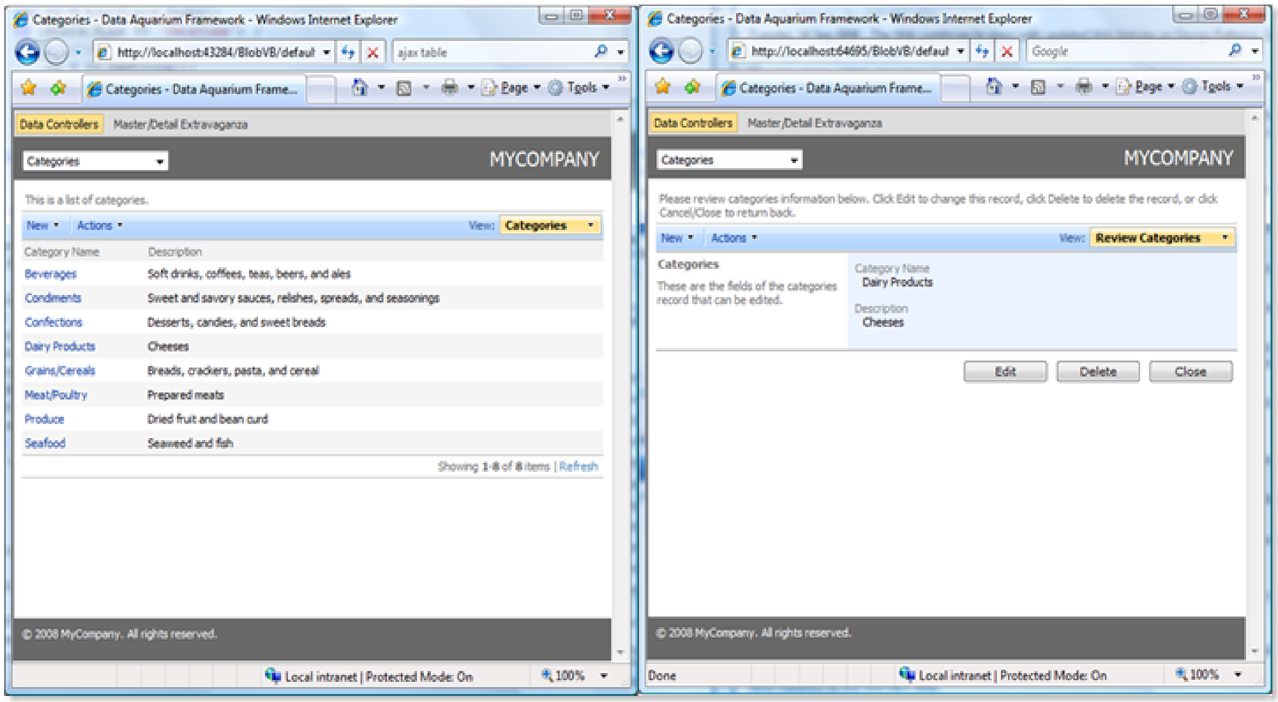
Working With Binary Large Objects (BLOB)

Modern web applications are frequently dealing with large data files of various formats commonly referred as **B**inary **L**arge **Ob**jects. In this article we will demonstrate how to implement BLOB support in Data Aquarium Framework and Aquarium Express applications.

Generate an Aquarium Express project from *Northwind* database and open the generated web site in *Visual Studio 2008* or *Visual Web Developer 2008 Express Edition*. Open the default page in a web browser and selected *Categories* data controller in the drop down at the top of the page.


Categories - Data Aquarium Framework - Windows Internet Explorer
http://localhost:43284/BlobVB/default
Data Controllers
Master/Detail Extravaganza
Categories
MYCOMPANY
New
Actions
View:
Categories
Local intranet | Protected Mode: On
100%
http://localhost:64695/BlobVB/default
Review Categories
Edit
Delete
Close
Done

This is a list of categories.

Category Name	Description
Beverages	Soft drinks, coffees, teas, beers, and ales
Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
Confections	Desserts, candies, and sweet breads
Dairy Products	Cheeses
Grains/Cereals	Breads, crackers, pasta, and cereal
Meat/Poultry	Prepared meats
Produce	Dried fruit and bean curd
Seafood	Seaweed and fish

Showing 1-8 of 8 items | Refresh

© 2008 MyCompany. All rights reserved.

Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

Categories

These are the fields of the categories record that can be edited.

Category Name
Dairy Products

Description
Cheeses

© 2008 MyCompany. All rights reserved.

The standard *grid* and *form* views are only displaying product *category name* and *description* data fields. Future versions of Data Aquarium Framework will provide automatic support for BLOB fields. We will enhance the presentation that can be generated today to include a category picture and support uploading of new pictures as well.

Image Handler

We will start by adding a generic handler *Blob.ashx* to the root of our web site. The purpose of this handler is to deliver a category picture to a web browser in response to a request that specifies a category ID.

Here is how the handler works.

First it verifies that there is a parameter *CategoryID* in the requested URL that represents a number. If so then a picture is retrieved by *SQL* query executed with the help of *MyCompany.Data.SqlText* utility class generated automatically for every project. An attempt is made to determine the binary format of the picture. We are trying to load the picture by calling the method *FromStream* of *System.Drawing.Image* class. If attempt has failed then we will try to read the picture again with offset of 78. This is only necessary if you are working with *Northwind* database, which supports one of the older proprietary formats that were introduced by *Microsoft* a long time ago.

Now it is time to return the picture to a requesting browser. We identify the content type of the picture by looking up a dictionary of image content types created by static (shared) constructor of the generic handler. Then we write the picture into the output stream.

The last and important step is to tell the requesting browser not to cache the output produced by handler.

Image handler *Blog.ashx* written in C#:

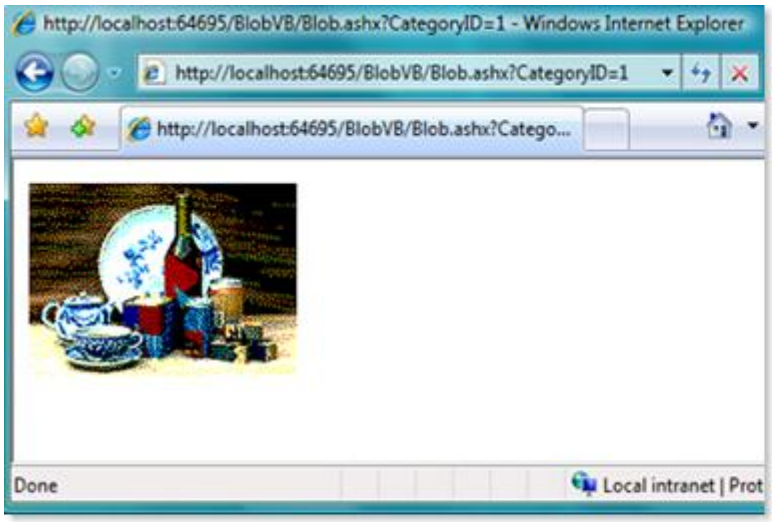
```
<%@ WebHandler Language="C#" Class="Blob" %>

using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.Collections.Generic;
using System.IO;
using System.Web;
using MyCompany.Data;

public class Blob : IHttpHandler
{
    public void ProcessRequest(HttpContext context)
    {
        int categoryId;
        if (Int32.TryParse(context.Request.Params["CategoryID"], out
categoryId))
            using (SqlText getPicture = new SqlText(
                "select Picture from Categories where CategoryId =
@CategoryId"))
            {
                getPicture.AddParameter("@CategoryId", categoryId);
                if (getPicture.Read() &&
!DBNull.Value.Equals(getPicture["Picture"]))
                {
                    byte[] picture = (byte[])getPicture["Picture"];
                    int offset = 0;
                    Image img = null;
                    try
                    {
                        img = Image.FromStream(new MemoryStream(picture));
                    }
                    catch (Exception)
                    {
                        offset = 78; // correction for Northwind image format
                        img = Image.FromStream(
                            new MemoryStream(picture, offset, picture.Length
- offset));
                    }
                    context.Response.ContentType =
ImageFormats[img.RawFormat.Guid];
```

```
                    context.Response.OutputStream.Write(picture, offset,
                        picture.Length - offset);
                }
            }
        context.Response.Cache.SetCacheability(HttpCacheability.NoCache);
    }

    public bool IsReusable { get { return false; } }

    public static SortedDictionary<Guid, string> ImageFormats;

    static Blob()
    {
        ImageFormats = new SortedDictionary<Guid, string>();
        ImageFormats.Add(ImageFormat.Bmp.Guid, "image/bmp");
        ImageFormats.Add(ImageFormat.Emf.Guid, "image/emf");
        ImageFormats.Add(ImageFormat.Exif.Guid, "image/exif");
        ImageFormats.Add(ImageFormat.Gif.Guid, "image/gif");
        ImageFormats.Add(ImageFormat.Jpeg.Guid, "image/jpeg");
        ImageFormats.Add(ImageFormat.Png.Guid, "image/png");
        ImageFormats.Add(ImageFormat.Tiff.Guid, "image/tiff");
        ImageFormats.Add(ImageFormat.Wmf.Guid, "image/wmf");
    }
}
```

Image handler *Blob.ashx* written in *VB.NET*:

```
<%@ WebHandler Language="VB" Class="Blob" %>

Imports System
Imports System.Web
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Collections.Generic
Imports System.IO
Imports MyCompany.Data

Public Class Blob : Implements IHttpHandler

    Public Sub ProcessRequest(ByVal context As HttpContext) _
        Implements IHttpHandler.ProcessRequest
        Dim categoryId As Integer
        If (Integer.TryParse(context.Request.Params("CategoryID"),
categoryId)) Then
            Using getPicture As SqlText = New SqlText( _
                "select Picture from Categories where CategoryId =
@CategoryId")
                getPicture.AddParameter("@CategoryID", categoryId)
                If (getPicture.Read() AndAlso Not
DBNull.Value.Equals(getPicture("Picture"))) Then
                    Dim picture As Byte() = CType(getPicture("Picture"),
Byte())
                    Dim offset As Integer
                    Dim img As Image = Nothing
                    Try
                        img = Image.FromStream(New MemoryStream(picture))
                    Catch ex As Exception
                        offset = 78
                        img = Image.FromStream( _
                            New MemoryStream(picture, offset, picture.Length
- offset))
                    End Try
                    context.Response.ContentType =
ImageFormats(img.RawFormat.Guid)
                    context.Response.OutputStream.Write(picture, offset,
picture.Length - offset)
                End If
            End Using
        End If
        context.Response.Cache.SetCacheability(HttpCacheability.NoCache)
    End Sub

    Public ReadOnly Property IsReusable() As Boolean _
        Implements IHttpHandler.IsReusable
        Get
            Return False
        End Get
    End Property

    Public Shared ImageFormats As SortedDictionary(Of Guid, Object)

    Shared Sub New()
        ImageFormats = New SortedDictionary(Of Guid, Object)
        ImageFormats.Add(ImageFormat.Bmp.Guid, "image/bmp")
        ImageFormats.Add(ImageFormat.Emf.Guid, "image/emf")
        ImageFormats.Add(ImageFormat.Exif.Guid, "image/exif")
        ImageFormats.Add(ImageFormat.Gif.Guid, "image/gif")
        ImageFormats.Add(ImageFormat.Jpeg.Guid, "image/jpeg")
        ImageFormats.Add(ImageFormat.Png.Guid, "image/png")
        ImageFormats.Add(ImageFormat.Tiff.Guid, "image/tiff")
        ImageFormats.Add(ImageFormat.Wmf.Guid, "image/wmf")
    End Sub

End Class
```

If you navigate to this handler in a web browser then a blank page will be displayed. Specify *CategoryID* parameter in a browser URL and a corresponding image will show up.


http://localhost:64695/BlobVB/Blob.ashx?CategoryID=1 - Windows Internet Explorer
http://localhost:64695/BlobVB/Blob.ashx?CategoryID=1
http://localhost:64695/BlobVB/Blob.ashx?Catego...
Done
Local intranet | Prot

Displaying Picture From Database in Form View

Create page *Categories.aspx* in the root of your web site and enter the page markup as follows.

```
<%@ Page Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false"
    CodeFile="Categories.aspx.vb" Inherits="Categories" Title="Categories" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="Header1Placeholder"
runat="Server">
    Categories
</asp:Content>
<asp:Content ID="Content3" ContentPlaceHolderID="Header2Placeholder"
runat="Server">
    Northwind
</asp:Content>
<asp:Content ID="Content4" ContentPlaceHolderID="BodyPlaceholder"
runat="Server">
    <div id="CategoryList" runat="server" />
    <aquarium:DataViewExtender ID="CategoriesExtender" runat="server"
```

```
        Controller="Categories" TargetControlID="CategoryList" />
    <div id="Categories_editForm1" style="display: none">
        {CategoryName}
        <br />
        {Description}
        <br />
        <a href="Blob.ashx?CategoryID={CategoryID:d}" target="_blank">
            <img src="Blob.ashx?CategoryID={CategoryID:d}" alt="Category"
                style="height: 120px; border: 0px" /></a>
    </div>
</asp:Content>
```

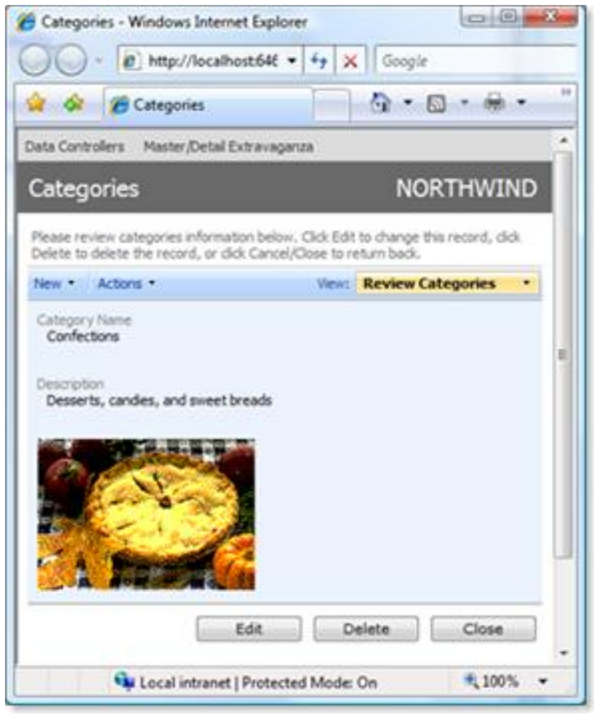
This page is based on standard master page *MasterPage.master* included in the generated project. Our page declares *CategoryList* element as a placeholder for categories and *CategoriesExtender,* which is responsible for making sure that the view will be rendered there.

Then there is also template *Categories_editForm1* for *editForm1* view that must be applied when the view is to be rendered on this page by data controller *Categories*. The template is not displayed in the page. Read about custom form templates to better understand form and grid template capabilities of the framework.

Template instructs client-side JavaScript component *Web.DataView* to substitute *CategoryName* and *Description* with corresponding markup that would otherwise be rendered if there was no template.

There are also two references to *CategoryID* that are included into URLs pointing to *Blob.ashx* generic handler. Both references are formatted as *CategoryID:d*. The letter following the colon is a format string that will force Web.DataView to take the corresponding field value and format the value with java script String.format method call, which looks like *String.format('{0:d}',v)* where *v* is a value. This inserts only the string presentation of category ID instead of the complicated markup that would have been rendered otherwise. That makes it possible to use ID in the URL that is passed to the action handler.

Open *Categories.aspx* in a web browser and select any category. A view similar to the one in the screen shot will be presented. Click on the category picture and a full size image will open in a new web browser window.


Categories - Windows Internet Explorer
http://localhost:64
Google
Categories
Data Controllers
Master/Detail Extravaganza
Edit
Delete
Close
Local intranet | Protected Mode: On
100%

Categories NORTHWIND

Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

New • Actions • View: Review Categories

Category Name
Confections

Description
Desserts, candies, and sweet breads

If your application is not dealing with images then you can get rid of the *img* element in the template and have it replaced with text *click to download* or some other phrase.

Uploading Picture to Database

Let's implement support for picture uploading, which will not be any different if you are dealing with other file formats.

Add web form *Uploader.aspx* to the root of our web site and make the following changes.

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Uploader.aspx.vb"
    Inherits="Uploader" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head id="Head1" runat="server">
    <title>Uploader</title>
    <style type="text/css">
        body, input
        {
            font-family: Tahoma;
            font-size: 8.5pt;
            margin: 2px;
        }
    </style>
    <script type="text/javascript">
    function uploadSuccess(){
        parent.window.Web.DataView.showMessage(
          '<b>Congratulations!</b> Category picture has been uploaded
successfully.')
        parent.window.Web.DataView.find('CategoriesExtender').goToPage(-1);
    }
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <div id="StartUpload" runat="server">
            Click
            <asp:LinkButton ID="ShowUploadControls" runat="server"
Text="here"
                OnClick="ShowUploadControls_Click" />
            to upload a category image.
        </div>
        <div id="UploadControls" runat="server" visible="false">
            <asp:Button ID="Submit" runat="server" Text="Submit"
                OnClick="Submit_Click" />
            <asp:FileUpload ID="FileUpload1" runat="server" Width="300px" />
        </div>
    </div>
    </form>
</body>
</html>
```

The body of the form has two *div* elements. The first contains a link that is supposed to display upload controls to user when clicked. The upload controls are hidden by default. These controls are *FileUpload1* control and *Submit* button. There is also a java script function *uploadSuccess* that we will discuss a little bit later.

Here is how a *C#* code-behind of the page looks:

```
using System;
using System.Web;
using MyCompany.Data;

public partial class Uploader : System.Web.UI.Page
{
    protected void ShowUploadControls_Click(object sender, EventArgs e)
    {
        UploadControls.Visible = true;
        StartUpload.Visible = false;
    }
    protected void Submit_Click(object sender, EventArgs e)
    {
        if (FileUpload1.HasFile)
            using (SqlText updatePicture = new SqlText(
                "update Categories set Picture=@Picture where CategoryID =
@CategoryID"))
            {
                updatePicture.AddParameter("@Picture",
FileUpload1.FileBytes);
```

```
                updatePicture.AddParameter("@CategoryID",
Request.Params["CategoryID"]);
                updatePicture.ExecuteNonQuery();
            }
        UploadControls.Visible = false;
        StartUpload.Visible = true;
        ClientScript.RegisterClientScriptBlock(typeof(Uploader), ClientID,
            "uploadSuccess();", true);
    }
}
```

VB.NET version of the same code-behind class is presented here:

```
Imports System
Imports System.Web
Imports MyCompany.Data

Partial Class Uploader
    Inherits System.Web.UI.Page

    Protected Sub ShowUploadControls_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles ShowUploadControls.Click
        UploadControls.Visible = True
        StartUpload.Visible = False
    End Sub

    Protected Sub Submit_Click(ByVal sender As Object, _
      ByVal e As System.EventArgs) Handles Submit.Click
        If (FileUpload1.HasFile) Then
            Using updatePicture As SqlText = New SqlText( _
              "update Categories set Picture=@Picture " + _
              "where CategoryID = @CategoryID")
                updatePicture.AddParameter("@Picture", FileUpload1.FileBytes)
                updatePicture.AddParameter("@CategoryID",
Request.Params("CategoryID"))
                updatePicture.ExecuteNonQuery()
            End Using
            UploadControls.Visible = False
            StartUpload.Visible = True
            ClientScript.RegisterClientScriptBlock( _
              Me.GetType(), ClientID, "uploadSuccess();", True)
        End If
    End Sub
End Class
```

Event handler *ShowUploadControls_Click* simply displays *Submit* button and *FileUpload1* file upload control.

Event handler *Submit_Click* writes the submitted file to the database with the help of utility class *MyCompany.Data.SqlText* that we have used previously. You can use your own favorite data access controls including standard *ADO.NET* classes. Then the visibility of web controls is reversed. The last step is to render a call to java script function *uploadSuccess* defined in the markup of *Uploader.aspx* web form. This method will execute as soon as the page is loaded in a browser window.

Let's take a closer look at this function.

```
<script type="text/javascript">
function uploadSuccess(){
    parent.window.Web.DataView.showMessage(
      '<b>Congratulations!</b> Category picture has been uploaded
successfully.')
    parent.window.Web.DataView.find('CategoriesExtender').goToPage(-1);
}
</script>
```

This function reaches out to a parent of a browser window and uses *Web.DataView.showMessage* method. Then it finds *CategoriesExtender* java script component and asks it to refresh itself.

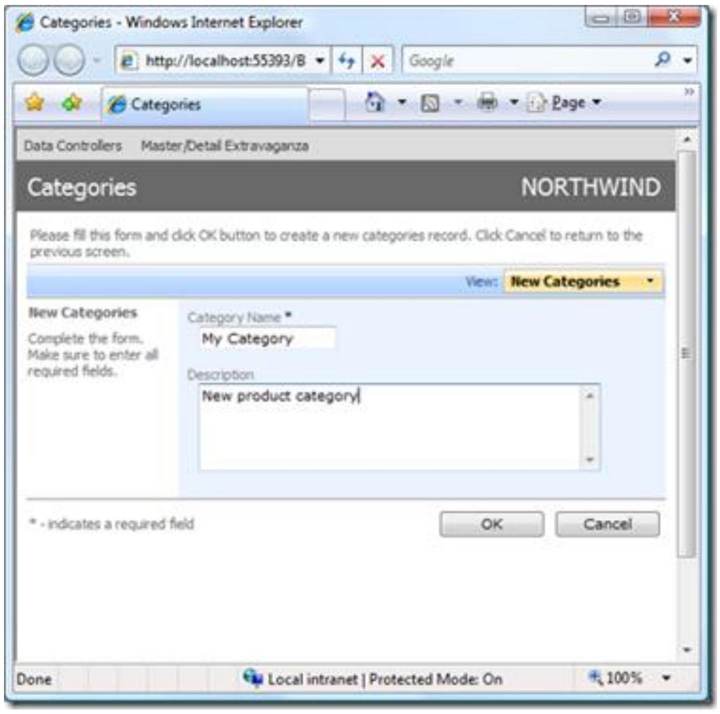
If you do open this page in a web browser then there is no parent window and the script will report an error when you upload a file. We intend to use this script in an *iframe* element that we will incorporate into the *Categories_editForm1* template to provide smooth user experience.

Let's return back to *Categories.aspx* for a minute and change the template for *editForm1* as follows.

```
<div id="Categories_editForm1" style="display: none">
    {CategoryName}
    <br />
    {Description}
    <br />
    <a href="Blob.ashx?CategoryID={CategoryID:d}" target="_blank">
        <img src="Blob.ashx?CategoryID={CategoryID:d}" alt="Category"
            style="height: 120px; border: 0px" /></a>
    <iframe src="Uploader.aspx?CategoryID={CategoryID:d}"
        style="width: 400px; height: 30px; margin-top: 4px"
        frameborder="0" scrolling="no"></iframe>
</div>
```

We have inserted *iframe* element at the end of the template. The source of the *iframe* is referring to *Uploader.aspx* and specifies a category ID in the URL of the *src* attribute.

Open *Categories.aspx* in a web browser and start creating new category.



Please fill this form and click OK button to create a new categories record. Click Cancel to return to the previous screen.

New Categories

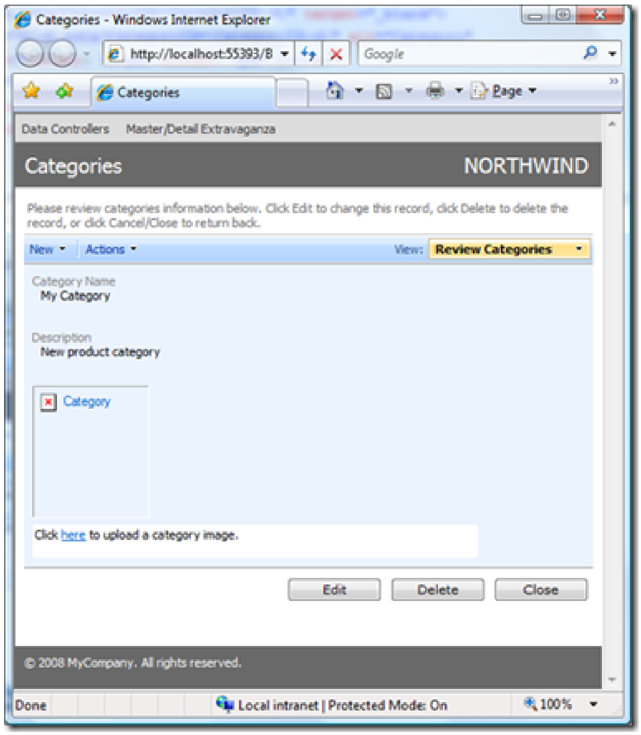
Complete the form. Make sure to enter all required fields.

Category Name *: My Category

Description: New product category

* - indicates a required field

View *createForm1* is presented. Enter category name and description and click *OK* button. Select the new category in the *grid* view. There is no picture defined for *My Category* yet, which will result in a missing image icon displayed under category description.



Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

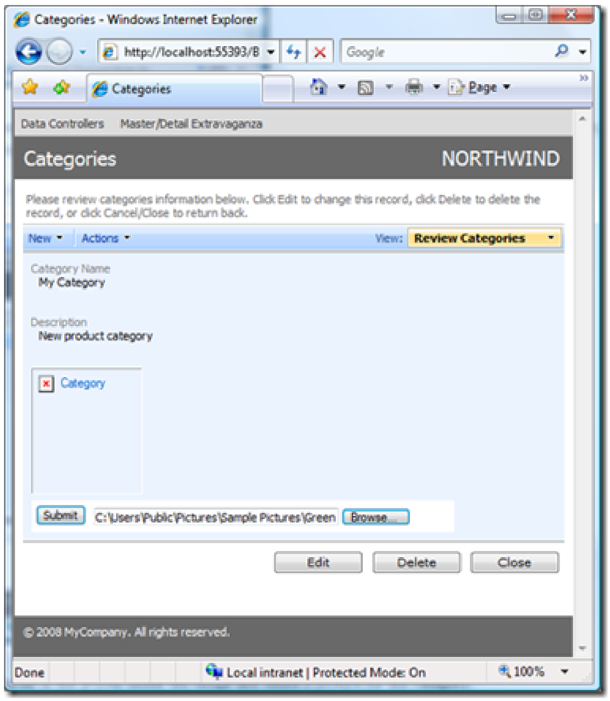
Category Name: My Category

Description: New product category

Click here to upload a category image.

© 2008 MyCompany. All rights reserved.

Click on the link displayed in the *iframe* under the image. *Submit* button and *FileUpload1* are visible now. Select a picture for the category.



Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

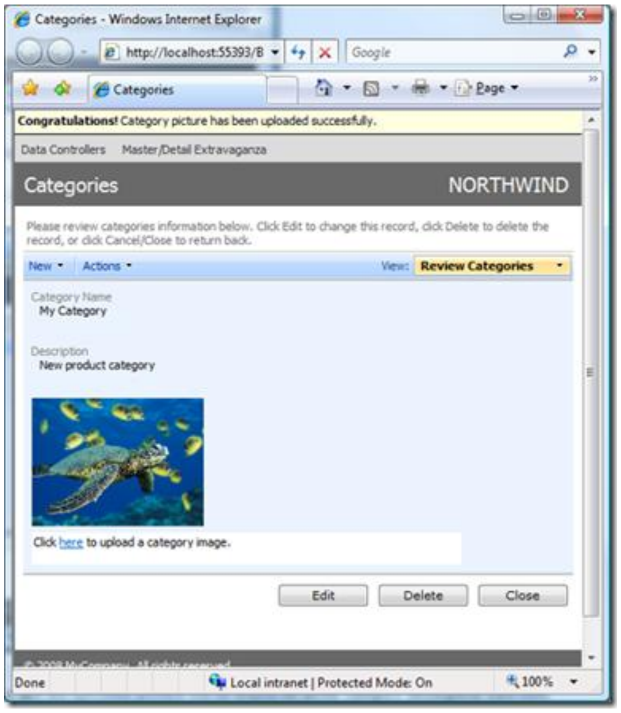
Category Name
My Category

Description
New product category

C:\Users\Public\Pictures\Sample Pictures\Green

© 2008 MyCompany. All rights reserved.

Click on *Submit* button and the uploaded picture will be displayed under category description data field.



Congratulations! Category picture has been uploaded successfully.

Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

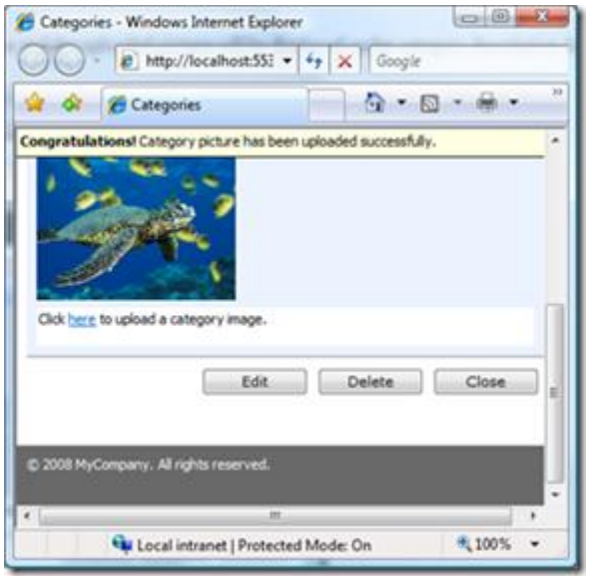
Category Name
My Category

Description
New product category

Click here to upload a category image.

The picture has been stored to the database by *Uploader.aspx* and retrieved by *Blob.ashx* generic handler when *CategoriesExtender* instance of java script *Web.DataView* class has refreshed presentation in response to *uploadSuccess* java script function call.

Also, notice a yellow bar at the top of the page. This message has been also displayed by *uploadSuccess*, which has taken advantage of *Web.DataView.showMessage* method that creates a static message bar that stays at the top of the page even if you scroll the page in a browser.



Congratulations! Category picture has been uploaded successfully.

Click here to upload a category image.

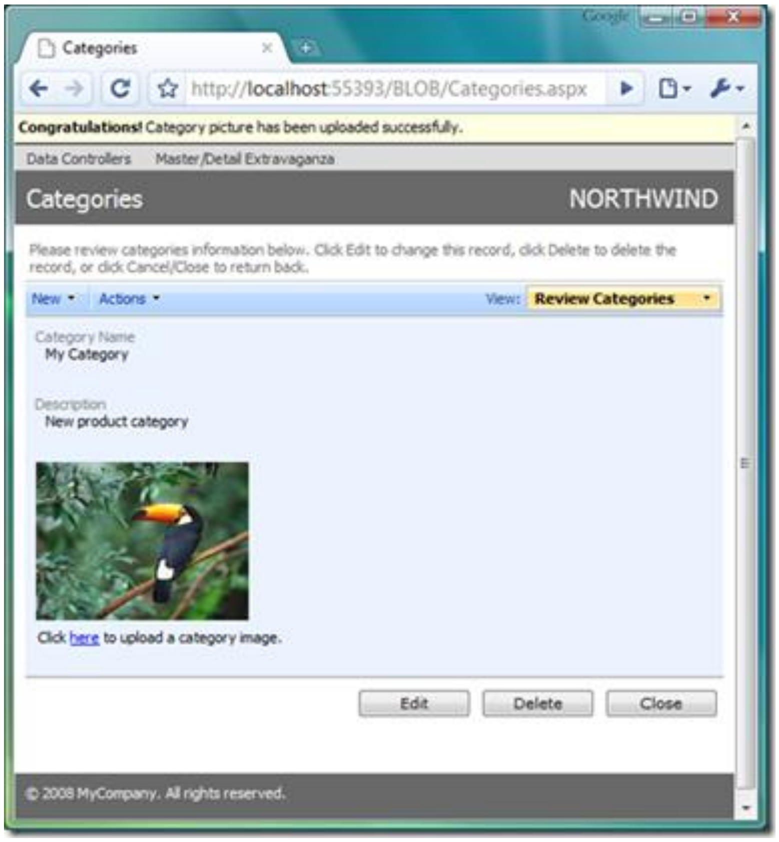
© 2008 MyCompany. All rights reserved.

Categories - Windows Internet Explorer
http://localhost:55
Google
Categories
Edit
Delete
Close
Local intranet | Protected Mode: On
100%

Any subsequent action in user interface will hide the message bar automatically. The same facility is used by *Web.DataView* component to report data update errors or any other important messages that occur in a lifecycle of the views.

Does This Work With All Browsers?

This technique does work with all modern browsers. Here is the same page displayed in *Google Chrome*.



Congratulations! Category picture has been uploaded successfully.

Categories NORTHWIND

Please review categories information below. Click Edit to change this record, click Delete to delete the record, or click Cancel/Close to return back.

Category Name
My Category

Description
New product category

Click here to upload a category image.

© 2008 MyCompany. All rights reserved.