[**http://dotnetmentors.com/aspnet/asp-net-website-performance-improvement-tips.aspx**](http://dotnetmentors.com/aspnet/asp-net-website-performance-improvement-tips.aspx)

**ASP.NET Website Performance Improvement Tips**

This article give you some of the best tips to improve your ASP.NET Website performance.

Writing an ASP.NET application is relatively easy with the use of Rapid Application Development controls however it is just not enough to use. You should use each component wisely to get performance benefit to your ASP.NET Website.   
  
Remember that 80-90% of page load time spent on downloading all components like HTML, CSS, Script files, images, flash. Rather than looking for re-designing application architecture you should first try to organize your static contents. Below are some easy tips which helps you to achieve a bigger reduction in response times.

1. **HTTP compression**

Make sure HTTP compression is turned **On**. HTTP compression is a algorithm which removes extra space, redundancy from file and produce clean output. It generates smaller size and reduce weight of file than original representation. The outcome of this is reduced bandwidth and time to load page.   
Although the support for compression is a feature of all modern browsers you should enable and configure or implement custom compression feature.   
You can configure **httpCompression** tag under **system.webServer** in ASP.NET Web.config file or Enable HttpCompression feature in IIS.   
You can also implement **HttpModule** to remove space and return compressed output. However it includes runtime cost to implement custom logic to remove unwanted space and redundancy. You can compress your physical files using GZipStream before deploy to production server. Check for more [IIS settings to improve ASP.NET Website Performance](http://dotnetmentors.com/aspnet/iis-settings-to-improve-web-site-performance.aspx).   
Below is sample code to compress your aspx file

private void CompressFile(string srcPath, string tarPath)

{

using(FileStream srcStream = File.OpenRead(srcPath))

{

using(FileStream tarStream = File.OpenWrite(tarPath));

{

using(GZipStream cStream = new GZipStream(srcStream,

CompressMode.Compress))

{

byte[]data = new byte[srcStream.Length];

srcStream.Read(data, 0, data.Length);

tarStream.Write(data, 0, data.Length);

}

}

}

}

1. **Set CacheControlMaxAge**

Set CacheControlMaxAge property in Web.config file. It caches static contents like image, css, js files for particular duration. One year is recommended settings.   
Below Web.config setting shows you how to configure caching for static contents. By configuring this browser will cache static contents till June 30, 2015.

<system.webServer>

<staticcontent>

<clientCache cacheControlMode="UseExpires"

httpExpires="Tue, 30 June 2015 03:14:07 GMT"/>

</staticcontent>

</system.webServer>

1. **Configure MiniProfiler**

Get [MiniProfiler](https://github.com/MiniProfiler/dotnet) from GitHub or use Nuget package manager and configure for your site. It is profiling library developed by StackExchange team which helps to monitor your Website performance. You can see detail execution time. Do not give an access to end user and once your testing is done remove configuration.

1. **Data Controls Paging**

For your Data controls like GridView, DataGrid, Repeater UI controls, make sure you are using custom or database level paging. The built in feature of control's paging is very heavy and expensive as it has to get all the records (no matter how big is your resultset) from database and then filter it at client side as per current page number and pagesize.   
  
When you implement custom paging you provide required page index and page size to database and database returns filtered rows as per input values. So it reduces the required bandwidth and loading time for GridView. For more details see [how to implement custom paging in ASP.NET](http://dotnetmentors.com/aspnet/repeater-control-and-database-level-paging.aspx).

1. **Client side input validations**

For rich user experience it is necessary to validate user input on client side using JavaScript or jQuery. You can also use RegEx expression to make sure input data is in valid format. This provides quick feedback and makes page more responsive. Also you should provide easy to understand error messages to users.   
Although client side validations are quick and responsive, do not avoid server side validations as client side validations easy to break.

1. **Reduce data sent across the network**

Reducing amount of data sent across the network can improve your application performance significantly. You can achieve this by below things   
1. **Combine script and css files:**Try to combine all script files in single file, it reduce required number of requests to load page. If you are using any third party script file, you can copy its content to your single script file. It will save DNS look up time.   
If you can not combine all scripts in one file, try to make it for one file for one domain script (e.g. make a one script file for all scripts referring from jQuery website). It reduces DNS lookup time.   
Apply same tips for css files.   
  
2. **Minify script and css files:**You should minify your script and css files to reduce its size. Minification will remove spaces from script or css files and replace long keyword with small characters. You can use this tool for [minification](http://www.jsmini.com/).   
  
3. **Script reference:**Use script references at the bottom of the page because async downloads halt when script reference is reached. Stylesheet and images can be download in async.   
  
4. **Async Scripts:**Below code allows you to download script files asynchronously.

<script async src="slow.js"></script>

If clients browsers are old. This can be use for script and css files.

<script type="text/javascript">

var link = document.createElement("link");

link.type = "text/css"; link.rel = "stylesheet";

link.href = "Styles/Site.min.css";

document.getElementsByTagName("head")[0].appendChild(link)</script>

1. **Content Delivery Network**

Use Content Delivery Network(CDN) for hosting images and script files. These files can be cached and reduce load on server.   
A content delivery network (CDN) is a bunch of web servers shared across multiple locations to deliver content more efficiently to users. Depending on user's location and measure of network proximity server is selected for delivering content. It look for server with the fewest network hops or the server with the quickest response time.   
The users location in respect to location has an impact on page load time. Deploying your content to geographically dispersed server or near to your most intended website audience. Watch this video to [learn how to use CDN for asp.net application](https://www.youtube.com/watch?v=MQt7w2jUmbw)

1. **Remove unused client scripts**

Visual Studio by default adds many client scripts to your ASP.NET application. All of them may not be useful for your application. So before deploying to production you must remove unused or duplicate script files.

1. **Optimization of Images**

Below are some tips to optimize images   
1. **Reduce size by optimizing Images:**You can download Image Optimizer from Visual Studio gallery which is easy to use. Download [Image Optimizer](http://visualstudiogallery.msdn.microsoft.com/a56eddd3-d79b-48ac-8c8f-2db06ade77c3/) from asp.net website. It helps you to reduce your image size by 20-25%.   
2. **Image height and width:**If you know the image height and width at design time you should set it for <img /> tag. Doing so space get allocated before image download.   
3. **Image Sprites:**Use Image Sprites to retrieve small images in single request. It preferred method for reducing number of image requests. Combine your background images into a single image and use the CSS background-image and background-position properties to display the desired image segment. Download [this nugget package](http://aspnet.codeplex.com/releases/view/65787) which gives you API for automatically generating sprites and inline images.

1. **AJAX Asynchronous**

Use AJAX to retrieve components asynchronously that may not be needed immediately, such as the content of a collapsed panel, content behind a tab, and so on.

1. **Remove HTTP Modules**

Remove HTTP modules which are not being used like Windows Authentication and disabled services like FTP, SMTP if you are not using them. It reduces the load on server to process requests and response.

1. **Caching**

Many developer say caching is the best and only solution to improve performance. However you should first take care of all of the above solutions and then implement caching. This strategy will give you significant boost to your application.   
Caching allows you to store objects on web server after first request and subsequent requests served through cache without much processing. In ASP.NET you can achieve caching in two ways.   
  
1. **Output Cache:**Output cache allows you to store entire aspx page or user control's html to be stored in web server's memory. This cache can be valid for certain duration and can be made invalid depending on different attributes.

<%@ OutputCache Duration="60" VaryByParam="None" VaryByCustom="browser" %>

2. **Cache Object:**Using Cache object you can store any serializeable object in memory. You can use Insert, Remove methods on cache object. These cache objects can be made invalid on some dependecies.

Cache.Insert("uservalues", userFileData, new

System.Web.Caching.CacheDependency(Server.MapPath("~/users.xml")));