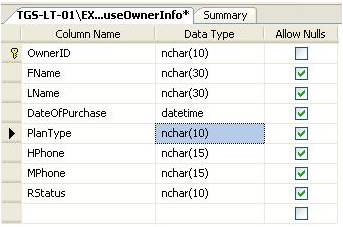
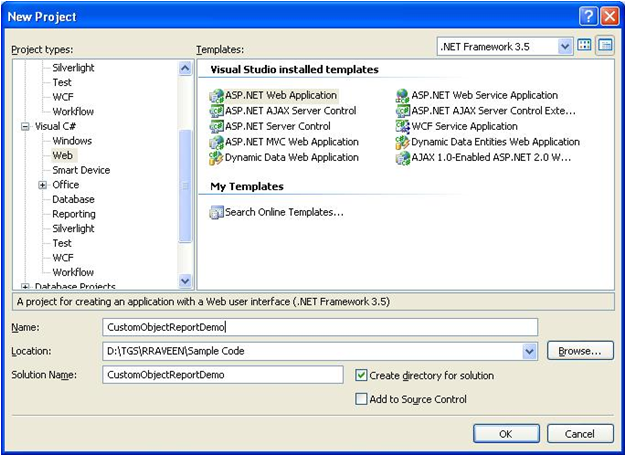
**Cómo generar un informe de Crystal usando la colección de objetos personalizados con ASP.NET**

Podemos usar la colección personalizada para enlazar con Gridview o cualquier otro control de enlace de datos. De la misma manera, podemos usar colecciones personalizadas para crear un informe de cristal. En este artículo, demostraremos cómo crear un informe utilizando una colección personalizada.

**Implementación:**  
  
aprendimos en el artículo anterior cómo crear un informe con un conjunto de datos en ASP.NET. Puede leer el artículo usando el enlace a continuación:  
  
<http://www.highoncoding.com/Articles/201_Creating_Crystal_Reports_Using_Typed_DataSet.aspx>  
  
Aquí primero necesitamos crear un objeto .NET para diseñar un informe, por lo tanto, debe diseñar una base de datos con una tabla.  
  
  
  
El nombre de la base de datos creada es "HouseRental" con la tabla "HouseOwnerInfo". Hay ocho columnas en “HouseOwnerInfo.  
  
Diseñemos un informe para esta tabla para obtener un informe resumido de los propietarios.  
Nota: El diseño de la base de datos es solo para fines de demostración.  
  
  
A continuación, debemos crear un proyecto ASP.NET con Visual Studio, abrir Visual Studio y hacer clic en Archivo -> nuevo -> Proyecto, luego seleccionar la aplicación web ASP.NET de la plantilla y darle el nombre como "CustomObjectReportDemo".  
   
  
  
Ahora necesitamos crear una clase personalizada llamada 'HouseOwnerInfo'. Ahora agregue los siguientes campos y propiedades a la clase.

|  |  |
| --- | --- |
| 01 | using System; |
| 02 | using System.Collections.Generic; | |

|  |  |  |
| --- | --- | --- |
| 03 | using System.Linq; | |
| 04 | using System.Web; |

|  |  |
| --- | --- |
| 05 |  |
| 06 | namespace CustomObjectReportDemo | |

|  |  |
| --- | --- |
| 07 | { |
| 08 | public class HouseOwnerInfo | |

|  |  |
| --- | --- |
| 09 | { |
| 10 | #region Variables | |

|  |  |
| --- | --- |
| 11 | int id; |
| 12 | string fName; | |

|  |  |
| --- | --- |
| 13 | string lName; |
| 14 | DateTime dateofPurchase; | |

|  |  |
| --- | --- |
| 15 | string hPhone; |
| 16 | string mPhone; |

|  |  |  |
| --- | --- | --- |
| 17 | string residencyStatus; | |
| 18 | #endregion |

|  |  |
| --- | --- |
| 19 |  |
| 20 | #region Cons | |

|  |  |  |
| --- | --- | --- |
| 21 | public HouseOwnerInfo() | |
| 22 | { |

|  |  |
| --- | --- |
| 23 |  |
| 24 | } | |

|  |  |  |
| --- | --- | --- |
| 25 | #endregion | |
| 26 |  |

|  |  |  |
| --- | --- | --- |
| 27 | #region Properties | |
| 28 | public int Id |

|  |  |
| --- | --- |
| 29 | { |
| 30 | get { return id; } | |

|  |  |  |
| --- | --- | --- |
| 31 | set { id = value; } | |
| 32 | } |

|  |  |  |
| --- | --- | --- |
| 33 | public string FirstName | |
| 34 | { |

|  |  |
| --- | --- |
| 35 | get { return fName; } |
| 36 | set { fName = value; } | |

|  |  |
| --- | --- |
| 37 | } |
| 38 | public string LastName | |

|  |  |
| --- | --- |
| 39 | { |
| 40 | get { return lName; } | |

|  |  |  |
| --- | --- | --- |
| 41 | set { lName = value; } | |
| 42 | } |

|  |  |  |
| --- | --- | --- |
| 43 | public DateTime DateofPurchase | |
| 44 | { |

|  |  |
| --- | --- |
| 45 | get { return dateofPurchase; } |
| 46 | set { dateofPurchase = value; } | |

|  |  |
| --- | --- |
| 47 | } |
| 48 | public string HomePhone | |

|  |  |
| --- | --- |
| 49 | { |
| 50 | get { return hPhone; } | |

|  |  |  |
| --- | --- | --- |
| 51 | set { hPhone = value; } | |
| 52 | } |

|  |  |  |
| --- | --- | --- |
| 53 | public string MobilePhone | |
| 54 | { |

|  |  |
| --- | --- |
| 55 | get { return mPhone; } |
| 56 | set { mPhone = value; } | |

|  |  |
| --- | --- |
| 57 | } |
| 58 | public string ResidencyStatus | |

|  |  |
| --- | --- |
| 59 | { |
| 60 | get { return residencyStatus; } | |

|  |  |  |
| --- | --- | --- |
| 61 | set { residencyStatus = value; } | |
| 62 | } |

|  |  |  |
| --- | --- | --- |
| 63 | #endregion | |
| 64 | } |

|  |  |
| --- | --- |
| 65 | } |

Hemos diseñado un objeto para HouseOwnerInfor. Hicimos este objeto a partir de las columnas de la tabla HouseOwnerInfo.  
  
Ahora tenemos una base de datos y un objeto .NET personalizado. Recuerde que nuestro escenario crea un informe utilizando el objeto .net personalizado. Para eso, necesitamos hacer una cosa más, que es la estructura XML para la tabla HouseOnwerInfo para diseñar el informe con el objeto .NET.  
Usando la estructura XML, necesita diseñar un informe de cristal. Ahora vaya al servidor SQL 2005 y abra una nueva ventana de Consulta, escriba las siguientes líneas.

|  |  |  |
| --- | --- | --- |
| 1 | SELECT [OwnerID] | |
| 2 | ,[FName] |

|  |  |
| --- | --- |
| 3 | ,[LName] |
| 4 | ,[DateOfPurchase] | |

|  |  |  |
| --- | --- | --- |
| 5 | ,[PlanType] | |
| 6 | ,[HPhone] |

|  |  |
| --- | --- |
| 7 | ,[MPhone] |
| 8 | ,[RStatus] | |

|  |  |
| --- | --- |
| 9 | FROM [HouseRental].[dbo].[HouseOwnerInfo] for XML AUTO,Elements |

Y ejecute la consulta, obtendrá la estructura de la tabla como el siguiente formato XML.

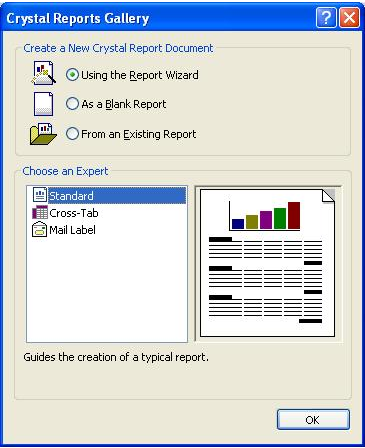
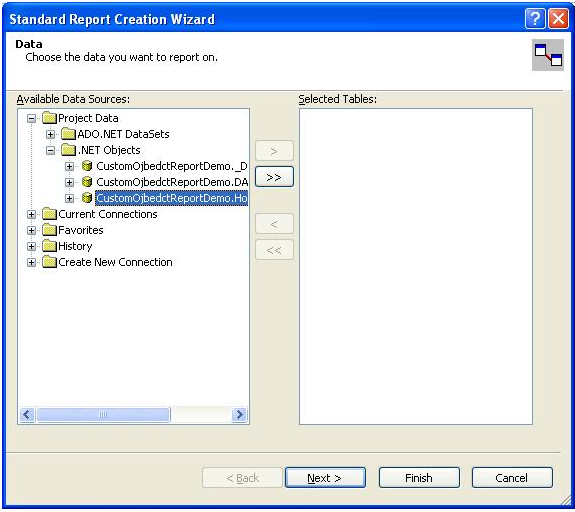
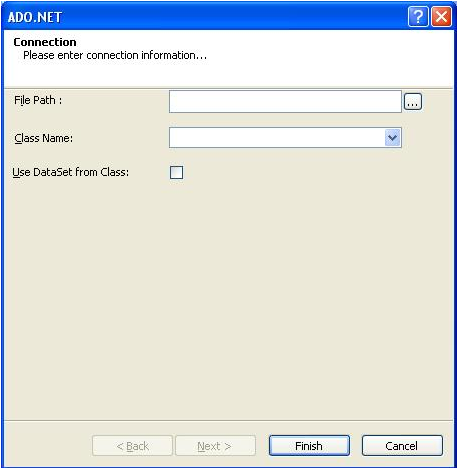
|  |  |
| --- | --- |
| 01 | <HouseOwnerInfo> |
| 02 | <OwnerID>0001      </OwnerID> | |

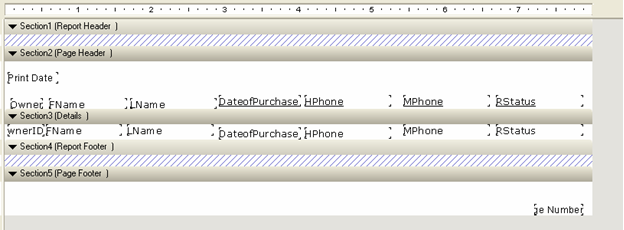
|  |  |
| --- | --- |
| 03 | <FName>John                          </FName> |
| 04 | <LName>Peter                         </LName> |

|  |  |  |
| --- | --- | --- |
| 05 | <DateOfPurchase>2009-03-20T00:00:00</DateOfPurchase> | |
| 06 | <PlanType>HDC       </PlanType> |

|  |  |
| --- | --- |
| 07 | <HPhone>678965443      </HPhone> |
| 08 | <MPhone>678965443      </MPhone> |

|  |  |  |
| --- | --- | --- |
| 09 | <RStatus>PR        </RStatus> | |
| 10 | </HouseOwnerInfo> |

A continuación, debemos copiar esto, pegarlo en el archivo xml y guardarlo. Luego agregue al proyecto.  
Comencemos a diseñar el informe.  
  
Haga clic con el botón derecho en el proyecto y seleccione agregar nuevo elemento, luego seleccione Informe de Crystal de la lista de plantillas de elementos y dé el nombre del informe como "CustomObjectReportDemo.rpt". Haga clic en el botón Aceptar.  
  
      
  
Esta es una ventana estándar para crear informes usando el asistente de diseño. Aquí puede dejar la selección predeterminada y hacer clic en el botón Aceptar para procesar el siguiente paso.  
   
  
  
Aquí, cuando haga clic en el botón ">>", verá una nueva ventana de la siguiente manera.  
  
 

Este es un lugar nuevo, cuando seleccionamos el informe del Objeto .NET, ya que necesitamos asignar un archivo XML al informe de diseño que tiene la misma estructura que el Objeto .NET. Para eso, haga clic en el botón '…' y busque la ruta del archivo XML. El nombre de la clase selecciona automáticamente el nodo de la vista de árbol en la ventana anterior de Crystal Report.  
Ahora asegúrese de haber seleccionado todos los campos del archivo XML. Luego continúe con el asistente y termínelo. Cuando haya terminado con todos los pasos correctamente, obtendrá el informe como se muestra a continuación.  
   
  
  
El siguiente paso es el más importante, ya que necesita realizar la colección de su objeto personalizado, que se construye utilizando datos de la base de datos.  
Por lo tanto, debe escribir el código de acceso a los datos para recuperar datos de la base de datos y crear la colección de HouseOwnerInfo.

Código:

|  |  |  |
| --- | --- | --- |
| 01 | public List<HouseOwnerInfo> GetAllHouseOnwersInfor() | |
| 02 | { |

|  |  |  |
| --- | --- | --- |
| 03 | try | |
| 04 | { |

|  |  |
| --- | --- |
| 05 |  |
| 06 | List<HouseOwnerInfo> collectionOfHouseOwners = new List<HouseOwnerInfo>(); | |

|  |  |  |  |
| --- | --- | --- | --- |
| 07 | using (SqlConnection connection = new SqlConnection(WebConfigurationManager.AppSettings["Connection"])) | | |
| 08 | | { |

|  |  |  |
| --- | --- | --- |
| 09 | | HouseOwnerInfo houseOwnerInfo; |
| 10 | if (connection.State!=System.Data.ConnectionState.Open) | | |

|  |  |
| --- | --- |
| 11 | { |
| 12 | connection.Open(); | |

|  |  |
| --- | --- |
| 13 | } |
| 14 | using (SqlCommand cmd = connection.CreateCommand()) | |

|  |  |
| --- | --- |
| 15 | { |
| 16 | cmd.CommandText = "Select \* from HouseOwnerInfo;"; | |

|  |  |
| --- | --- |
| 17 | cmd.CommandType = System.Data.CommandType.Text; |
| 18 | using (SqlDataReader reader = cmd.ExecuteReader()) |

|  |  |
| --- | --- |
| 19 | { |
| 20 | while (reader.Read()) | |

|  |  |
| --- | --- |
| 21 | { |
| 22 | houseOwnerInfo = new HouseOwnerInfo(); | |

|  |  |  |
| --- | --- | --- |
| 23 | if (reader["OwnerID"] != DBNull.Value) | |
| 24 | { |

|  |  |  |
| --- | --- | --- |
| 25 | houseOwnerInfo.OwnerID = int.Parse(reader["OwnerID"].ToString()); | |
| 26 | } |

|  |  |  |
| --- | --- | --- |
| 27 | if (reader["FName"] != DBNull.Value) | |
| 28 | { |

|  |  |  |
| --- | --- | --- |
| 29 | houseOwnerInfo.FName = reader["FName"].ToString(); | |
| 30 | } |

|  |  |  |
| --- | --- | --- |
| 31 | if (reader["LName"] != DBNull.Value) | |
| 32 | { |

|  |  |  |
| --- | --- | --- |
| 33 | houseOwnerInfo.LName = reader["LName"].ToString(); | |
| 34 | } |

|  |  |  |
| --- | --- | --- |
| 35 | if (reader["DateOfPurchase"] != DBNull.Value) | |
| 36 | { |

|  |  |  |
| --- | --- | --- |
| 37 | houseOwnerInfo.DateofPurchase = DateTime.Parse(reader["DateOfPurchase"].ToString()); | |
| 38 | } |

|  |  |  |
| --- | --- | --- |
| 39 | if (reader["HPhone"] != DBNull.Value) | |
| 40 | { |

|  |  |  |
| --- | --- | --- |
| 41 | houseOwnerInfo.HPhone = reader["HPhone"].ToString(); | |
| 42 | } |

|  |  |  |
| --- | --- | --- |
| 43 | if (reader["MPhone"] != DBNull.Value) | |
| 44 | { |

|  |  |  |
| --- | --- | --- |
| 45 | houseOwnerInfo.MPhone = reader["MPhone"].ToString(); | |
| 46 | } |

|  |  |  |
| --- | --- | --- |
| 47 | if (reader["RStatus"] != DBNull.Value) | |
| 48 | { |

|  |  |  |
| --- | --- | --- |
| 49 | houseOwnerInfo.MPhone = reader["RStatus"].ToString(); | |
| 50 | } |

|  |  |  |  |
| --- | --- | --- | --- |
| 51 | collectionOfHouseOwners.Add(houseOwnerInfo); | | |
| 52 | | } |

|  |  |  |
| --- | --- | --- |
| 53 | } | |
| 54 |  |

|  |  |  |
| --- | --- | --- |
| 55 | } | |
| 56 | } |

|  |  |
| --- | --- |
| 57 |  |
| 58 | return collectionOfHouseOwners; | |

|  |  |
| --- | --- |
| 59 | } |
| 60 | catch (Exception ex) | |

|  |  |  |
| --- | --- | --- |
| 61 | { | |
| 62 |  |

|  |  |  |
| --- | --- | --- |
| 63 | throw ex; | |
| 64 | } |

|  |  |
| --- | --- |
| 65 | } |

Aquí hemos escrito un código de acceso a datos simple que lee los datos del lector, los asigna a nuestro objeto personalizado (HouseOwnerInfo), luego finalmente lo agrega a la lista y regresa como una lista.  
  
Ahora tenemos la lista de houseOwnerInfo, el siguiente paso, necesita configurar su informe. Antes de eso, debe configurar el visor de informes de Crystal en su página como se muestra a continuación.  
  
**Código HTML:**

|  |  |
| --- | --- |
| 1 | <u><b> |
| 2 | </b></u><form id="form1" runat="server"> | |

|  |  |
| --- | --- |
| 3 | <div> |
| 4 | <CR:CrystalReportViewer ID="cryreportViewer" runat="server" AutoDataBind="true" /> | |

|  |  |
| --- | --- |
| 5 | </div> |
| 6 | </form> | |

Ahora hemos configurado el visor de informes de Crystal en la página web. Necesitamos cargar un informe en un documento de informe.

|  |  |
| --- | --- |
| 1 | ReportDocument report = new ReportDocument(); |
| 2 | report.Load(Server.MapPath("CustomObjectReportDemo.rpt")); | |

Nota: Aquí no es necesario configurar la información de inicio de sesión de la base de datos para reportdocument.  
Asigne la fuente de datos al documento de informe  
report.SetDataSource (new DataBaseManager (). GetAllHouseOnwersInfor ());  
Por último, configura el objeto de informe en el visor de informes para representar el informe en la página.  
Código completo para cargar, configurar la fuente de datos y representar el informe en la página.  
  
**Código:**

|  |  |  |
| --- | --- | --- |
| 01 | protected void Page\_Load(object sender, EventArgs e) | |
| 02 | { |

|  |  |  |
| --- | --- | --- |
| 03 | if (!IsPostBack) | |
| 04 | { |

|  |  |  |
| --- | --- | --- |
| 05 | BiulReport(); | |
| 06 | } |

|  |  |  |
| --- | --- | --- |
| 07 | } | |
| 08 |  |

|  |  |  |
| --- | --- | --- |
| 09 | private void BiulReport() | |
| 10 | { |

|  |  |  |
| --- | --- | --- |
| 11 | | ReportDocument report = new ReportDocument(); |
| 12 | report.Load(Server.MapPath("CustomObjectReportDemo.rpt")); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | report.SetDataSource(new DataBaseManager().GetAllHouseOnwersInfor()); | | |
| 14 | | cryreportViewer.ReportSource = report; |

|  |  |
| --- | --- |
| 15 | } |

Ahora cree la solución y ejecútela. Si no tiene ninguna excepción, obtendrá un resultado como:  
   
