testoTagXml1 = "<" + testoNodo + ">";
                                        testoTagXml2 = "</" + testoNodo + ">";
                                        break;
                                }

                        }
                        testoTagXmlCompleto = testoTagXml1 + (areaTesto.getSelectedText()) + testoTagXml2;
                        areaTesto.replaceSelection(testoTagXmlCompleto);
                        stato.setText("Inserito tag " + testoTagXml1);
                }
                catch (Exception e)
                {
                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(this, "Informazione", "Eccezione rilevata nell'aggiunta del tag xml: " +
                        stato.setText("Rilevata eccezione nell'aggiunta del tag xml.");
                }

}


// **** metodo che consente la rimozione dei tag xml dal testo selezionato ****

private void azioneRimuoviTagXml()
{
        String testoSelezionato;
        String[] testoArray;

        testoSelezionato = areaTesto.getSelectedText();
        if (testoSelezionato == null)
        {
                JOptionPane.showInternalMessageDialog(getContentPane(), "Per rimuovere i tag e' necessario selezionare un paragrafo che li includ
        }
        else
        {
                testoArray = testoSelezionato.split("</"); // suddivide il testo in antecedente e posteriore al carattere </
                testoSelezionato = testoArray[0];
                testoArray = testoSelezionato.split(">"); // suddivide il testo in antecedente e posteriore al carattere >
                areaTesto.replaceSelection(testoArray[1]);
                stato.setText("Tag rimossi dal paragrafo.");
        }
}


// **** restituisce true se la stringa testo termina con contiene ****

private boolean selezioneContieneFine(String testo, String contiene)
{
        if (testo != null)
                return testo.endsWith(contiene);
        else
                return false;
}


// **** restituisce true se la stringa testo inizia con contiene ****

private boolean selezioneContieneInizio(String testo, String contiene)
{
        if (testo != null)
```

```
                return testo.startsWith(contiene);
        else
                return false;
}


// **** metodo che consente la selezione di un paragrafo compreso tra due tag xml ****

private void azioneSelezionaTagXml()
{
        String testo; // include la porzione di testo da esaminare
        boolean uscita = false; // consente di uscire dal ciclo while
        int posizioneDot = areaTesto.getCaret().getDot(); // rileva la posizione corrente del cursore
        int salvaPosizioneDotInizio, salvaPosizioneDotFine, salvaPosizioneDotCentrale;

        cursoreClessidra();
        areaTesto.requestFocus(); // richiama il focus sulla JTextArea areaTesto
        salvaPosizioneDotCentrale = posizioneDot; // salva la posizione dove l'utente ha posizionato il cursore

        while (!uscita)
        {
                areaTesto.select(areaTesto.getCaret().getMark(), areaTesto.getCaret().getDot());
                testo = areaTesto.getSelectedText();
                if (selezioneContieneFine(testo, ">") == true) // verifica se e' stato raggiunto l'elemento >
                {
                        uscita = true;
                }
                else
                {
                        posizioneDot = posizioneDot + 1; // sposta il cursore avanti di una posizione
                        areaTesto.getCaret().moveDot(posizioneDot); // realizza lo spostamento del cursore
                }
        }
        salvaPosizioneDotFine = posizioneDot; // individuato l'estremo destro della selezione

        areaTesto.getCaret().setDot(salvaPosizioneDotCentrale);
        posizioneDot = salvaPosizioneDotCentrale;
        uscita = false;
        while (!uscita)
        {
                areaTesto.select(areaTesto.getCaret().getDot(), areaTesto.getCaret().getMark());
                testo = areaTesto.getSelectedText();
                if (selezioneContieneInizio(testo, "<") == true) // verifica se e' stato raggiunto l'elemento <
                {
                        uscita = true;
                }
                else
                {
                        posizioneDot = posizioneDot - 1; // sposta il cursore indietro di una posizione
                        areaTesto.getCaret().moveDot(posizioneDot); // realizza lo spostamento del cursore
                }
        }
        salvaPosizioneDotInizio = posizioneDot; // individuato l'estremo sinistro della selezione
        areaTesto.select(salvaPosizioneDotInizio, salvaPosizioneDotFine); // seleziona il testo
        areaTesto.getCaret().setSelectionVisible(true); // rendi visibile la selezione effettuata
        cursoreNormale();
}


// **** crea un documento .dtd a partire dal file documento .xml attivo ****
```

```
private boolean azioneCreaDtd()
{
	boolean creadtd = false;

	// determina il nome del file DTD
	String nomefilexml = fileCorrente.getPath(); // nome del file xml caricato
	String estensione = nomefilexml.substring(nomefilexml.length()-4); // estrae l'estensione del file
	if (estensione.equals(".xml") == true) // verifica che l'estensione del file caricato sia xml
	{
		String nomefile = nomefilexml.substring(0, nomefilexml.length()-4); // nomefile contiene il path e il nome del file DTD
		String nomefileDTD = nomefile + ".dtd";

		if (azioneBenFormatoXml()) // verifica che il documento Xml caricato sia ben formato
		{
			try
			{
				DTDGenerator oggettoCreaDTD = new DTDGenerator(); // o getAbsolutePath()
				oggettoCreaDTD.run(fileCorrente.getPath());
				oggettoCreaDTD.printDTD(nomefileDTD);
				JOptionPane.showInternalMessageDialog(getContentPane(), "Il file DTD " + nomefileDTD + " e' stato creato.",
				stato.setText("File DTD " + nomefileDTD + " creato.");
				creadtd = true;
			}
			catch (Exception e)
			{
				JOptionPane.showInternalMessageDialog(getContentPane(), "Errore rilevato durante la creazione del file DTD " + nomefil
				stato.setText("File DTD non creato.");
			}
	}
  }
  else
  {
	JOptionPane.showInternalMessageDialog(getContentPane(), "Per creare il file DTD e' prima necessario aprire un file XML ben formato.", "Inform
	stato.setText("File xml per creare il DTD non caricato.");
  }
	return creadtd;
}


// **** crea, a partire dal documento xml corrente, un documento xml valido (che include il dtd associato) ****

private void azioneCreaDocumentoXmlValido()
{
	if (azioneCreaDtd()) // crea il DTD associato al documento XML
	{
		try
		{
			String nomefilexml = fileCorrente.getName(); // nome del file xml caricato
			String nomefile = nomefilexml.substring(0, nomefilexml.length()-4); // nomefile contiene il path e il nome del file DTD
			String nomefileDTD = nomefile + ".dtd";

			String stringa = "<?xml version=\"1.0\" encoding=\"iso-8859-1\" standalone=\"no\"?>\n<!DOCTYPE documento SYST
			areaTesto.replaceRange(stringa, 0, areaTesto.getLineEndOffset(1));

			JOptionPane.showInternalMessageDialog(getContentPane(), "Il documento " + fileCorrente.getName() + " e' ora valido.",
			stato.setText("Il documento " + fileCorrente.getName() + " e' valido.");
		}
		catch (Exception e)
		{
```

```
                                JOptionPane.showInternalMessageDialog(getContentPane(), "Errore rilevato durante la creazione del documento valido "
                                stato.setText("Documento xml valido " + fileCorrente.getName() + " non creato.");
                        }
                }
}


// **** nella creazione del foglio di stile, selezionato un nodo, ne restituisce il testo ****

private String restituisciNodoperCSS()
{
        Node nodoSelezionato; // nodo attualmente selezionato
        String testoNodo = null; // contiene il testo del nodo selezionato
        boolean uscita = false; // permette da uscire dal ciclo while

        try
        {
                while (!uscita)
                {
                        JOptionPane.showInternalMessageDialog(getContentPane(), "Selezionare un tag per la sua colorazione nel CSS.", "Inforn
                        nodoSelezionato = oggettoXmlTag.restituisciNodoSelezionato(); // in nodoselezionato viene posto il nodo selezionato nell'

                        switch (nodoSelezionato.getNodeType()) // estrapola il tipo del nodo dal nodo attualmente esaminato
                        {
                                case Node.ELEMENT_NODE: // il nodo e' di tipo elemento nel file xml
                                {
                                        testoNodo = nodoSelezionato.getNodeName();
                                        uscita = true;
                                        break;
                                }
                                case Node.TEXT_NODE: // il nodo e' di tipo testuale
                                {
                                        // testoNodo = nodoSelezionato.getNodeValue(); // essendo il nodo testuale, preleva il suo valor
                                        JOptionPane.showInternalMessageDialog(getContentPane(), "E' necessario selezionare il nodo e n
                                        break;
                                }
                        }
                }
        } catch (Exception e)
        {
                JOptionPane.showInternalMessageDialog(getContentPane(), "Errore durante la selezione del nodo nella creazione del file CSS.", "Info
        }
        return testoNodo;
}


// **** crea il foglio di stile CSS ****

private void elaboraSalvaCSS(File fileCss)
{
        boolean uscita = false; // permette di uscire dal ciclo while
        int ritornoValore; // valore restituito per le domande
        String valoreRosso, valoreVerde, valoreBlu, valoreRGB, nodoInCSS; // stringhe relativi ai colori in formato RGB e al testo del nodo selezionato

        try
        {
                BufferedWriter out = new BufferedWriter(new FileWriter(fileCss)); // crea il fileCSS
                out.write("documento { color : #000000; font-family : Arial Black; }"); // inizia a scrivere l'intestazione di default nel file
                out.newLine(); // scrivi una nuova linea
```

```
                        while (!uscita)
                        {
                                coloreTestoDefault = JColorChooser.showDialog(this, "Scegliere colore per testo individuato dal tag", Color.black);

                                //String valore = int2hexstr(coloreTestoDefault.getRGB());
                                valoreRosso = Integer.toHexString(coloreTestoDefault.getRed());
                                if (valoreRosso.length() == 1)
                                        valoreRosso += "0";
                                valoreVerde = Integer.toHexString(coloreTestoDefault.getGreen());
                                if (valoreVerde.length() == 1)
                                        valoreVerde += "0";
                                valoreBlu = Integer.toHexString(coloreTestoDefault.getBlue());
                                if (valoreBlu.length() == 1)
                                        valoreBlu += "0";

                                valoreRGB = valoreRosso + valoreVerde + valoreBlu;

                                nodoInCSS = restituisciNodoperCSS();

                                out.write(nodoInCSS + " { color : #" + valoreRGB + "; border : solid 1px; font-family : Arial Black; padding: 4px; width: 100%
                                out.newLine(); // scrivi una nuova linea

                                ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera memorizzare un altro colore per tag sul
                                if (ritornoValore == JOptionPane.NO_OPTION)
                                        uscita = true;
                         }
                         out.close( );
                } catch (Exception e)
                {
                        JOptionPane.showInternalMessageDialog(getContentPane(), "Errore durante l'elaborazione dei colori nella creazione del file CSS.", "I
                }
}


// **** crea e salva il foglio di stile Css ****

private void azioneCreaCss()
{
         boolean salvareFile = false;

        JFileChooser fileChooser = new JFileChooser(directoryCorrente); // crea l'oggetto per l'apertura del file sulla directory corrente
        // imposta la corrente directory di lavoro, esempio /Tesi/Tesi XMLStudio oppure "."

        fileChooser.setApproveButtonText("Salva"); // imposta il valore del bottone di esecuzione
        fileChooser.setDialogTitle("Salva file css"); // imposta il titolo della finestra Dialog

        // imposta ed aggiunge i filtri opportuni
        fileChooser.resetChoosableFileFilters(); // reinizializza l'oggetto fileChooser
        FiltroFile cssFiltro = new FiltroFile("css", "File css *.CSS"); // imposta il filtro per i file .css
        fileChooser.addChoosableFileFilter(cssFiltro); // aggiunge il filtro alla finestra di apertura
        fileChooser.setFileFilter(cssFiltro); // imposta il filtro xmlFiltro come filtro di default

        int ritornoValore = fileChooser.showSaveDialog(getContentPane()); // mostra la finestra dialog per il salvataggio del file

        if (ritornoValore == JFileChooser.APPROVE_OPTION)
        {
                try
                {
                        File f = fileChooser.getSelectedFile(); // associa all'oggetto f il file selezionato
                        if (f.exists()) // il file selezionato esiste gia' nella directory corrente
```

```
                                        {
                                                ritornoValore = JOptionPane.showConfirmDialog(getContentPane(), "Si desidera sovrascrivere il file " + f.getI
                                                        if (ritornoValore == JOptionPane.YES_OPTION)
                                                salvareFile = true; // e' stato selezionato di sovrascrivere il file
                                                        if (ritornoValore == JOptionPane.NO_OPTION)
                                                salvareFile = false; // e' stato selezionato di non sovrascrivere il file
                                        }
                                        else
                                        {
                                                salvareFile = true; // il file non esiste nella directory corrente e viene salvato
                                        }

                                        if (salvareFile == true) // gestisce il salvataggio del file
                                        {
                                                elaboraSalvaCSS(f);

                                                aggiungiIntestazioneCSS(f.getName());

                                                stato.setText("File " + f.getName() + " salvato e intestazione aggiornata."); // indica l'avvenuto caricamento

                                                //Runtime.getRuntime().exec("cmd /c start "+fileCorrente.getPath());
                                        }
                                        else // non salva il file
                                        {
                                                stato.setText("File " + f.getName() + " non salvato."); // indica l'avvenuto caricamento
                                        }
                                }
                                catch(Exception e)
                                {
                                        stato.setText("Errore di salvataggio del file " + fileCorrente.getName() + ".");
                                        System.err.println("Errore di salvataggio del file " + fileCorrente.getName() + ".");
                                }
                        }
                }


// **** aggiunge l'intestazione relativa al file CSS elaborato ****

private void aggiungiIntestazioneCSS(String nomeFileCSS)
{
        String testoCSS = "<?xml-stylesheet type=\"text/css\" href=\"" + nomeFileCSS + "\"?>\n<documento>\n";

        String testo = areaTesto.getText();
        int posizione = testo.indexOf("<documento>"); // ricerca in testoCompleto la stringa parola a partire da posizioneIniz
        areaTesto.select(posizione, posizione + 11); // seleziona il testo individuato tra le specifiche locazioni
        areaTesto.replaceSelection(testoCSS);
}


// **** metodo main del programma ****

public static void main(String[] args)
{
        // verifica della versione delle librerie Swing installate sulla macchina

        try
        {
                String versione = System.getProperty("java.version");

                if (versione.compareTo("1.1.2") < 0)
```

```
                        {
                                System.out.println("Attenzione: Le librerie Swing devono essere eseguite con una versione 1.1.2 o superiore di Virtual Mac
                        }
                } catch (Throwable t)
                {
                        System.out.println("Errore, eccezione rilevata: " + t);
                        t.printStackTrace();
                }

                // creazione dell'oggetto editor XMLStudio

                XMLStudio XMLStudio = new XMLStudio();
                XMLStudio.setVisible(true);
        }
}
```

Listato XmlTag.java

```
/* ***************************************************************************

   · XMLStudio - Editor integrato per documenti XML
   · Programma per la Tesi di Laurea di Paolo Guagliumi - pguagliumi@libero.it
   · Docente relatore: Prof.ssa Paola Giannini - giannini@di.unito.it

   · XmlTag.java - Definizione classe per l'implementazione dell'oggetto XmlTag
   (C) 2002 Paolo Guagliumi - Il programma adotta la licenza GPL allegata

   *  ================================================= *
   *  This program contains code from the book "Swing"    *
   *  2nd Edition by Matthew Robinson and Pavel Vorobiev  *
   *  http://www.spindoczine.com/sbe                      *
   *  ================================================= *

  *************************************************************************** */


// **** importa i packages utili ****

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.event.*;

import javax.xml.parsers.*;
import org.w3c.dom.*;

/* **** questa classe consente di definire l'oggetto costituito da un JTree e da una JTable
      che permette l'inserimento dei tag all'interno dei documenti testuali **** */

class XmlTag extends JPanel implements ActionListener
{
            protected JButton nuovotag, apritag, salvatag, salvanometag, aggiunginodoxmltag, editanodoxmltag,
            cancellanodoxmltag, aggiungiattributonodoxmltag, modificaattributonodoxmltag, cancellaattributonodoxmltag,
            cancellaSelezioneJTable; // definizione pulsanti della ToolBar

            public static final String nomeClasseXmlTag = "Tag in formato Xml";

            protected Document documentoXmlTag; // interfaccia che rappresenta l'intero documento xml tag
            protected File fileXmlTagCorrente; // file attualmente visualizzato nell'albero dei tag
            protected JTree alberoXmlTag; // albero che mostra i tag selezionabili dall'utente
            protected DefaultTreeModel modelloAlberoXmlTag; // implementa l'interfaccia TreeModel

            protected DefaultTreeCellEditor editorCellaAlbero; // editor di celle dell'albero per consentire l'editing di nodi di testo
            protected Node nodoAttualmenteEditato = null; // rappresenta il nodo attualmente in fase di editing da parte dell'utente

             protected JTable tavolaAttributiXmlTag; // definisce la JTable per gli attributi dell'albero di parsing
             protected TavolaAttributi modelloTavolaAttributiXmlTag; // definisce un'estensione di AbstractTableModel per implementare la JTable

             XMLStudio oggettoXMLStudioGlobale;
```

```
// **** crea la ToolBar relativa all'oggetto XmlTag ****

public JToolBar creaToolBar()
{
		JToolBar barraTag = new JToolBar();
		// barraTag.setFloatable(false);

		nuovotag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/nuovo.gif")));
		barraTag.add(nuovotag);
		nuovotag.addActionListener(this);
		nuovotag.setToolTipText("Crea un nuovo file xml relativo ai tag");

		apritag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/apri.gif")));
		barraTag.add(apritag);
		apritag.addActionListener(this);
		apritag.setToolTipText("Apre un nuovo file xml relativo ai tag");

		salvatag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/salva.gif")));
		barraTag.add(salvatag);
		salvatag.addActionListener(this);
		salvatag.setToolTipText("Salva il file xml relativo ai tag");

		salvanometag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/salvanome.gif")));
		barraTag.add(salvanometag);
		salvanometag.addActionListener(this);
		salvanometag.setToolTipText("Salva il file xml relativo ai tag indicandone il nome");

		barraTag.addSeparator();

		barraTag.add(new JLabel("Nodo "));
		aggiunginodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
		barraTag.add(aggiunginodoxmltag);
		aggiunginodoxmltag.addActionListener(this);
		aggiunginodoxmltag.setToolTipText("Aggiungi un nuovo tag xml");

		editanodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));

		barraTag.add(editanodoxmltag);
		editanodoxmltag.addActionListener(this);
		editanodoxmltag.setToolTipText("Modifica il nodo tag xml");

		cancellanodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancella.gif")));
		barraTag.add(cancellanodoxmltag);
		cancellanodoxmltag.addActionListener(this);
		cancellanodoxmltag.setToolTipText("Cancella il nodo tag xml");

		barraTag.addSeparator();

		barraTag.add(new JLabel("Attributi "));
		aggiungiattributonodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/creaxml.gif")));
		barraTag.add(aggiungiattributonodoxmltag);
		aggiungiattributonodoxmltag.addActionListener(this);
		aggiungiattributonodoxmltag.setToolTipText("Aggiunge un nuovo attributo al nodo tag xml");

		modificaattributonodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/tag.gif")));
		barraTag.add(modificaattributonodoxmltag);
		modificaattributonodoxmltag.addActionListener(this);
		modificaattributonodoxmltag.setToolTipText("Modifica il valore dell'attributo del nodo tag xml");
```

```
		cancellaattributonodoxmltag = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancella.gif")));
		barraTag.add(cancellaattributonodoxmltag);
		cancellaattributonodoxmltag.addActionListener(this);
		cancellaattributonodoxmltag.setToolTipText("Cancella l'attributo del nodo tag xml");

		cancellaSelezioneJTable = new JButton(new ImageIcon(ClassLoader.getSystemResource("./risorse/cancellatable.gif")));
		barraTag.add(cancellaSelezioneJTable);
		cancellaSelezioneJTable.addActionListener(this);
		cancellaSelezioneJTable.setToolTipText("Deseleziona le celle degli attributi e valori evidenziati");

		return barraTag;
}


// **** metodi invocati quando avviene un action event ****

public void actionPerformed(ActionEvent e)
{
		String evento = e.getActionCommand();

		if (e.getSource() == nuovotag)
			azioneNuovoDocumentoXmlTag();
		if (e.getSource() == apritag)
			azioneApriDocumentoXmlTag();
		if (e.getSource() == salvatag)
			azioneSalvaDocumentoXmlTag(false);
		if (e.getSource() == salvanometag)
			azioneSalvaDocumentoXmlTag(true);

		if (e.getSource() == aggiunginodoxmltag)
			azioneAggiungiNodoXmlTag();
		if (e.getSource() == editanodoxmltag)
			azioneEditaNodoXmlTag();
		if (e.getSource() == cancellanodoxmltag)
			azioneCancellaNodo();
		if (e.getSource() == aggiungiattributonodoxmltag)
			azioneAggiungiAttributoNodo();
		if (e.getSource() == modificaattributonodoxmltag)
			azioneModificaAttributoNodo();
		if (e.getSource() == cancellaattributonodoxmltag)
			azioneEliminaAttributoNodo();

		if (e.getSource() == cancellaSelezioneJTable)
			azioneCancellaSelezioneJTable();
}


// **** costruttore della classe ****

public XmlTag(XMLStudio oggettoXMLStudio) //final String fileXmlTagCorrente
{
		// System.out.println("Valore fileXmlTagCorrente:" + oggettoXMLStudio.getfileXmlTagCorrente());

		this.setLayout(new BorderLayout()); // imposta il layout dell'oggetto XmlTag

		JToolBar barraTag = creaToolBar(); // crea la JToolBar relativa all'oggetto XmlTag
		this.add(barraTag, BorderLayout.NORTH); // posiziona la JToolBar

		// crea un nodo generico che non ha ne' padre, ne' figli, ma che consente di avere figli
```

```
DefaultMutableTreeNode nodoRadiceXmlTag = new DefaultMutableTreeNode("Nessun tag xml");
// crea un albero, passandovi la radice, in cui ogni nodo ha dei figli
modelloAlberoXmlTag = new DefaultTreeModel(nodoRadiceXmlTag);
// restituisce un'istanza di JTree che mostra il nodo radice e l'albero e' creato usando uno specifico modello dati
alberoXmlTag = new JTree(modelloAlberoXmlTag);
// imposta la selezione dell'albero a SINGLE_TREE_SELECTION: consente di selezionare solo un nodo dell'albero per volta
alberoXmlTag.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
alberoXmlTag.setShowsRootHandles(true); // gli handles sono mostrati
//alberoXmlTag.setEditable(false); // imposta l'albero JTree come non editabile

RenderAlbero render = new RenderAlbero(); // definisce un nuovo oggetto render di tipo RenderAlbero
 alberoXmlTag.setCellRenderer(render); // applica all'albero JTree il render con le associazioni di icone e tooltip relative

 // **** definizione di una sottoclasse editor di celle personalizzato per consentire l'editing dei nodi **** ooo
  // costruisce un oggetto DefaultTreeCellEditor per un JTree utilizzando lo specifico renderer applicato all'alberoXmlTag

 editorCellaAlbero = new DefaultTreeCellEditor(alberoXmlTag, render)
{
        // il realeditor che gestisce l'editing ritorna true se l'editing puo' essere iniziato

        public boolean isCellEditable(EventObject event)
        {
                Node nodo = restituisciNodoSelezionato();
                if (nodo != null && nodo.getNodeType() == Node.TEXT_NODE)
                        return super.isCellEditable(event);
                else
                        return false;
        }

        // configura l'editor passandovi l'albero da editare, il valore della cella da editare, se la cella e' selezionata, se il nodo e' espanso, se

        public Component getTreeCellEditorComponent(JTree albero, Object valore, boolean selezionato, boolean espanso, boolean foglia, int r
        {
                if (valore instanceof XmlNodo)
                        nodoAttualmenteEditato = ((XmlNodo)valore).restituisciXmlNodo();
                return super.getTreeCellEditorComponent(albero, valore, selezionato, espanso, foglia, riga);
        }
};

editorCellaAlbero.addCellEditorListener(new XmlEditorRileva());
alberoXmlTag.setCellEditor(editorCellaAlbero);
alberoXmlTag.setEditable(true);
alberoXmlTag.setInvokesStopCellEditing(true);

JScrollPane scorreAlberoXmlTag = new JScrollPane(); // viene creato uno scrollpane per l'albero JTree che visualizza il documento Xml Tag
scorreAlberoXmlTag.getViewport().add(alberoXmlTag); // l'albero e' posizionato in uno scrollpane
scorreAlberoXmlTag.setToolTipText("Rappresentazione ad albero del documento tag xml");

/* modelloTavolaAttributiXmlTag e' un oggetto di tipo TavolaAttributi con campi nodoCorrente e
  attributiCorrenti ed implementa TableModel, interfaccia usata da JTable per interrogare
  un modello di dati ti tipo tabulare. Sono sufficienti due righe di codice per permettere
  alla JTable di visualizzare qualsiasi modello dati che implementi TableModel */

modelloTavolaAttributiXmlTag = new TavolaAttributi(); // nuovo oggetto TavolaAttributi che estende AbstractTableModel
tavolaAttributiXmlTag = new JTable(modelloTavolaAttributiXmlTag); // assegna alla JTable l'interfaccia specifica appena creata

JScrollPane scorreTavolaXmlTag = new JScrollPane(tavolaAttributiXmlTag); // consente lo scorrimento di tavolaAttributiXmlTag
scorreTavolaXmlTag.getViewport().setBackground(tavolaAttributiXmlTag.getBackground()); // imposta il background

JSplitPane splitAlberoXmlTag = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, scorreAlberoXmlTag, scorreTavolaXmlTag);
```

```
		splitAlberoXmlTag.setDividerLocation(500);
		//splitAlbero.setDividerSize(30);
		splitAlberoXmlTag.setContinuousLayout(true);
		splitAlberoXmlTag.setOneTouchExpandable(true); // fornisce simbolo per rapida esapansione-chiusura splitPane
		this.add(splitAlberoXmlTag, BorderLayout.CENTER);

		/* Se si e' interessati nel sapere quando la selezione cambia, bisogna implementare l'interfaccia
		  TreeSelectionListener ed aggiungere l'istanza usando il metodo addTreeSelectionListener.
		  valueChanged sara' invocato (una volta sola) quando cambia la selezione ovvero quando l'utente
		  clicca due volte consecutive sullo stesso nodo */

		// e' un listener che e' modificato quando cambia la selezione in un TreeSelectionModel*/
		TreeSelectionListener listenerAlberoXmlTag = new TreeSelectionListener() // per rilevare quando l'utente seleziona un nodo nell'albero ooo
		{
			public void valueChanged(TreeSelectionEvent e) // richiamato ogni qualvolta cambia la selezione di un valore
			{
				Node nodoselezionatoXmlTag = restituisciNodoSelezionato();

				// aggiorna l'albero con il nodo attualmente selezionato
				aggiornaAttributiNodoInTavola(nodoselezionatoXmlTag); // aggiorna l'albero xml con il nodo attualmente selezionato
				abilitaPulsantiNodi(); // rileva se abilitare o disabilitare i pulsanti dei nodi
				abilitaPulsantiAttributi(); // rileva se abilitare o disabilitare i pulsanti degli attributi
			}
		};

		ListSelectionListener listenerSelezione = new ListSelectionListener() // questo listener riceve notifica quando cambia il valore della selezione di
		{
			public void valueChanged(ListSelectionEvent e)
			{
				abilitaPulsantiAttributi(); // rileva se abilitare o disabilitare i pulsanti degli attributi
			}
		};
		tavolaAttributiXmlTag.getSelectionModel().addListSelectionListener(listenerSelezione); // aggiunge il listener alla JTable

		alberoXmlTag.addTreeSelectionListener(listenerAlberoXmlTag); // assegna all'alberoXmlTag il listener definito per eventi TreeSelection
		caricaDocumentoXmlTag(oggettoXMLStudio.getfileXmlTagCorrente()); // carica il file xml di default per i tag
		oggettoXMLStudioGlobale = oggettoXMLStudio;
}


// **** carica l'attuale nodo selezionato e lo restituisce come istanza di XmlNodo ****

public XmlNodo restituisciNodoSelezionatoAlbero()
{
		TreePath path = alberoXmlTag.getSelectionPath(); // restituisce il path del nodo selezionato grazie a getSelectionPath
		if (path == null) // restituisce null se il path ottenuto e' null
			return null;
		Object oggetto = path.getLastPathComponent(); // in oggetto restituisce l'ultimo componente che ha determinato questo path
		if (!(oggetto instanceof XmlNodo)) // se l'oggetto non e' istanza di XmlNodo ritorna null
			return null;
		return (XmlNodo)oggetto; // se l'oggetto che ha determinato l'ultimo path e' di tipo XmlNodo, restituiscilo con un casting
}


// **** restituisce l'attuale nodo selezionato richiamando restituisciNodoSelezionatoAlbero ****

public Node restituisciNodoSelezionato()
{
		XmlNodo nodoAlberoXmlTag = restituisciNodoSelezionatoAlbero(); // in nodoAlberoXml viene messo il nodo attualmente selezionato
		if (nodoAlberoXmlTag == null)
```

```
                        return null;
                return nodoAlberoXmlTag.restituisciXmlNodo(); // restituisce da un oggetto XmlNodo il nodo di tipo Node
}


// **** aggiorna l'albero xml con il nodo attualmente selezionato ****

public void aggiornaAttributiNodoInTavola(Node nodo)
{
                /* modelloTavolaAttributiXmlTag e' un oggetto di tipo TavolaAttributi con campi nodoCorrente e
                  attributiCorrenti ed implementa TableModel, interfaccia usata da JTable per interrogare
                  un modello di dati ti tipo tabulare */

                modelloTavolaAttributiXmlTag.assegnaNodo(nodo);

                /* TableModelEvent e' usato per notificare i listener che il modello della tavola e' cambiato
                  Tutte le righe contenenti dati nella tavola sono cambiate, i listener devono perdere ogni stato
                  basato sulle precedenti righe e richiedere al TableModel il nuovo numero di linee e di valori */

                tavolaAttributiXmlTag.tableChanged(new TableModelEvent(modelloTavolaAttributiXmlTag));
}


// **** dato il nodo radice crea l'oggetto alberoNodi di tipo XmlNodo che estende DefaultMutableTreeNode ****

private DefaultMutableTreeNode creaAlberoNodi(Node nodoRadice) // dato un nodo crea la gerarchia di nodi ad alber
{
                if (!possoVisualizzareNodo(nodoRadice)) // se non puo' visualizzare la radice, ritorna null
                {
                        // stato.setText("Analisi del file xml: non posso visualizzare la radice.");
                        return null;
                }

                XmlNodo alberoNodi = new XmlNodo(nodoRadice); // crea un oggetto alberoNodi di tipo XmlNodo per rappresentare la radice

                NodeList listaNodi = nodoRadice.getChildNodes(); // listaNodi e' una collezione ordinata di nodi: rappresenta tutti i nodi figli del nodo radice

                // per ogni nodo figlio viene creato un albero di nodi richiamando ricorsivamente creaAlberoNodi
                for (int k = 0; k < listaNodi.getLength(); k++) // getLength restituisce il numero di nodi nella lista
                {
                        Node nodo = listaNodi.item(k); // ritorna il k-esimo nodo selezionato
                        DefaultMutableTreeNode nodoFiglio = creaAlberoNodi(nodo); // chiamata ricorsiva del metodo passandogli il nodo e assumendolo cor
                        if (nodoFiglio != null)
                                alberoNodi.add(nodoFiglio); // aggiunge all'oggetto alberoNodi il nodo figlio esaminato
                }
                return alberoNodi; // alla fine restituisci l'oggetto alberoNodi creato
}


// **** metodo responsabile di espandere ogni nodo a partire dalla radice affinche' sia visibile l'albero nel viewer ****

private static void espandiAlbero(JTree alberoJTree) // riceve come parametro alberoXml da espandere
{
   // oggettoRadice e' nodo dell'albero in un JTree
   // getModel restituisce il modello di tipo TreeModel usato per i dati
   // getRoot restituisce il nodo radice
   TreeNode nodoRadice = (TreeNode)alberoJTree.getModel().getRoot(); // individua il nodo radice di questo albero
   TreePath path = new TreePath(nodoRadice); // individua il path della radice di questo albero
   for (int k = 0; k < nodoRadice.getChildCount(); k++) // per ogni nodo figlio della radice fai
   {
```

```
            TreeNode nodoFiglio = (TreeNode)nodoRadice.getChildAt(k); // individua il k-esimo nodo figlio della radice
            espandiAlbero(alberoJTree, path, nodoFiglio); // espandi l'albero dati il Jtree, il path della radice e il nodoFiglio
    }
}


// **** metodo responsabile di espandere ogni nodo affinche' sia visibile l'albero nel viewer ****

private static void espandiAlbero(JTree alberoJTree, TreePath path, TreeNode nodo)
{
            if (path == null || nodo == null) // se il path del nodo radice e' nullo o il nodo radice e' nullo
                        return;                                                 // esci dal metodo
            alberoJTree.expandPath(path); // dato il path, assicura che il nodo identificato dal path specificato sia espanso e visibile
            TreePath nuovoPath = path.pathByAddingChild(nodo); // restituisce un nuovo path contenente tutti gli elementi del nodo padre e del nodo figlio
            for (int k = 0; k < nodo.getChildCount(); k++) // per ogni nodo figlio di nodo (che e' gia' un nodo interno dell'albero) fai
            {
                        TreeNode nodoFiglio = (TreeNode)nodo.getChildAt(k); // individua il k-esimo nodo figlio del nodo considerato
                        if (nodoFiglio != null) // se nodoFiglio individuato non e' null allora espandi quella porzione dell'albero
                        {
                                    espandiAlbero(alberoJTree, nuovoPath, nodoFiglio); // alberoJTree, il path del nodo padre, il nodo figlio
                        }
            }
}


// **** verifica se il nodo (radice) e' un elemento o un nodo testuale ****

private boolean possoVisualizzareNodo(Node nodocorrente) // se il nodo non e' elemento o testuale non viene visualizzato nell'albero
{
            switch (nodocorrente.getNodeType())  // estrapola il tipo del nodo attualmente esaminato
            {
                        case Node.ELEMENT_NODE: // il nodo e' di tipo elemento nel file xml
                                    return true;
                        case Node.TEXT_NODE: // il nodo e' di tipo testuale
                                    String testonodo = nodocorrente.getNodeValue().trim(); // essendo il nodo testuale, preleva il suo valore ovvero del testo
                                    return !(testonodo.equals("") || testonodo.equals("\n") || testonodo.equals("\r\n")); // ritorna true o false dopo i confr
            }
            return false;
}


// **** metodo per la creazione di un nuovo documento xml che contiene i tag da inserire ****

protected void azioneNuovoDocumentoXmlTag()
{
            azioneRicordaSalva(); // chiede all'utente se salvare il file tag (xml) attualmente caricato

            String nomeNodoTag = (String)JOptionPane.showInputDialog(this, "Inserire il nome del nodo radice del nuovo documento tag (.xml)", "Inserimento
            if (nomeNodoTag == null) // se e' stato premuto il pulsante annulla, esci
                        return;
            if (nomeXmlValido(nomeNodoTag))
            {
                        oggettoXMLStudioGlobale.cursoreClessidra(); // imposta il cursore clessidra in tutta l'applicazione
                        try
                        {
                                    // DocumentBuilderFactory definisce API che consentono ad applicazioni di ottenere un parser che produce oggetti DOM
                                    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance(); // crea una nuova istanza di Docu
                                    // DocumentBuilder e' usato per ottenere un'istanza di documento DOM da un documento XML
                                    DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder(); // crea una nuova istanza di DocumentBuilder
                                    documentoXmlTag = docBuilder.newDocument(); // ottiene una nuova istanza di un oggetto document DOM per la costruz
```

```
                    Element nodoRadice = documentoXmlTag.createElement(nomeNodoTag); // crea un elemento del tipo specificato dall'inpu
                    nodoRadice.normalize(); // assicura che la vista del documento DOM sia la stessa come se fosse salvato e caricato
                    documentoXmlTag.appendChild(nodoRadice); // aggiunge il nodoRadice alla fine della lista dei nodi figli di questo nodo

                    DefaultMutableTreeNode topAlbero = creaAlberoNodi(nodoRadice); // dato il nodo radice crea l'albero

                    modelloAlberoXmlTag.setRoot(topAlbero); // imposta topAlbero come radice
                    alberoXmlTag.treeDidChange(); // impostato quando il JTree e' cambiato a tal punto da necessitare il ridimensionamento
                    espandiAlbero(alberoXmlTag); // espande l'albero JTree legato al documento Xml affinche' sia visibile nel viewer
                    aggiornaAttributiNodoInTavola(null); // imposta a null gli attributi nella tabella dei nodi
                }
                catch (Exception e)
                {
                    JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "Eccezione rile
                    oggettoXMLStudioGlobale.stato.setText("Errore nella creazione del documento tag (.xml)");
                    oggettoXMLStudioGlobale.cursoreNormale(); // imposta il cursore normale in tutta l'applicazione
                }
                finally
                {
                    oggettoXMLStudioGlobale.cursoreNormale(); // imposta il cursore normale in tutta l'applicazione
                }
            }
            else
            {
                JOptionPane.showInternalMessageDialog(oggettoXMLStudioGlobale.getContentPane(), "E' stato inserito un nome di tag relativo al nod
            }
}

// **** metodo per l'apertura di un documento xml che contiene i tag da inserire ****

protected void azioneApriDocumentoXmlTag()
{
            Thread runner = new Thread() // uso dei thread per consentire il cambio del cursore in clessidra durante il caricamento
            {
                public void run()
                {
                    azioneRicordaSalva(); // chiede all'utente se salvare il file tag (xml) attualmente caricato
                    //JFileChooser fileChooser = new JFileChooser(directoryCorrente); // crea un nuovo oggetto di tipo JFileChooser
                    JFileChooser fileChooser = new JFileChooser("./tag/"); // crea un nuovo oggetto di tipo JFileChooser
                    fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY); // consenti la selezione solo di file
                    FiltroFile xmlFiltro = new FiltroFile("xml", "File xml *.XML"); // imposta il filtro per i file .xml (che rappresentano i tag)
                    fileChooser.addChoosableFileFilter(xmlFiltro); // aggiunge il filtro alla finestra di apertura
                    fileChooser.setFileFilter(xmlFiltro); // imposta il filtro xmlFiltro come filtro di default

                    int ritornoValore = fileChooser.showOpenDialog(oggettoXMLStudioGlobale.getContentPane()); // mostra la finestra dialog

                    if (ritornoValore == JFileChooser.APPROVE_OPTION)
                    {
                        File f = fileChooser.getSelectedFile(); // associa all'oggetto f il file selezionato
                        if (f == null || !f.isFile()) // se f e' nullo o non e' un file allora esci dal metodo
                            return;

                        oggettoXMLStudioGlobale.cursoreClessidra(); // imposta il cursore clessidra in tutta l'applicazione
                        try
                        {
                            // DocumentBuilderFactory definisce API che consentono ad applicazioni di ottenere un parser ch
                            DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance(); // crea un
                            // DocumentBuilder e' usato per ottenere un'istanza di documento DOM da un documento XML
                            DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder(); // crea una nuova istar
```

```
                                        documentoXmlTag = docBuilder.parse(f); // parsifica il contenuto di un dato file come un documer

                                        Element nodoRadice = documentoXmlTag.getDocumentElement(); // individua il nodo radice dal do
                                        nodoRadice.normalize(); // assicura che la vista del documento DOM sia la stessa come se fosse

                                        DefaultMutableTreeNode nodoRadiceXmlTag = creaAlberoNodi(nodoRadice);

                                        modelloAlberoXmlTag.setRoot(nodoRadiceXmlTag); // imposta nodoRadiceXmlTag come radice
                                        alberoXmlTag.treeDidChange(); // impostato quando il JTree e' cambiato a tal punto da necessitar
                                        espandiAlbero(alberoXmlTag); // espande l'albero JTree legato al documento Xml affinche' sia visi
                                        aggiornaAttributiNodoInTavola(null); // imposta a null gli attributi nella tabella dei nodi
                                        fileXmlTagCorrente = f;
                                        oggettoXMLStudioGlobale.stato.setText("File " + f.getName() + " caricato."); // indica l'avvenuto c
                                }
                                catch (Exception e)
                                {
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Inforn
                                        oggettoXMLStudioGlobale.stato.setText("Errore di apertura del file " + f.getName());
                                        System.err.println("Errore di apertura del file " + f.getName());
                                        oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
                                }
                                finally
                                {
                                        oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
                                }
                        }
                }
        };
        runner.start();
}


// **** metodo per il caricamento automatico del file xml che contiene i tag da inserire ****

protected void caricaDocumentoXmlTag(final String fileXmlTag)
{
        Thread runner = new Thread() // uso dei thread per consentire il cambio del cursore in clessidra durante il caricamento
        {
                public void run()
                {
                        if (fileXmlTag == null) // se non e' presente il file di default dei tag xml da caricare
                                return;

                        File f = new File(fileXmlTag); // associa all'oggetto f il file selezionato
                        if (f == null || !f.isFile()) // se f e' nullo o non e' un file allora esci dal metodo
                        {
                                JOptionPane.showInternalMessageDialog(oggettoXMLStudioGlobale.getContentPane(), "Non e' stato caricato i
                                return;
                        }

                        oggettoXMLStudioGlobale.cursoreClessidra(); // imposta il cursore clessidra in tutta l'applicazione
                        try
                        {
                                // DocumentBuilderFactory definisce API che consentono ad applicazioni di ottenere un parser che produce d
                                DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance(); // crea una nuova ista
                                // DocumentBuilder e' usato per ottenere un'istanza di documento DOM da un documento XML
                                DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder(); // crea una nuova istanza di Docun
                                documentoXmlTag = docBuilder.parse(f); // parsifica il contenuto di un dato file come un documento XML e re

                                Element nodoRadice = documentoXmlTag.getDocumentElement(); // individua il nodo radice dal documento Xm
```

```
                                        nodoRadice.normalize(); // assicura che la vista del documento DOM sia la stessa come se fosse salvato e ca

                                        DefaultMutableTreeNode nodoRadiceXmlTag = creaAlberoNodi(nodoRadice);

                                        modelloAlberoXmlTag.setRoot(nodoRadiceXmlTag); // imposta nodoRadiceXmlTag come radice
                                        alberoXmlTag.treeDidChange(); // impostato quando il JTree e' cambiato a tal punto da necessitare il ridimen
                                        espandiAlbero(alberoXmlTag); // espande l'albero JTree legato al documento Xml affinche' sia visibile nel view
                                        aggiornaAttributiNodoInTavola(null); // imposta a null gli attributi nella tabella dei nodi
                                        fileXmlTagCorrente = f;
                                        oggettoXMLStudioGlobale.stato.setText("File " + f.getName() + " precaricato per tag xml."); // indica l'avvenu
                                }
                                catch (Exception e)
                                {
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "E
                                        oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
                                }
                                finally
                                {
                                        oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
                                }
                        }
                };
                runner.start();
}


// **** metodo per il salvataggio effettivo del file xml che contiene i tag da inserire ****

protected void azioneSalvaGenerica(File filesalva)
{
        boolean salvareFile = false;

        if (filesalva.exists()) // il file selezionato esiste gia' nella directory corrente
        {
                int ritornoValore = JOptionPane.showConfirmDialog(oggettoXMLStudioGlobale.getContentPane(), "Si desidera sovrascrivere il file " +
                if (ritornoValore == JOptionPane.YES_OPTION)
                        salvareFile = true; // e' stato selezionato di sovrascrivere il file
                if (ritornoValore == JOptionPane.NO_OPTION)
                        salvareFile = false; // e' stato selezionato di non sovrascrivere il file
        }
        else
        {
                salvareFile = true; // il file non esiste nella directory corrente e viene salvato
        }

        if (salvareFile == true) // gestisce il salvataggio del file
        {
                oggettoXMLStudioGlobale.cursoreClessidra(); // imposta il cursore clessidra in tutta l'applicazione
                fileXmlTagCorrente = filesalva; // in fileCorrente salva i dati relativi al fine appena salvato per successivo uso
                try // scrive su file l'albero relativo ai tag xml
                {
                        FileWriter out = new FileWriter(filesalva);
                        XMLRoutine.writeDocumento(documentoXmlTag, out);
                        out.close();
                        CopiaCorretta copia = new CopiaCorretta(filesalva);
                        oggettoXMLStudioGlobale.stato.setText("File " + filesalva.getName() + " salvato."); // indica l'avvenuto salvataggio
                }
                catch (Exception e)
                {
                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "Eccezione rile
```

```
                        oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
                        oggettoXMLStudioGlobale.stato.setText("Errore di salvataggio del file " + filesalva.getName() + "."); // indica l'avvenuto s
                        // JOptionPane.showInternalMessageDialog(oggettoXMLStudioGlobale.getContentPane(), "Salvato e correntemente usat
                        // JOptionPane.showInternalMessageDialog(XmlTag.this, "Salvato e correntemente usato il file xml tag" + fileXmlTagCorr
                }
                oggettoXMLStudioGlobale.cursoreNormale(); // reimposta il cursore predefinito
        }
        else
        {
                oggettoXMLStudioGlobale.stato.setText("File " + filesalva.getName() + " non salvato."); // indica l'avvenuto salvataggio
        }
}


// **** metodo per il savaltaggio del file xml che contiene i tag da inserire ****

protected void azioneSalvaDocumentoXmlTag(boolean salvaNome)
{

        if (documentoXmlTag != null) // e' attualmente caricato un documento relativo a tag xml
        {
                if (salvaNome == true) // si desidera salvare il file xml tag assegnandogli un nuovo nome
                {
                        try
                        {
                                JFileChooser fileChooser = new JFileChooser("./tag/"); // crea l'oggetto per l'apertura del file sulla directo

                                fileChooser.setApproveButtonText("Salva"); // imposta il valore del bottone di esecuzione
                                fileChooser.setDialogTitle("Salva file xml tag"); // imposta il titolo della finestra Dialog

                                // imposta ed aggiunge i filtri opportuni
                                fileChooser.resetChoosableFileFilters(); // reinizializza l'oggetto fileChooser
                                FiltroFile xmlFiltro = new FiltroFile("xml", "File xml *.XML"); // imposta il filtro per i file .xml
                                fileChooser.addChoosableFileFilter(xmlFiltro); // aggiunge il filtro alla finestra di apertura
                                fileChooser.setFileFilter(xmlFiltro); // imposta il filtro xmlFiltro come filtro di default

                                int ritornoValore = fileChooser.showSaveDialog(oggettoXMLStudioGlobale.getContentPane()); // mostra la fin

                                if (ritornoValore == JFileChooser.APPROVE_OPTION)
                                {
                                        File f = fileChooser.getSelectedFile(); // associa all'oggetto f il file selezionato
                                        azioneSalvaGenerica(f);
                                }
                        }
                        catch (Exception e)
                        {
                                JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "E
                        }
                }
                else // si desidera sovrascrivere il file xml tag senza assegnargli un nuovo nome
                {
                        if (fileXmlTagCorrente.isFile() == true)
                        {
                                try
                                {
                                        azioneSalvaGenerica(fileXmlTagCorrente);
                                }
                                catch(Exception e)
                                {
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Inform
```

```
                                        }
                                }
                        }
                }
}


// **** metodo per ricordare di salvare il file xml tag attualmente aperto ****

protected void azioneRicordaSalva()
{
        if (oggettoXMLStudioGlobale.getfileXmlTagCorrente() != null)
        {
                int ritornoValore = JOptionPane.showConfirmDialog(oggettoXMLStudioGlobale.getContentPane(), "Si desidera salvare il file corrente
                if (ritornoValore == JOptionPane.YES_OPTION)
                        azioneSalvaDocumentoXmlTag(false); // salva il file senza associare un nuovo nome
        }
}


// **** metodo per l'aggiunta di un nodo tag all'albero xml tag ****

protected void azioneAggiungiNodoXmlTag()
{
        if (documentoXmlTag == null) // se non fosse caricato nessun documento xml relativo ai tag
                return;

        XmlNodo nodoSelezionato = restituisciNodoSelezionatoAlbero(); // in nodoSelezionato vi e' il nodo attualmente selezionato
        if (nodoSelezionato == null) // se non e' selezionato nessun nodo
                return;

        Node nodoGenitoreNodoTag = nodoSelezionato.restituisciXmlNodo(); // e' effettivamente stato selezionato un nodo
        if (nodoGenitoreNodoTag == null) // se non e' null, prosegui
                return;

        String nomeTag = (String)JOptionPane.showInputDialog(this, "Inserire il nome del nuovo tag xml", "Inserimento tag", JOptionPane.PLAIN_MESSA
        if (nomeTag == null) // se e' stato premuto il pulsante annulla, esci
                return;
        if (nomeXmlValido(nomeTag))
        {
                try
                {
                        Element nuovoElementoTag = documentoXmlTag.createElement(nomeTag); // specificando un nomeTag crea un elemento
                        XmlNodo nuovoTagNodo = new XmlNodo(nuovoElementoTag); // crea un nodo di tipo XmlNodo
                        nodoSelezionato.aggiungiXmlNodo(nuovoTagNodo); // aggiunge il nodo figlio nuovoTagNodo al nodo preesistente padre

                        modelloAlberoXmlTag.nodeStructureChanged(nodoSelezionato); // necessario per mostrare il nuovo nodo
                        TreePath path = alberoXmlTag.getSelectionPath(); // in path vi e' il cammino del nodo attualmente selezionato
                        if (path != null)
                        {
                                path = path.pathByAddingChild(nuovoTagNodo);
                                alberoXmlTag.setSelectionPath(path); // seleziona il nodo specificato dal path indicato
                                alberoXmlTag.scrollPathToVisible(path); // espande i cammini dei componenti sino all'attuale che e' mostrato
                        }
                }
                catch (Exception e)
                {
                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "Eccezione ril
                }
        }
```

```
		else
		{
			JOptionPane.showInternalMessageDialog(oggettoXMLStudioGlobale.getContentPane(), "E' stato inserito un nome di tag non valido.", "
		}
}


// **** metodo per la modifica (editing) di un nodo xml tag ****

protected void azioneEditaNodoXmlTag()
{
		TreePath path = alberoXmlTag.getSelectionPath(); // in path vi e' il cammino del nodo attualmente selezionato
		XmlNodo nodoAlberoPadre = restituisciNodoSelezionatoAlbero(); // in nodoAlberoPadreXml viene messo il nodo attualmente selezionato
		if (nodoAlberoPadre == null)
			return;
		Node nodo = nodoAlberoPadre.restituisciXmlNodo(); // restituisce il nodo corrente di tipo node dal nodo di tipo XmlNodo
		if (nodo == null)
			return;
		try
		{
			switch (nodo.getNodeType()) // estrapola il tipo del nodo attualmente esaminato
			{
				case Node.ELEMENT_NODE: // il nodo e' di tipo element

					// individua il nodo testuale figlio per editarlo
					for (int k=0; k < nodoAlberoPadre.getChildCount(); k++) // per ogni nodo figlio della radice fai
					{
						XmlNodo childNode = (XmlNodo)nodoAlberoPadre.getChildAt(k); // individua il k-esimo nodo figlio
						Node nd = childNode.restituisciXmlNodo(); // restituisce il nodo corrente di tipo node dal nodo di t
						if (nd instanceof Text)
						{

							path = path.pathByAddingChild(childNode);
							alberoXmlTag.setSelectionPath(path); // seleziona il nodo specificato dal path indicato
							alberoXmlTag.scrollPathToVisible(path); // espande i cammini dei componenti sino all'
							alberoXmlTag.startEditingAtPath(path); // seleziona il nodo identificato dal path ed iniz
							return; // esce
						}
					}
					// non ha individuato il nodo testuale figlio, allora crea un nuovo nodo testuale figlio
					Text testo = documentoXmlTag.createTextNode(""); // crea un nodo testuale con testo vuoto
					XmlNodo nodoFiglio = new XmlNodo(testo); // crea un nuovo nodo di tipo XmlNodo
					nodoAlberoPadre.aggiungiXmlNodo(nodoFiglio); // aggiunge il nodo figlio al nodo padre
					modelloAlberoXmlTag.nodeStructureChanged(nodoAlberoPadre);
					path = path.pathByAddingChild(nodoFiglio);
					alberoXmlTag.setSelectionPath(path); // seleziona il nodo specificato dal path indicato
					alberoXmlTag.scrollPathToVisible(path); // espande i cammini dei componenti sino all'attuale che e' mostrato
					alberoXmlTag.startEditingAtPath(path); // seleziona il nodo identificato dal path ed inizia il suo editing
					return; // esce
				case Node.TEXT_NODE: // il nodo e' di tipo testuale
					alberoXmlTag.startEditingAtPath(path); // seleziona il nodo identificato dal path ed inizia il suo editing
					return; // esce
			}
		}
		catch (Exception e)
		{
			JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "Eccezione rilevata nella r
		}
}
```

```
// **** metodo per la cancellazione di un nodo xml tag ****

protected void azioneCancellaNodo()
{
	TreePath path = alberoXmlTag.getSelectionPath(); // restituisce il path del nodo selezionato grazie a getSelectionPath
	XmlNodo nodoAlbero = restituisciNodoSelezionatoAlbero(); // in nodoAlbero viene messo il nodo attualmente selezionato
	if (nodoAlbero != null)
	{
		Node nodo = nodoAlbero.restituisciXmlNodo(); // restituisce il nodo corrente di tipo node dal nodo di tipo XmlNodo
		if (nodo != null)
		{
			int nomeAttributo = JOptionPane.showConfirmDialog(oggettoXMLStudioGlobale.getContentPane(), "Si desidera cancellare
			if (nomeAttributo == JOptionPane.YES_OPTION)
			{
				try
				{
					TreeNode nodoPadre = nodoAlbero.getParent(); // determina il nodo padre del nodo di tipo XmlNo
					nodoAlbero.rimuovi(); // rimuove il nodoAlbero e posiziona la selezione sul nodo padre
					modelloAlberoXmlTag.nodeStructureChanged(nodoPadre); // aggiorna la struttura dell'albero in b
				}
				catch (Exception e)
				{
					JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Inform
				}
			}
		}
	}
}


// **** aggiunge un attributo al nodo selezionato ****

protected void azioneAggiungiAttributoNodo()
{
	Node nodo = modelloTavolaAttributiXmlTag.restituisciNodoCorrente(); // restituisce il nodo esaminato
	if ((nodo instanceof Element) == true) // se siamo in presenza di un nodo di tipo element  (e non della porzione #text)
	{
		String nomeNuovoAttributo = (String)JOptionPane.showInputDialog(this, "Inserire un nuovo nome di attributo", "Inserimento attributi
		if (nomeNuovoAttributo == null) // se e' stato premuto il pulsante annulla, esci
			return;
		if (nomeXmlValido(nomeNuovoAttributo) == true) // se il valore inserito e' un nome di attributo valido in xml
		{
			try
			{
				((Element)nodo).setAttribute(nomeNuovoAttributo, ""); // Element identifica un elemento xml e aggiunge un a
				aggiornaAttributiNodoInTavola(nodo); // aggiorna l'albero xml con il nodo attualmente selezionato
				for (int k=0; k < modelloTavolaAttributiXmlTag.getRowCount(); k++) // cerca l'attributo e poi edita il suo valor
				{
					if (modelloTavolaAttributiXmlTag.getValueAt(k, 0).equals(nomeNuovoAttributo)) // ha individuato il
					{
						tavolaAttributiXmlTag.requestFocus(); // si assegna il focus alla JTable in modo da co
						tavolaAttributiXmlTag.editCellAt(k, 1); // edita la cella valor della JTable relativa all'attr
						break; // se ha individuato l'attributo e quindi il valore, esce dal ciclo
					}
				}
			}
			catch (Exception e)
			{
				JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Informazione", "E
```

```
                                        }
                                }
                        }
                }


// **** modifica il valore dell'attributo relativo al nodo selezionato ****

protected void azioneModificaAttributoNodo()
{
        // e' possibile richiamare questo metodo solo dopo che un attributo della JTable e' stato selezionato

        tavolaAttributiXmlTag.requestFocus(); // si assegna il focus alla JTable in modo da consentire il diretto inserimento del valore
        int riga = tavolaAttributiXmlTag.getSelectedRow(); // restituisce l'indice relativo alla riga selezionata
        if (riga >= 0) // se e' stata selezionata una riga
                tavolaAttributiXmlTag.editCellAt(riga, 1); // modifica il valore dell'attributo relativo a riga
}


// **** elimina un attributo dal nodo selezionato ****

protected void azioneEliminaAttributoNodo()
{
        // e' possibile richiamare questo metodo solo dopo che un attributo della JTable e' stato selezionato

        int rigaAttributoSelezionato = tavolaAttributiXmlTag.getSelectedRow(); // restituisce l'indice relativo alla riga selezionata
        if (rigaAttributoSelezionato >= 0) // se e' stata selezionata una riga della JTable relativa ad un attributo e ad un valore
        {
                Node nodoEliminare = restituisciNodoSelezionato(); // in nodoEliminare e' presente il nodo da eliminare
                if (nodoEliminare instanceof Element) // se siamo effettivamente in presenza di un nodo
                {
                        String nomeAttributo = (String)tavolaAttributiXmlTag.getValueAt(rigaAttributoSelezionato, 0);
                        int ritornoValore = JOptionPane.showConfirmDialog(oggettoXMLStudioGlobale.getContentPane(), "Si desidera eliminare l'
                        if (ritornoValore == JOptionPane.YES_OPTION) // e' confermata l'eliminazione del nodo
                        {
                                try
                                {
                                        ((Element)nodoEliminare).removeAttribute(nomeAttributo); // rimuove un attributo avendone indi
                                        aggiornaAttributiNodoInTavola(nodoEliminare); // aggiorna l'albero xml con il nodo attualmente se
                                }
                                catch (Exception e)
                                {
                                        JOptionPaneEsteso mostraEccezione = new JOptionPaneEsteso(oggettoXMLStudioGlobale, "Inform
                                }
                        }
                }
        }
}


// **** restituisce vero se il testo e' un nome valido come tag di un documento xml ****

private static boolean nomeXmlValido(String testo)
{
        if (testo == null || testo.length() == 0)
                return false;
        for (int k=0; k < testo.length(); k++)
        {
                char carattere = testo.charAt(k);
                if (Character.isLetter(carattere) || (carattere == '_') || (carattere == ':') || (k>0 && (Character.isDigit(carattere) || (carattere == '.')
```

```
                                continue;
                        return false;
                }
                return true;
}


// **** analizza il nodo attualmente selezionato e abilita e disabilita i pulsanti per il corretto funzionamento ****

protected void abilitaPulsantiNodi()
{
                boolean b1 = (restituisciNodoSelezionato() instanceof Element); // siamo in presenza di un nodo tag non text#
                boolean b2 = (restituisciNodoSelezionato() != null);

                aggiunginodoxmltag.setEnabled(b1);
                oggettoXMLStudioGlobale.jmi_aggiunginodotag.setEnabled(b1);

                editanodoxmltag.setEnabled(b2);
                oggettoXMLStudioGlobale.jmi_modificanodotag.setEnabled(b2);

                cancellanodoxmltag.setEnabled(b2);
                oggettoXMLStudioGlobale.jmi_eliminanodotag.setEnabled(b2);
}


// **** analizza il nodo attualmente selezionato e abilita e disabilita i pulsanti per il corretto funzionamento ****

protected void abilitaPulsantiAttributi()
{
                boolean b1 = (modelloTavolaAttributiXmlTag.restituisciNodoCorrente() instanceof Element); // siamo in presenza di un nodo tag non text#
                boolean b2 = (tavolaAttributiXmlTag.getSelectedRowCount() > 0); // e' selezionata una riga specifica della JTable

                aggiungiattributonodoxmltag.setEnabled(b1);
                oggettoXMLStudioGlobale.jmi_aggiungiattributonodotag.setEnabled(b1);

                modificaattributonodoxmltag.setEnabled(b2);
                oggettoXMLStudioGlobale.jmi_modificaattributonodotag.setEnabled(b2);

                cancellaattributonodoxmltag.setEnabled(b2);
                oggettoXMLStudioGlobale.jmi_eliminaattributonodotag.setEnabled(b2);
}


// **** deseleziona le celle degli attributi e valori evidenziati nella JTable ****

protected void azioneCancellaSelezioneJTable()
{
                tavolaAttributiXmlTag.clearSelection();
}


// **** definisce un'interfaccia per rilevare cambiamenti nell'editing della cella **** ooo

class XmlEditorRileva implements CellEditorListener
{
                // **** questo metodo e' invocato e segnala al listener che e' stato correttamente terminato l'editing ****

                public void editingStopped(ChangeEvent e)
                {
                                String valore = editorCellaAlbero.getCellEditorValue().toString(); // in valore vi e' il nuovo valore del nodo
```

```
                        if (nodoAttualmenteEditato != null) // in nodoAttualmenteEditato vi e' il nodo in fase di editing
                                nodoAttualmenteEditato.setNodeValue(valore); // imposta il nuovo valore nel nodo editato
                        TreePath path = alberoXmlTag.getSelectionPath(); // in path vi e' il cammino del nodo attualmente selezionato
                        if (path != null)
                        {
                                // definisce un generico nodo dell'albero ricavandolo dal cammino path e da getLastPathComponent che restituisce il con
                                DefaultMutableTreeNode alberoNodi = (DefaultMutableTreeNode)path.getLastPathComponent();
                                // imposta lo UserObject del nodo al nodoAttualmenteEditato
                                alberoNodi.setUserObject(nodoAttualmenteEditato);
                                // nodeStructureChanged e' opportuno invocarlo perche' e' cambiata la struttura dell'albero con questo nuovo alberoNod
                                modelloAlberoXmlTag.nodeStructureChanged(alberoNodi);
                        }
                        nodoAttualmenteEditato = null; // si reimposta il fatto che attualmente non vi e' piu' alcun nodo editato
                }

                // **** questo metodo e' invocato e segnala al listener che e' stato interrotto l'editing della cella ****

                public void editingCanceled(ChangeEvent e)
                {
                        nodoAttualmenteEditato = null;
                }
        }

}
```