# Performance ASP.NET, EPRTR website.

The following points have been done to get the maximize throughput of the EPRTR Asp.Net website.

|  |  |
| --- | --- |
| Debug flag | The debug flag is set to false in the release build, web.config:  <compilation debug="false"> |
| Tracing | Tracing is turns off in the release build, web.config:  <trace enabled="false"/> |
| Unused http Modules | Any unused http module in the web.config files has been removed  <httpModules>..</httpModules> |
| Page buffering | By default this is enabled on all pages, web config:  <pages buffer="true"> |
| Use of IsPostBack property | We use the IsPostBack property to conditionally execute code depending on whether the page is generated in response to a server control event. |
| Process Model | The default values are applied:  maxWorkerThreads: The default value is 20 per process and it determines the maximum number for request that ASP.NET can process in a given second. For application that are not CPU intensive and most of time wait on database request or any external processing this can be increased to get better performance.  memoryLimit: The default is 60%. This is the max memory ASP.NET can use until worker process is refreshed. If you have a dedicated web server with no other services running you can increase this value for better results. |
| Response.Redirect &  Server.Transfer | We only use redirect when the user clicks the new search link buttons from the sheets.  Although Server.Transfer performs better than Response.Redirect we cannot use transfer in this case since the transfer call do not update the url, and the url is used for setting the filter settings in the (new) search. |
| Server side Validation &  Client side validation | Generally, client side validation is better than server side validation from a performance point of view. However the search fields require server-side validation because of the population of content in the different combo boxes. This cannot be moved to the client side. |
| Round trips to the server | We have avoided unnecessary round trips to the server in the design. The functionality of our website only takes selected values entered by the user, and invokes a search in the database. |
| DataBinder.Eval | DataBinder.Eval uses reflection; we only use Container.DataBind to get the best performance possible when binding. |
| ViewState | We only uses ViewState on pages which store (serialize) objects, so we can retain their state after postback.  ASP.NET serializes all objects and controls in the ViewState and transmits them in a hidden field to the browser For this reason a big ViewState will increase page size and therefore increase network traffic.  We have keep the ViewState as small as possible on all pages. |
| Data Paging / Sorting | Only the numbers of rows displayed on each sheet page are received and binded. We do not query or bind rows that are not showed in the lists. This gives us the best performance possible (with paging functionality). |
| Use HTTP Compression | We do not currently use HTTP Compression. Not all browsers support HTTP Compression.  Brief:  *If your page size is large enough to cause noticeable lag between subsequent request and response you can use HTTP compression. HTTP compression is a feature of IIS and what it means is that you can compress data sent to the client using compression techniques like GZIP and Deflate. On the other side the browser decompresses the data and shows the response to the client.* |
| Caching | We have used data cache on the static pages, and we have used data cache for the dropdown lists in the search pages.  Because of the many search combination we cannot cache pages from the search results. |
| Database index | All tables in the database have indexes according to the most typical searches. The gives us the best possible database lookup performance. |
| Getting data, LINQ | In our design we have use the latest technique from Microsoft to get data from the database. This is referred to as LINQ. We have implemented our LINQ layer in such a way, that we have divided the data classes up in several modules in order to keep things separate. This gives us the lowest memory footprint and it is the best practice according to discussions from coding forums on the net. |