|  |  |
| --- | --- |
| namespace DevExpress.MyGridLookUpEditors {  //The attribute that points to the registration method  [UserRepositoryItem("RegisterMyGridLookUpEdit")]  public class RepositoryItemMyGridLookUpEdit : RepositoryItemGridLookUpEdit {  //The static constructor which calls the registration method  static RepositoryItemMyGridLookUpEdit() { RegisterMyGridLookUpEdit(); }  //Initialize new properties  public RepositoryItemMyGridLookUpEdit() {  }  protected override void OnLoaded() {  base.OnLoaded();  if (IsDesignMode) return;  // Create two columns  GridColumn colId = new GridColumn();  colId.FieldName = colId.Caption = "ID";  colId.VisibleIndex = 0;  GridColumn colName = new GridColumn();  colName.FieldName = colName.Caption = "Name";  colName.VisibleIndex = 1;  GridView gView = this.View;  gView.Columns.Clear();  gView.Columns.Add(colId);  gView.Columns.Add(colName);  // Hide the group panel  gView.OptionsView.ShowGroupPanel = true;  // Initialize data source  DisplayMember = "Name";  ValueMember = "ID";  DataSource = GetData();  }  private DataTable GetData() {  DataTable table = new DataTable();  table.Columns.Add(new DataColumn("ID", typeof(int)));  table.Columns.Add(new DataColumn("Name", typeof(string)));  table.Rows.Add(new object[] { 0, "A" });  table.Rows.Add(new object[] { 1, "B" });  table.Rows.Add(new object[] { 2, "C" });  return table;  }  //The unique name for the custom editor  public const string MyGridLookUpEditName = "MyGridLookUpEdit";  //Return the unique name  public override string EditorTypeName { get { return MyGridLookUpEditName; } }  //Register the editor  public static void RegisterMyGridLookUpEdit() {  //Icon representing the editor within a container editor's Designer  Image img = null;  EditorRegistrationInfo.Default.Editors.Add(new EditorClassInfo(MyGridLookUpEditName,  typeof(MyGridLookUpEdit), typeof(RepositoryItemMyGridLookUpEdit),  typeof(GridLookUpEditBaseViewInfo), new ButtonEditPainter(), true, img));  }  //Override the Assign method  public override void Assign(RepositoryItem item) {  BeginUpdate();  try {  base.Assign(item);  RepositoryItemMyGridLookUpEdit source =  item as RepositoryItemMyGridLookUpEdit;  if (source == null) return;  }  finally {  EndUpdate();  }  }  }  public class MyGridLookUpEdit : GridLookUpEdit {  //The static constructor which calls the registration method  static MyGridLookUpEdit() {  RepositoryItemMyGridLookUpEdit.RegisterMyGridLookUpEdit(); }  //Initialize the new instance  public MyGridLookUpEdit() {  //...  }  //Return the unique name  public override string EditorTypeName { get { return  RepositoryItemMyGridLookUpEdit.MyGridLookUpEditName; } }  //Override the Properties property  //Simply type-cast the object to the custom repository item type  [DesignerSerializationVisibility(DesignerSerializationVisibility.Content)]  public new RepositoryItemMyGridLookUpEdit Properties {  get { return base.Properties as RepositoryItemMyGridLookUpEdit; }  }  }  } | |
| **Visual Basic** | CopyCode image复制代码 |
| Imports System.Reflection  Imports System.Drawing  Imports System.ComponentModel  Imports System.Windows.Forms  Imports DevExpress.XtraEditors  Imports DevExpress.XtraEditors.Repository  Imports DevExpress.XtraEditors.Registrator  Imports DevExpress.XtraEditors.Drawing  Imports DevExpress.XtraEditors.ViewInfo  Imports DevExpress.XtraGrid.Columns  Imports DevExpress.XtraGrid.Views.Grid  'The attribute that points to the registration method  <UserRepositoryItem("RegisterMyGridLookUpEdit")> \_  Public Class RepositoryItemMyGridLookUpEdit  Inherits RepositoryItemGridLookUpEdit  'The static constructor which calls the registration method  Shared Sub New()  RegisterMyGridLookUpEdit()  End Sub  'Initialize new properties  Public Sub New()  End Sub  Protected Overrides Sub OnLoaded()  MyBase.OnLoaded()  If IsDesignMode Then Return  ' Create two columns  Dim colId As New GridColumn  colId.FieldName = "ID"  colId.Caption = "ID"  colId.VisibleIndex = 0  Dim colName As New GridColumn  colName.FieldName = "Name"  colName.Caption = "Name"  colName.VisibleIndex = 1  Dim gView As GridView = View  gView.Columns.Clear()  gView.Columns.Add(colId)  gView.Columns.Add(colName)  ' Hide the group panel  gView.OptionsView.ShowGroupPanel = True  ' Initialize data source  DisplayMember = "Name"  ValueMember = "ID"  DataSource = GetData()  End Sub  Public Function GetData() As DataTable  Dim table As New DataTable()  table.Columns.Add(New DataColumn("ID", GetType(Integer)))  table.Columns.Add(New DataColumn("Name", GetType(String)))  table.Rows.Add(New Object() {0, "A"})  table.Rows.Add(New Object() {1, "B"})  table.Rows.Add(New Object() {2, "C"})  Return table  End Function  'The unique name for the custom editor  Public Const MyGridLookUpEditName As String = "MyGridLookUpEdit"  'Return the unique name  Public Overrides ReadOnly Property EditorTypeName() As String  Get  Return MyGridLookUpEditName  End Get  End Property  'Register the editor  Public Shared Sub RegisterMyGridLookUpEdit()  'Icon representing the editor within a container editor's Designer  Dim img As Image = Nothing  EditorRegistrationInfo.Default.Editors.Add(New EditorClassInfo(MyGridLookUpEditName, \_  GetType(MyGridLookUpEdit), GetType(RepositoryItemMyGridLookUpEdit), \_  GetType(GridLookUpEditBaseViewInfo), New ButtonEditPainter, True, img))  End Sub  'Override the Assign method  Public Overrides Sub Assign(ByVal item As RepositoryItem)  BeginUpdate()  Try  MyBase.Assign(item)  Dim source As RepositoryItemMyGridLookUpEdit = \_  CType(item, RepositoryItemMyGridLookUpEdit)  If source Is Nothing Then Return  Finally  EndUpdate()  End Try  End Sub  End Class  Public Class MyGridLookUpEdit  Inherits GridLookUpEdit  'The static constructor which calls the registration method  Shared Sub New()  RepositoryItemMyGridLookUpEdit.RegisterMyGridLookUpEdit()  End Sub  'Return the unique name  Public Overrides ReadOnly Property EditorTypeName() As String  Get  Return RepositoryItemMyGridLookUpEdit.MyGridLookUpEditName  End Get  End Property  'Override the Properties property  'Simply type-cast the object to the custom repository item type  <DesignerSerializationVisibility(DesignerSerializationVisibility.Content)> \_  Public Shadows ReadOnly Property Properties() As RepositoryItemMyGridLookUpEdit  Get  Return CType(MyBase.Properties, RepositoryItemMyGridLookUpEdit)  End Get  End Property  End Class | |