# Website Performance

By John Liu

The key points regarding SharePoint website performance enhancements focus on configuration and customization.

There are several blogs and articles relating to enhancing SharePoint performance, and some research was undertaken to look for ideas and evaluate different approaches.  The options that we surmised fall directly into either SharePoint configuration or customization.

Many blogs and MSDN articles deal with setting up the SharePoint configuration settings.

## Metrics

Two metrics were used to evaluate performance:

* YSlow for Firebug (an add-on for FireFox).
* Fiddler for Microsoft Internet Explorer

## SharePoint Configuration Options

These modifications were easy to implement and are available out of the box.

### Setup page caching for anonymous access

Setting up page caching for the anonymous profile was the easiest thing to do that gave us the biggest speed improvements.  Whilst the settings are very easy to set up, there is a catch – the page output is cached for the first visit. When the first user visits the website with Firefox, ASP.NET generates an output for Firefox. This cached version is sent to all users regardless of their browser type until the cache expires.  This was a really odd problem that may seem harmless at a glance, but when coupled with some of the customizations later this can lead to interesting problems.

Make sure you run the forms lockdown.

### Setup object caching for lists

Setting up a SharePoint object caching is also easy to implement.  The idea is that SharePoint can store query results instead of re-querying the underlying database.  This is particularly useful for Content Query web parts.

1. Setup blob cache – except this didn’t work properly!  Bad Microsoft!    
     
   Blob cache is a setting to let SharePoint cache BLOB objects from the SQL Server on the hard disk for easier access.  This is a great feature for caching pictures, JavaScript, css and other static contents that wouldn’t need to be re-fetched from the database.    
   In our testing however, we discover that BlobCache would sometimes mysteriously crop CSS or JS files, leaving them unusable and causing client-side errors.  
   We deployed through features and we read at least another blog report of BlobCache corrupting the files.  This was a shame because we think this would have giving another boost in performance (relatively cheap in terms of development time).    
     
   Reference  
   <http://blogs.msdn.com/maximeb/archive/2008/01/13/real-life-team-based-development-for-web-content-management-with-sharepoint-2007.aspx>

## SharePoint Customizations Options

The biggest performance gain aside from using page caching was to force SharePoint to start behaving the way we wanted it to.

We didn’t want to have a lot of JavaScript or CSS files that SharePoint sent to anonymous users.

1. Design a public website with performance in mind – improving performance in retrospect can be harder

When we decided to optimize the output for anonymous users, this created a new problem – essentially, there are now two categories of user that will see the website. Each feature that we modified in the site needed to be tested for both types of access.

1. SharePoint adds many additional scripts that can’t be easily removed

init.js, core.js, core.css, many hidden fields

core.js buries many functionality that SharePoint needs – if you truly want to remove core.js, you really need to make sure you test all the various links properly.

1. Understanding ScriptLink

SharePoint also relies on a ScriptLink control to exist on the master page (it will add this if it doesn’t exist), and ScriptLink control renders all registered script references in this one place.  ScriptLink has a minimum mode – to create a ScriptLink in minimal mode – create it without the name attribute.

1. Understanding CssLink

SharePoint relies on a control “CssLink” to exist on the master page somewhere (it will add this if it doesn’t exist), and the CssLink control renders all the registered Css references in one place.  One of the catch of the CssLink is that it has a

1. Creating a wrapper container to hide controls and references for non-authenticated users.

We looked at the RegisterCoreNoLoad published in MSDN, but due to the way that CssLink and ScriptLink works – it’s pretty difficult to control this.

One of the easier ways we’ve used to tackle some of these scripts and CssLinks was to use a customized control for selectively rendering child controls based on whether the user is authenticated.  We’ve called this the AuthenticatedPanel control.

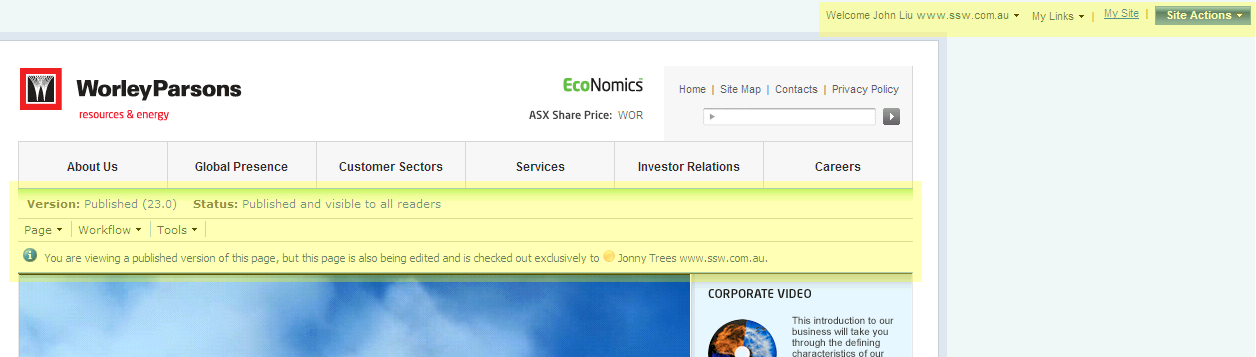
-insert code

1. Using a HttpModule to filter outgoing HTML for ultimate control
   1. This is the best way to really customize your outbound HTML and ensure w3c compliance
   2. Note that different browsers cause different HTML to be rendered in ASP.NET, and the caching controls actually caches output based on which browser visits a page first – make sure you test your HttpModule thoroughly

Even after the various work done on CssLink and ScriptLink, you will find that SharePoint will still generate a lot of additional HTML that’s just really difficult to get rid of.  To make this final modification, we’ve tackled it at the HttpModule level – this essentially is our last line of defence and  this is something that requires a lot of testing

1. Killing the login controls on a public site

A public user doesn’t necessarily need the login control to appear for them, but an authenticated user will still need to see both the Welcome control as well as the DesignModeConsole.  These can be easily handled by the AuthenticatedPanel.



## Conclusion

There are several approaches that can be taken to improve the performance of public SharePoint sites. Some are simple and others more complex, but the results of a combined approach can result in an improved user experience.