



SharePoint Authenticated Content WebPart
Formation Technology, 2006

SharePoint Authenticated Content WebPart	2
What is it?.....	2
Problem	2
Environment.....	2
Technology	2
WebPart Properties.....	2
Placeholders.....	3
Rendering WebPart Output.....	4
Installing the WebPart	4
About Formation Technology.....	5

SharePoint Authenticated Content WebPart

What is it?

A SharePoint Portal Server Webpart that can publish content depending on a user's authentication.

This tool was developed by Formation Technology, Melbourne Australia.

Problem

A customer was running a single portal accessible via 2 URL's: One being authenticated, the other being unauthenticated. They need a simple way to move users into an authenticated state without change the URL.

They requested a "sign-in" button for the unauthenticated user. Once authenticated the content would then change to a personalised welcome message.

Environment

The WebPart is written for SharePoint Portal Server 2003 SP2. The WebPart requires access to the SharePoint API and so must run as a trusted assembly. It therefore is installed in to the GAC to avoid security issues.

The WebPart is installed using stsadm.exe as part of the SharePoint package. Typically found in: \Program Files\Common Files\Microsoft Shared\web server extensions\60\BIN

Technology

It is important to note that you can not write WebParts using Visual Studio 2005 as SharePoint Portal Server has very limited compatibility with the .NET Framework 2.0.

The WebPart was written as a class library using the SharePoint SDK extension for Visual Studio 2003. The extension is a new project type for creating a WebPart library. The library will include a sample WebPart cs file, a dwp file and a manifest file. Found here:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=CAC3E0D2-BEC1-494C-A74E-75936B88E3B5&displaylang=en>

You can download the SharePoint 2003 SDK here:

<http://www.microsoft.com/downloads/details.aspx?familyid=aa3e7fe5-dae-4d10-980f-789b827967b0&displaylang=en>

General information about the structure of a WebPart can be found here:

<http://www.devx.com/dotnet/Article/17518/0/page/2>

WebPart Properties

Only two properties are included on the WebPart. One is for entering the html to display when the user is authenticated and other is for entering the html to display when the user is

unauthenticated. There is no provision for sharing or reusing html between the two properties. You will just need to duplicate any common html.

```
[Browsable(true),
    Category("Authentication Content"),
    DefaultValue(defaultText),
    WebPartStorage(Storage.Shared),
    FriendlyName("Authenticated HTML"),
    Description("Insert HTML you wish to appear when the user is in an
authenticated state.")]
public string AuthenticatedHtml {
    get {
        return authenticatedHtml;
    }
    set {
        authenticatedHtml = value;
    }
}

[Browsable(true),
    Category("Authentication Content"),
    DefaultValue(defaultText),
    WebPartStorage(Storage.Shared),
    FriendlyName("Un-Authenticated HTML"),
    Description("Insert html you wish to appear when the user is in an unauthenticated
state.")]
public string UnAuthenticatedHtml {
    get {
        return unAuthenticatedHtml;
    }
    set {
        unAuthenticatedHtml = value;
    }
}
```

Figure 1

These two properties simply store the entered html as a string. Using the SharePoint properties attributes you can control how the property is presented to the administrator when adding the WebPart to the SharePoint page.

The value for the Category attribute can be anything, and does not need to be created elsewhere. FriendlyName is the name of the property as it appears in the SharePoint administration UI.

Placeholders

There are three placeholders you can use to add dynamic value's in to the authenticated HTML content. These placeholders will not work for the unauthenticated HTML (by design, not limitation).

The placeholders are:

- %N
Replaced with the currently authenticated user's display name (ie. Their full name)
- %L
Replaced with the currently authenticated user's login name (ie. ACME\Sam)
- %E
Replaced with the currently authenticated user's email address.

Note that with data such as the display name or email address it must have first been entered in to SharePoint or imported from the Active Directory.

Rendering WebPart Output

To render the WebPart output a check is first made to determine if the current user is authenticated.

If the user is not authenticated then the unauthenticated HTML content is directly outputted. If the user is authenticated then a call is made to the SharePoint API to retrieve the instance of the current user, represented in the object SPUser. SPUser provides the Name, LoginName and Email properties we require to replace the placeholders with the correct values. If the value is not in the SPUser then the placeholder will be removed, in it's place nothing will be inserted.

```
protected override void RenderWebPart(HtmlTextWriter output) {
    if (Context.User.Identity.IsAuthenticated){
        SPUser myUser = SPControl.GetContextWeb(Context).CurrentUser;
        authenticatedHtml = authenticatedHtml.Replace("%N", myUser.Name);
        authenticatedHtml = authenticatedHtml.Replace("%L", myUser.LoginName);
        authenticatedHtml = authenticatedHtml.Replace("%E", myUser.Email);

        output.Write(authenticatedHtml);
    }
    else
        output.Write(unAuthenticatedHtml);
}
```

Figure 2

Installing the WebPart

While there a few (manual) ways to install a webpart the easiest is to package the part in a CAB file and use stsadm.exe to install the part. To do this create a CAB package setup project and insert the Content Files output and Primary output for your WebPart class library. This ensures your DLL, DWP and XML manifest are included in the CAB file.

Verify that the "Build Action" for the DWP file and XML manifest file are set to "Content". Do this by selecting the file in the Solution Explorer and viewing it's properties in the properties window.

Build the CAB project and copy the outputted CAB file to your SharePoint server. The path to stsadm.exe is shown in the Environment section of this document. Run:

```
stsadm.exe -filename mycabfile.cab -globalinstall -force
```

The argument `-globalinstall` tells stsadm to insert your WebPart into the GAC and added to the WebPart library on all instances of SharePoint on the current server. The argument `-force` tells stsadm to replace any already installed instances of the WebPart with the one presently in the cab file. If there are no previous instances, this argument has no effect.

That's it!

Be sure to send us an email with your feedback.

EMAIL: SharePoint@formation.com.au

About Formation Technology

Formation Technology is a professional services company providing innovative IT business solutions. We have a team of MCP's are trained and dedicated to deal with complex business challenges. We have built our reputation on a no-nonsense, "on time and on budget" approach that has seen the successful deployment of: solutions that support thousands of concurrent users, portals that improve collaboration and knowledge worker productivity, and expert internet and intranet systems.

To find out more, visit: www.formation.com.au

Formation Technology
EMAIL: sales@formation.com.au