Image uploads

Textbox.io handles uploads of local images by embedding them into the content (as a base64 data uri), then extracting them on the server on PostBack.

When each image is extracted, its binary data is passed to the ImageSave event handler. You can implement this event handler to write the image to disk, database or other storage.

The signature of the ImageSave event handler is as follows:

```
TextboxioControl.ImageSaveAction Editor1_ImageSave(string imageType, byte[] imageBytes);
```

Your event handler must return one of the three TextboxioControl.ImageSaveAction values:

a. ImageSaveAction.ReplaceUrl(serverImageUrl)
   This indicates that you have successfully saved the image, and that Textbox.io for ASP.NET should replace the embedded image with the new image URL.
b. ImageSaveAction.KeepEmbeddedImage()
   This indicates to Textbox.io that it should keep the embedded image in the content. You may wish to return this value if image save failed.
c. ImageSaveAction.RemoveImage()
   This indicates to Textbox.io that it should remove the embedded image from the content. You may wish to return this value if image save failed.

Handling errors

If any errors occur while extracting the local images, the ImageSaveError event handler is called. You may wish to implement this event handler to display an error in your application.

The signature for this handler is as follows:

```
void Editor1_ImageSaveError(IList<TextboxioControl.ImageProcessingError> errors);
```

ImageProcessingError has 3 fields (all read-only):

Field Name	Type	Description
Content	string	The HTML content from the editor, before image extraction.
ErrorContent	string	Content specific to the error type: For UnableToParseHtmlDocument: Empty string. For InvalidBase64Data: The "src" attribute of the image
ErrorType	ImageProcessingErrorType	UnableToParseHtmlDocument or InvalidBase64Data

Example

This example is from the ImageUpload.aspx.cs file in the demo project in the Textbox.io for ASP.NET distribution.

Example

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using TextboxioControl;
public partial class examples_ImageUpload : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e) {
    }
    private static Random rand = new Random();
    protected TextboxioControl.ImageSaveAction Editor1_ImageSave(string imageType, byte[] imageBytes) {
        // Generate a random filename for the image
        string filename = "upload/" + rand.Next().ToString() + "." + imageType;
        // Write the image file to disk - you could write to a db or whatever storage you wish.
        File.WriteAllBytes(Server.MapPath(filename), imageBytes);
        // Tell textbox.io to replace the embedded image with the server image filename
        return ImageSaveAction.ReplaceUrl(filename);
        // Alternatively, if the image failed to save, you may wish to remove the image...
        // return ImageSaveAction.RemoveImage;
        // ... or keep the embedded image in the document.
        // return ImageSaveAction.KeepEmbeddedImage;
    }
    protected void Editor1_ImageSaveError(IList<TextboxioControl.ImageProcessingError> errors) {
        // This handler is called when certain errors occur - e.g. if the embedded image is corrupt.
        // In this sample app, we're going to display any of these errors in a label.
        // To see this in action, add a local image to textbox.io, go to code view, then change the base64
image data
        // to something invalid
        string msg = "Errors:\n";
        foreach (ImageProcessingError e in errors) {
            msg += e.ErrorType + ": " + e.ErrorContent;
        }
        Errors.Text = msg;
    }
    protected void Button1_Click(object sender, EventArgs e) {
        SavedContent.Text = Editor1.Content;
    }
}
```

Can I upload using a normal HTTP upload handler?

i.e. using the "Uploading Local Images" instructions from Handling Local Images in the Textbox.io JavaScript SDK documentation.

No. This is not supported by Textbox.io for ASP.NET.

The main reason is that, with the asynchronous image uploads, an integrator must wait for image uploads to finish when posting a form. It is difficult to integrate this with the form submission in the ASP.NET framework.