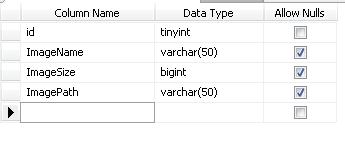
**[UPLOADING AND STORING IMAGE PATH TO DATABASE AND IMAGE TO FOLDER - PART 1](http://geekswithblogs.net/dotNETvinz/archive/2009/08/02/uploading-and-storing-image-path-todatabase-and--image-to.aspx" \o "Title of this entry.)**

I decided to write this example because this has been asked many times at the forums. In my previous article I have shown on how to Upload and Save the Images to Database, In this article I will show on how to upload and save the image to folder and path to database.

To get started, let’s create a simple database table for storing the Image information and path to the database. I this example I named the table as “ImageInfo” with the following fields below:



**Note:**I set the Id to auto increment so that the id will be automatically generated for every new added row in the table. To do this select the Column name “Id” and in the column properties set the “Identity Specification” to yes.

Now, in Visual Studio, let’s create a folder under the root application for storing the actual image. The folder structure would look something like this below:

Solution

  -Application Name

  -AppCode

  -AppData

  -**ImageStorage**   - //we will save the image in this folder

  -Default.aspx

  -web.config

After that we can now design our WebForm. For the simplicity of this demo, I just set up the form like below:

|  |
| --- |
| <html xmlns="http://www.w3.org/1999/xhtml">  <head runat="server">      <title>Untitled Page</title>  </head>  <body>      <form id="form1" runat="server">      <div>          <asp:FileUpload ID="FileUpload1" runat="server" />          <br />          <asp:Button ID="Button1" runat="server" Text="Save"onclick="Button1\_Click" />      </div>      </form>  </body>  </html> |

Simple right? I just contain a single Button and a FileUpload control. The next step is to set up our connection string in the web.config file. See below markup:

|  |
| --- |
| <connectionStrings>              <add name="DBConnection" connectionString="Data Source=SERVERNAME\SQLEXPRESS;Initial Catalog=SampleDB;Integrated Security=SSPI;" providerName="System.Data.SqlClient"/>  </connectionStrings> |

Now let’s go ahead and switch to code behind file of the form and create the method for Uploading and saving the Images. Here are the code blocks below:

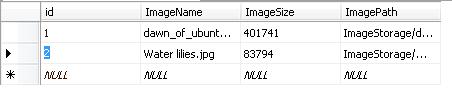
|  |
| --- |
| private void StartUpLoad()      {          //get the file name of the posted image          string imgName = FileUpload1.FileName.ToString();          //sets the image path          string imgPath = "ImageStorage/" + imgName;          //then save it to the Folder          FileUpload1.SaveAs(Server.MapPath(imgPath));            //get the size in bytes that          int imgSize = FileUpload1.PostedFile.ContentLength;            //validates the posted file before saving          if (FileUpload1.PostedFile != null && FileUpload1.PostedFile.FileName != "")          {              if (FileUpload1.PostedFile.ContentLength > 5120) // 5120 KB means 5MB              {                  Page.ClientScript.RegisterClientScriptBlock(typeof(Page), "Alert","alert('File is too big')", true);              }              else              {                  //save the file                  //Call the method to execute Insertion of data to the Database                  ExecuteInsert(imgName, imgSize, imgPath);                  Response.Write("Save Successfully!");              }          }      }        private string GetConnectionString()      {          //sets the connection string from your web config file. "DBConnection" is the name of your Connection String          returnSystem.Configuration.ConfigurationManager.ConnectionStrings["DBConnection"].ConnectionString;      }        private void ExecuteInsert(string name, int size, string path)      {            SqlConnection conn = new SqlConnection(GetConnectionString());          string sql = "INSERT INTO ImageInfo (ImageName, ImageSize, ImagePath) VALUES "                      + " (@ImgName,@ImgSize,@ImgPath)";          try          {                conn.Open();              SqlCommand cmd = new SqlCommand(sql, conn);              SqlParameter[] param = new SqlParameter[3];                param[0] = new SqlParameter("@ImgName", SqlDbType.NVarChar, 50);              param[1] = new SqlParameter("@ImgSize", SqlDbType.BigInt, 9999);              param[2] = new SqlParameter("@ImgPath", SqlDbType.VarChar, 50);                param[0].Value = name;              param[1].Value = size;              param[2].Value = path;                for (int i = 0; i < param.Length; i++)              {                  cmd.Parameters.Add(param[i]);              }                cmd.CommandType = CommandType.Text;              cmd.ExecuteNonQuery();          }          catch (System.Data.SqlClient.SqlException ex)          {              string msg = "Insert Error:";              msg += ex.Message;              throw new Exception(msg);          }          finally          {              conn.Close();          }      }        protected void Button1\_Click(object sender, EventArgs e)      {          StartUpLoad();      } |

Method Definitions:

**StartUpload()** – is a method where the we get the information of the posted file and save the actual image to the folder. It is also where we validate the upload file and the file size before we insert the file information in the database.

**ExecuteInsert()** – is a method that accepts three parameters which will be save in the database.

Running the code above will show a FileUpload control with a Button in the page for saving the image. The actual image will be stored in the folder that we have created above and the image information will be saved to database table as shown below:



That’s it! Hope you will find this example useful!

[**UPLOADING AND STORING IMAGE PATH TO DATABASE AND IMAGE TO FOLDER - PART 2 (DISPLAYING OF IMAGES)**](http://geekswithblogs.net/dotNETvinz/archive/2009/08/02/uploading-and-storing-image-path-todatabase-and-image-to-folder.aspx)

In my [previous example](http://geekswithblogs.net/dotNETvinz/archive/2009/08/02/uploading-and-storing-image-path-todatabase-and--image-to.aspx), we have learned on how to save the actual image to a folder and image path to the database. In this example, I’m going to show on how to display those images in a GridView and Repeater control.

To get started, let’s create a method for fetching the image information from the database. Here’s the code block below:

|  |
| --- |
| private DataTable GetData()      {          DataTable dt = new DataTable();          SqlConnection connection = newSqlConnection(GetConnectionString());            try          {              connection.Open();              string sqlStatement = "SELECT \* FROM ImageInfo";              SqlCommand sqlCmd = new SqlCommand(sqlStatement, connection);              SqlDataAdapter sqlDa = new SqlDataAdapter(sqlCmd);                sqlDa.Fill(dt);          }          catch (System.Data.SqlClient.SqlException ex)          {              string msg = "Fetch Error:";              msg += ex.Message;              throw new Exception(msg);          }          finally          {              connection.Close();          }            return dt;      } |

As you can see, the code above is very straight forward and self explanatory. Now since we already have a method for fetching the data from database then we can bind the result in a Data Representation control. First let’s use a GridView control for displaying the image.

In Visual Studio, grab a GridView control from the toolbox and place it in the webform. The GridView markup would look something like below:

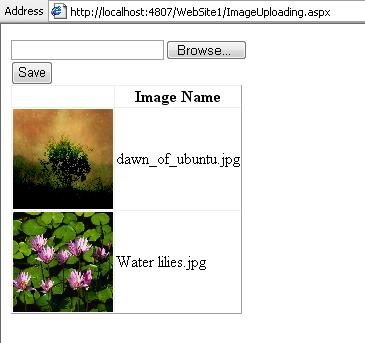
|  |
| --- |
| <asp:GridView ID="GridView1" runat="server"AutoGenerateColumns="false">          <Columns>              <asp:ImageField DataImageUrlField ="ImagePath" ControlStyle-Width="100px" ControlStyle-Height="100px"></asp:ImageField>              <asp:BoundField DataField="ImageName" HeaderText="Image Name"/>          </Columns>          </asp:GridView> |

As you can see, we used ImageField column for displaying the image using the path and used a BoundField column for displaying the image infornamtion.

Now, to make it work then let’s bind the GridView control at Page\_Load event. See below:

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)      {          if (!Page.IsPostBack)          {              DataTable dt = GetData();              if (dt.Rows.Count > 0)              {                  //Binding GridView                  GridView1.DataSource = dt;                  GridView1.DataBind();              }          }      } |

Running the code above will show something like below in the page:



Now, let’s go ahead and display the image in the Repeater control. Here’s the markup of Repeater control:

|  |
| --- |
| <asp:Repeater ID="Repeater1" runat="server">          <ItemTemplate>              <asp:Image ID="Image1" runat="server" Width="100px"Height="100px" ImageUrl='<%# Bind("ImagePath") %>' />              <asp:Label ID="Label1" runat="server" Text='<%# Bind("ImageName")%>'></asp:Label><br />          </ItemTemplate>          </asp:Repeater> |

Unlike GridView, a Repeater control doesn’t have an ImageField or BoundField columns so we use ItemTemplate so that we can add our own server control for displaying the image and the information as shown above.

At Page\_Load event, we can bind the Repeater control as what we did for binding the GridView. See the code block below:

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)      {          if (!Page.IsPostBack)          {              DataTable dt = GetData();              if (dt.Rows.Count > 0)              {                  Repeater1.DataSource = dt;                  Repeater1.DataBind();              }          }      } |

Running the code above will show something like below in the page:



That’s it! Hope you will find this example useful.