Prova 5

Avvertenze per la consegna

All'inizio di ogni file inserire in un commento i propri dati: **cognome, nome e numero di matricola**. Al termine dell'esame, **consegnare (inviare) i file sorgenti di cui sopra**.

Premessa

Lo scopo dell'applicazione è quello di copiare i dati contenuti nelle **`DataTable`** di un **`DataSet`** (con struttura NON nota) in liste di istanze di classi "adatte" a contenere tali dati (una lista per **`DataTable`**). Le classi "adatte" di cui sopra sono contenute in un *assembly* di cui non si conosce la struttura. Il *match* fra le classi e le **`DataTable`** deve essere fatto *run-time* tramite *reflection*. Per verificare che il trasferimento dei dati abbia avuto successo, occorre creare una *form* che consenta di selezionare una delle liste create e di visualizzarne il contenuto.

Passo 0: Startup

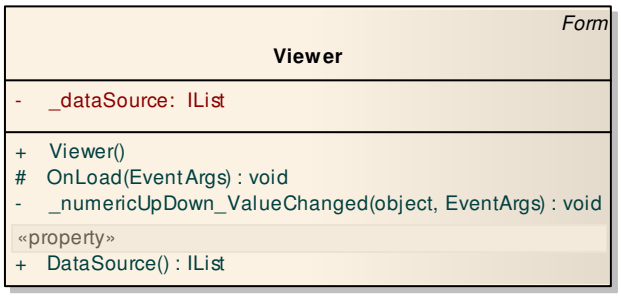
Creare un nuovo progetto di tipo *Windows Forms Application*, di nome **`Prova5`** e rinominare il file e la classe **`Form1`** in **`Viewer`**. Scaricare il file "Prova5Start.zip" contenente l'*assembly* **`Library.dll`** e aggiungere al progetto un riferimento a tale *assembly*.

Passo 1: La classe `Viewer`

Aggiungere alla *form* **`Viewer`** uno **`SplitContainer`** contenente a sinistra un controllo **`NumericUpDown`** e a destra una **`DataGridView`**; il risultato finale deve essere simile a quello di figura.

Il diagramma UML relativo alla classe **`Viewer`** contiene i metodi e le proprietà da realizzare.

La proprietà **`DataSource`** consente di accedere a un oggetto di tipo **`IList`**, che contiene oggetti di tipo **`IList`** – cioè, una lista (di primo livello) di liste (di secondo livello).



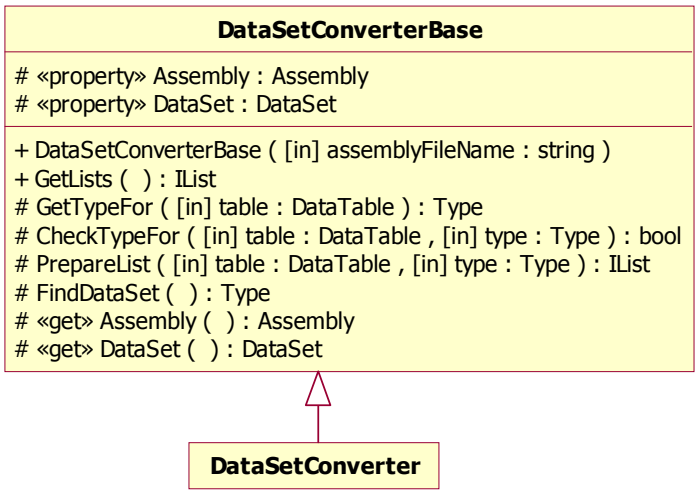
Il **`NumericUpDown`** deve mostrare l'indice della lista di secondo livello correntemente visualizzata nella **`DataGrid`**; per fare questo, occorre assegnare alle proprietà **`Minimum`**, **`Maximum`** e **`Value`** del controllo valori appropriati. Se la lista di primo livello è vuota, il **`NumericUpDown`** deve essere disabilitato e la **`DataGrid`** deve essere vuota.

Per verificare il corretto funzionamento del **`Viewer`**, definire la classe **`DataSetConverter`** sottoclasse di **`DataSetConverterBase`** (come da diagramma UML a pagina 2) con il solo costruttore che deve invocare in modo opportuno il costruttore della classe base (**`DataSetConverterBase`** è definita in **`Library.dll`**).

Nel metodo **`OnLoad`** del **`Viewer`** creare un'istanza di **`DataSetConverter`** passando come argomento la stringa "**`Library.dll`**" e assegnare alla proprietà **`DataSource`** del **`Viewer`** il risultato dell'invocazione del metodo **`GetLists`**. Il risultato dovrà essere simile a quello riportato nella prima figura.

Passo 2: La classe `DataSetConverter`

Il costruttore della classe **`DataSetConverterBase`** carica l'*assembly* di cui è stato passato come argomento il nome (la proprietà **`Assembly`** restituisce il riferimento all'*assembly* caricato) e quindi crea un'istanza dell'unica sottoclasse di **`DataSet`** definita in tale *assembly* (la proprietà **`DataSet`** restituisce il riferimento a tale istanza).



Tutti gli altri metodi implementati nella classe **`DataSetConverterBase`** sono virtuali. Tali metodi devono essere ridefiniti nella classe **`DataSetConverter`** in modo che il funzionamento dell'applicazione rimanga invariato.

Il metodo **`FindDataSet`** deve restituire la prima sottoclasse di **`DataSet`** trovata nell'**`Assembly`**, oppure **`null`**.

Il metodo **`CheckTypeFor`** deve verificare se il **`Type type`** è "adatto" ad ospitare i dati relativi ad una riga della **`DataTable table`**. A tale scopo, occorre verificare che

- il numero di colonne di **`table`** sia minore o uguale al numero di proprietà pubbliche di **`type`**;
- per ogni colonna di **`table`** esista una proprietà pubblica di **`type`** con lo stesso nome e lo stesso tipo e che tale proprietà possa essere utilizzata sia in lettura, sia in scrittura.

Il metodo **`GetTypeFor`** deve restituire la classe, definita nell'**`Assembly`**, "adatta" ad ospitare i dati di una riga della **`DataTable table`**, oppure **`null`** se una tale classe non esiste. A tale scopo, si utilizzi in modo opportuno il metodo **`CheckTypeFor`**.

Il metodo **`PrepareList`** deve restituire un **`ArrayList`** creato nel seguente modo:

- l'**`ArrayList`** deve contenere tanti oggetti di tipo **`type`** quante sono le righe della **`DataTable table`**;
- ogni oggetto di tipo **`type`** deve essere inizializzato con i valori della corrispondente riga di **`table`**; a tale scopo, assegnare il valore di ogni colonna di **`table`** alla corrispondente proprietà dell'oggetto.

Il metodo **`GetLists`** deve restituire un **`ArrayList`** creato nel seguente modo:

- l'**`ArrayList`** deve contenere tante istanze di **`ArrayList`** quante sono le **`DataTable`** nel **`DataSet`** alle quali è possibile associare una classe dell'**`Assembly`** "adatta" a contenere i dati della **`DataTable`**;
- ogni istanza di **`ArrayList`** deve essere ottenuta utilizzando in modo opportuno il metodo **`PrepareList`**.