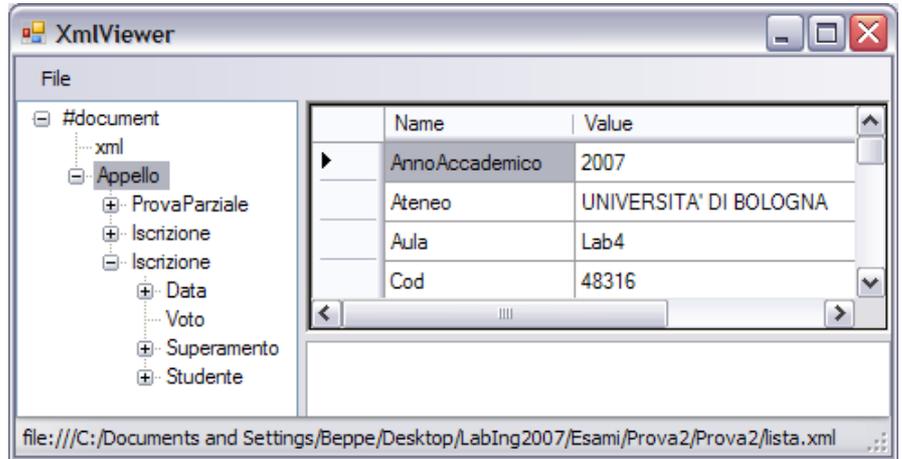


Avvertenze

All'inizio di ogni file sorgente inserire in un commento i propri dati: **cognome, nome e numero di matricola**. Al termine dell'esame, **consegnare tutti i file sorgenti realizzati dal candidato**.

Premessa

Si deve realizzare un visualizzatore di documenti XML. L'applicazione si presenta come in figura: a sinistra è possibile navigare nei documenti XML caricati e selezionare uno dei nodi; in alto a destra vengono visualizzati tutti gli eventuali attributi XML associati al nodo correntemente selezionato; in basso a destra viene visualizzato l'eventuale testo associato al nodo correntemente selezionato.



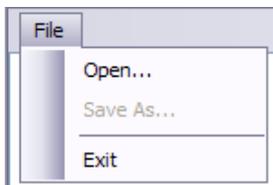
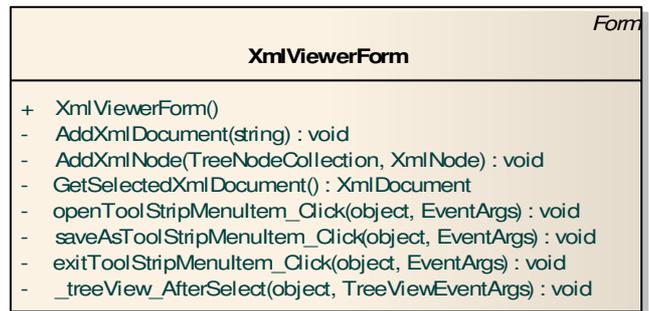
Il file "Prova6Start.zip" contiene alcuni file XML da utilizzare durante la fase di test dell'applicazione.

Passo 0

Creare un nuovo progetto di nome **Prova6** di tipo *Windows Forms Application*, rinominare la classe **Form1** (e il corrispondente file) in **XmlViewerForm** e fare il *build*.

Passo 1

La *form* **XmlViewerForm** è descritta dal diagramma semplificato UML di figura. Inserire nella *form* un **MenuStrip**, uno **StatusStrip** contenente una **ToolStripStatusLabel** (con proprietà **Spring = True** e **TextAlign = MiddleLeft**) e un primo **SplitContainer** che riempia l'area restante della *form*. Nello **SplitContainer** inserire a sinistra un **TreeView** (di nome **_treeView**) e a destra un secondo **SplitContainer** con orientamento orizzontale. Nel secondo **SplitContainer** lasciare momentaneamente vuoto il pannello superiore e inserire in basso una **TextBox** (di nome **_textBox**) multilinea, *readonly* e con lo sfondo bianco. Modificare in modo opportuno la proprietà **Dock** dei vari controlli. Sempre mediante *designer*, inserire il gestore dell'evento **AfterSelect** del **TreeView**.



Il menù "File" deve apparire come in figura. Il gestore di "Open..." deve chiedere all'utente di selezionare un file XML da caricare e quindi deve invocare il metodo **AddXmlDocument** passando come argomento il nome del file selezionato. Il gestore di "Save As..." deve permettere all'utente di salvare il documento XML cui appartiene il nodo correntemente selezionato (restituito dal metodo **GetSelectedXmlDocument**); si noti che la voce di menù è inizialmente disabilitata e deve essere abilitata al caricamento del primo documento XML. Il gestore di "Exit" deve chiudere l'applicazione.

Il metodo **AddXmlDocument** deve caricare il contenuto del file XML in un documento XML in memoria e associare il documento XML al **TreeView**. Poiché sia il documento XML, sia il **TreeView** hanno una struttura gerarchica, l'associazione consiste nel creare un **TreeNode** per ogni **XmlNode** esistente nel documento XML, dare al **TreeNode** lo stesso nome del nodo XML, associare al **TreeNode** il nodo XML (mediante la proprietà **Tag**) e aggiungere il **TreeNode** nella giusta posizione all'interno del **TreeView**. Tutte queste operazioni vengono effettuate dal metodo **AddXmlNode** che crea in modo opportuno il **TreeNode**, lo aggiunge alla **TreeNodeCollection** passata come primo argomento e quindi va in ricorsione sugli eventuali figli del nodo XML passato come secondo argomento. Il metodo **AddXmlDocument** inizia la fase di associazione del documento XML al **TreeView** invocando il metodo **AddXmlNode** e passando come primo argomento la lista

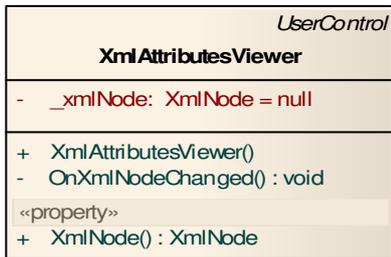
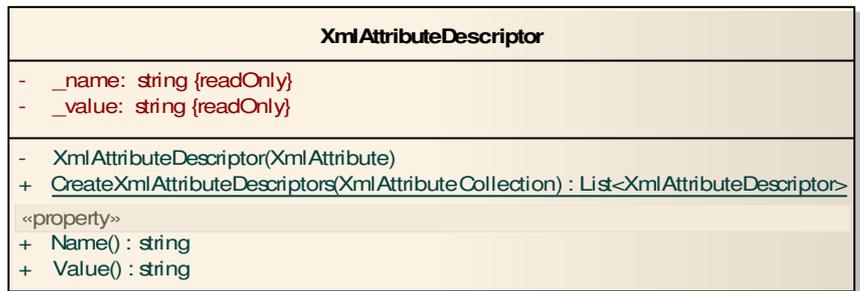
dei nodi del **TreeView** (proprietà **Nodes**) e come secondo argomento il documento XML stesso. Infine, **AddXmlDocument** deve fare in modo che nel **TreeView** venga selezionato il **TreeNode** associato al documento XML appena caricato – si utilizzi la proprietà **SelectedNode** del **TreeView**.

Il metodo **GetSelectedXmlDocument** deve restituire il documento XML cui appartiene l'**XmlNode** associato al **TreeNode** correntemente selezionato (si rammenti che il **TreeView** può visualizzare più documenti XML contemporaneamente) – a tal fine, si sfrutti la proprietà **OwnerDocument** di **XmlNode** e si tenga presente che il valore di tale proprietà è **null** nel caso in cui il nodo sia l'**XmlDocument** stesso.

Il metodo **_treeView_AfterSelect**, in base all'**XmlNode** correntemente selezionato, deve aggiornare sia la **TextBox** in modo che visualizzi l'eventuale testo dell'**XmlNode** (proprietà **Value** di **XmlNode**), sia la **ToolStripStatusLabel** in modo che visualizzi il nome completo del file contenente il documento XML cui appartiene l'**XmlNode** (si sfrutti la proprietà **BaseURI** di **XmlDocument**).

Passo 2

Aggiungere al progetto la classe **XmlAttributeDescriptor** (ved. diagramma UML). Tale classe descrive un singolo attributo XML con una coppia di proprietà **Name**, **Value**. L'unico metodo pubblico **CreateXmlAttributeDescriptors** accetta come argomento una collezione di attributi XML e restituisce una lista di descrittori creati in modo opportuno.



Aggiungere al progetto uno **UserControl** di nome **XmlAttributeViewer** (ved. diagramma semplificato UML). Tale controllo deve gestire la visualizzazione degli eventuali attributi di un nodo XML (**_xmlNode**). A tal fine, inserire nel controllo una **DataGridView** (di nome **_dataGridView**) che riempia tutto il controllo, che non permetta di modificare il contenuto visualizzato e che abbia il valore della proprietà **AutoSizeColumnsMode** uguale a **AllCells**. Inoltre, quando alla proprietà **XmlNode** viene assegnato un nuovo nodo XML, occorre invocare il metodo **OnXmlNodeChanged**. Nel metodo **OnXmlNodeChanged**, è necessario

prima assegnare alla proprietà **DataSource** di **_dataGridView** il valore **null** e quindi, se il nodo corrente esiste e può avere attributi, assegnare a **DataSource** la collezione restituita dal metodo **CreateXmlAttributeDescriptors**.

Infine, nella *form* principale inserire nel pannello superiore del secondo **SplitContainer** uno **XmlAttributeViewer** (di nome **_xmlAttributesViewer**) e inserire nel metodo **_treeView_AfterSelect** l'istruzione che aggiorna il valore della proprietà **XmlNode** di **_xmlAttributesViewer**.