DataGrid Sparklines Example

Add sparklines to grid cells.

This page provides information about Sparklines and their usage. Sparklines are small charts that can now be embedded in a DataGrid (Table). They allow users to display past data or performance next to the current value. Use Sparkline trends alongside current values to provide context and indicate potential future trends.

- Understanding Sparklines in DataGrids
- Styling
- Requirements and Procedures
  - Create a Database
  - Configure Tables and Queries
  - Project Configuration
  - Implement CodeBehind
- Runtime Execution

Understanding Sparklines in DataGrids

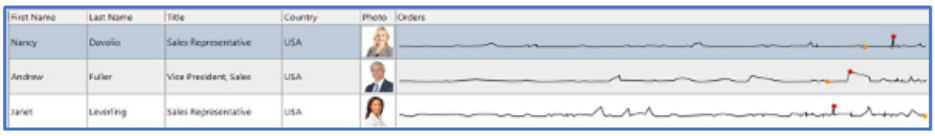
A sparkline is a small line chart, typically drawn without axes or coordinates. Sparklines represent the general shape of a variation over time in a desired measurement, such as temperature or stock market price. They are small enough to be embedded in text or combined as elements of a small group.

Typical charts show as much data as possible and are set off from the text flow. In contrast, sparklines are succinct and located directly in the text.

Styling

Sparklines are available in four styles: **Line**, **Area**, **Bar**, and **WinLoss**.

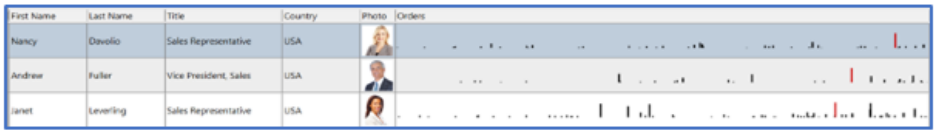
Line



Area



Bar



WinLoss



Requirements and Procedures

Create a Database

Store numeric information for sparklines.

1. Go to **Datasets DBs**.
2. Create a new database
3. Select a **Provider** and **Database** at your choice.

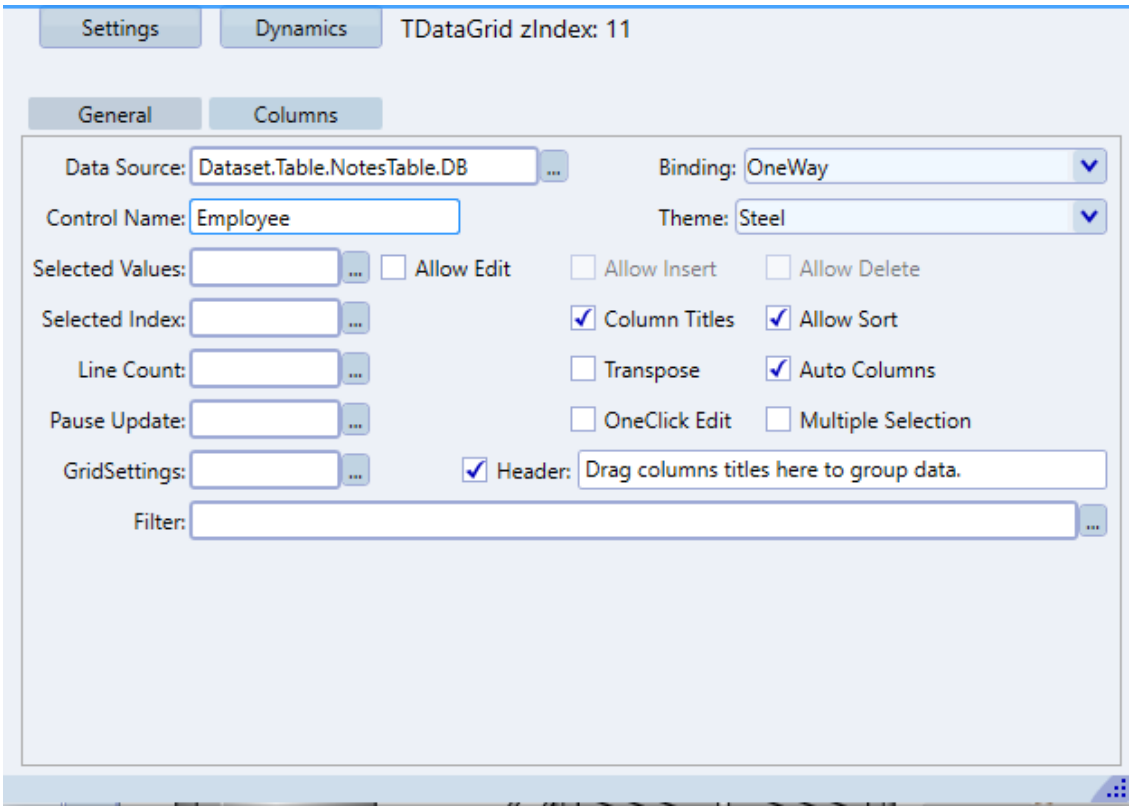
Configure Tables and Queries

Set up tables and queries for the data grid.

1. Go to **Datasets Tables** or **Queries**.
2. Create internal table and queries.
3. Map these tables to existing database tables.
4. Define each query for a specific database with a predefined SQL Statement if needed.

Project Configuration

1. Go to **Displays Draw**.
2. On the Components Panel, select **Modules**, then **DataGrid.**
3. Click or drag-and-drop it on the Drawing area to use it.
4. Double-click the component to open the configuration window.
5. Set a table as the DataSource at **Data Source**,
6. Use the **Control Name** field to manipulate the DataGrid control in CodeBehind.



Implement CodeBehind

Implement the code to display sparklines. Use the following example to set up the DataGrid control:

```
TDataGrid grid = CurrentDisplay.GetDataGrid("Employee");
grid.TableGridControl.SetColumnSparkline("Orders", this.SparklineCollectionDelegate, style);
// SetColumnSparkline parameters - fieldName (string)
//                       - populateCallback (SparklineCollectionDelegate)
//                       - style (string)
```

Define a delegate to refer to a method. The delegate must match the method's signature and return type. Use the following code to create the Sparkline:

```
Dictionary<int, ArgumentValue[]> mapEmployeeToOrders = null;

private ArgumentValue[] SparklineCollectionDelegate(string fieldName, System.Data.DataRow row)
{
    ArgumentValue[] sourceCollection;
    int emplID = TK.To<int>(row["EmployeeID"]);
    if (!this.mapEmployeeToOrders.TryGetValue(emplID, out sourceCollection))
        sourceCollection = new ArgumentValue[0];
    return sourceCollection;
}
```


A delegate will only call a method that agrees with its signature and return type. A method can be either a static method associated with a class, or it can be an instance method associated with an object.

Below is a callback. The system requires this exact code to create the Sparkline. To use this feature, you must include this code in your CodeBehind.

```
Dictionary<int, ArgumentValue[]> mapEmployeeToOrders = null;
private ArgumentValue[] SparklineCollectionDelegate(string fieldName, System.Data.DataRow row)
{
ArgumentValue[] sourceCollection;
int  emplID  =  TK.To< int>(row["EmployeeID"]);
if (!this.mapEmployeeToOrders.TryGetValue(emplID, out sourceCollection))
       sourceCollection = new ArgumentValue[0];
return sourceCollection;
}
```

Update the sparkline using the following logic:

```
//         Get data from tables         // DataTable employees =  @Dataset.Table.Employees.SelectCommand();
DataTable orders = @Dataset.Table.Orders.SelectCommand();
DataTable invoices = @Dataset.Query.Invoices.SelectCommand();
//         //

this.SetSparkline(style); // Set Style to Sparkline this.mapEmployeeToOrders = new Dictionary<int,
ArgumentValue[]>();
foreach (DataRow empl in employees.Rows)
{
int  emplID  =  TK.To<int>(empl["EmployeeID"]);
List<ArgumentValue> sourceCollection = new List<ArgumentValue>();
foreach (DataRow order in orders.Select("EmployeeID = " + emplID))
{
int  orderID  =  TK.To< int>(order["OrderID"]); ArgumentValue av = new ArgumentValue();
av.Argument  =  order["OrderDate"];  //  x-axis  for  sparkline
foreach (DataRow invoice in invoices.Select("OrderID = " + orderID))
{
double value = TK.To< double >(av.Value);
double  quantity  =  TK.To< double >(invoice["Quantity"]);
double  unitPrice  =  TK.To< double >(invoice["UnitPrice"]);
av.Value = value + (quantity * unitPrice); // y-axis for sparkline
}
sourceCollection.Add(av);
}
//  map  values  (y-axis)  and  order  date  (x-axis)  to  employee  ID this.mapEmployeeToOrders[emplID] =
sourceCollection.ToArray();
}
```

Runtime Execution

When the solution runs, the DataGrid control with Sparklines will display real-time data from the configured database. The sparklines will visually represent trends and patterns in the numeric data for each row in the DataGrid. Users can interact with the DataGrid by sorting, filtering, and selecting rows, but the sparklines themselves are display-only and do not support direct interaction.

This setup is particularly useful for monitoring key performance indicators (KPIs) in real-time, as it provides a quick visual summary of trends without the need for detailed analysis. Sparklines help in identifying patterns and anomalies in large datasets, making them valuable in dashboards and reports where space is limited but trend visualization is needed. Additionally, incorporating sparklines in decision support systems offers at-a-glance insights, aiding in quick and informed decision-making.

In this section...