

Unit 6 - Caching

1.What is Caching?

- Caching is a technique of storing frequently used data/information in memory, so that, when the same data/information is needed next time, it could be directly retrieved from the memory instead of being generated by the application.
- Caching is extremely important for performance boosting in ASP.Net, as the pages and controls are dynamically generated here. It is especially important for data related transactions, as these are expensive in terms of response time.
- Caching places frequently used data in quickly accessed media like the random access memory of the computer. The ASP.Net runtime includes a key-value map of CLR objects called cache. This lives with the application and is available via the HttpContext and System.Web.UI.Page.
- The data will not be available if its lifetime expires, or the application releases its memory, or caching does not take place for some reason.
- You can access items in the cache using an indexer and may control the lifetime of objects in the cache and set up links between the cached objects and their physical sources

2. Explain Page Output Caching and its properties.

- Page Output Caching caches an entire page. It provides you to cache the entire rendered contents of a page in memory. The next time that any user requests the same page, the page is retrieved from the cache.
- Rendering a page may involve some complex processes like, database access, rendering complex controls etc. Output caching allows bypassing the round trips to server by caching data in memory.
- The **OutputCache** directive is responsible of output caching. It enables output caching and provides certain control over its behaviour.
- Syntax for OutputCache directive:

```
<%@ OutputCache Duration="15" VaryByParam="None" %>
```
- Put this directive under the page directive . This tells the environment to cache the page for 15 seconds.

- The following event handler for page load would help in testing that the page was really cached:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("This page was generated and cache at:" +
        DateTime.Now.ToString());
}
```

Properties :

Attribute	Values	Description
DiskCacheable	true/false	Specifies that output could be written to a disk based cache.
NoStore	true/false	Specifies that the "no store" cache control header is sent or not.
CacheProfile	String name	Name of a cache profile as to be stored in web.config.
VaryByParam	None * Param- name	Semicolon delimited list of string specifies query string values in a GET request or variable in a POST request.
VaryByHeader	* Header names	Semicolon delimited list of strings specifying headers that might be submitted by a client.
VaryByCustom	Browser Custom string	Tells ASP.Net to vary the output cache by browser name and version or by a custom string.
Location	Any Client Downstream Server None	Any: page may be cached anywhere. Client: cached content remains at browser. Downstream: cached content stored in downstream and server both. Server: cached content saved only on server. None: disables caching.
Duration	Number	Number of seconds the page or control is cached.

3. Explain Partial page caching.

- Partial-Page Output Caching, or page fragment caching, allows specific regions of pages to be cached.
- ASP.NET provides a way to take advantage of this powerful technique, requiring that the part(s) of the page you wish to have cached appear in a User Control.
- One way to specify that the contents of a User Control should be cached is to supply an OutputCache directive at the top of the User Control. That's it! The content inside the User Control will now be cached for the specified period, while the ASP.NET Web page that contains the User Control will continue to serve dynamic content.
- (Note that for this you should not place an OutputCache directive in the ASP.NET Web page that contains the User Control - just inside of the User Control.)

4. Explain Absolute Expiration Caching with syntax.

- Absolute expiration means It will expire cache after some time period set at the time of activating cache.
- This will be absolute expiration whether cache will be used or not It will expire the cache.
- This type of expiration used to cache data which are not frequently changing.
- string cacheData = "The data to be cached";
//Absolute Expiration

```
Cache.Insert("AbsoluteCacheKey", cacheData, null, DateTime.Now.AddMinutes(1),  
System.Web.Caching.Cache.NoSlidingExpiration);
```

5. Explain Sliding Expiration Caching with syntax.

- Sliding expiration means It will expire cache after time period at the time of activating cache if any request is not made during this time period.
- This type of expiration is useful when there are so many data to cache.
- So It will put those items in the cache which are frequently used in the application.
- So it will not going to use unnecessary memory.
- string cacheData = "The data to be cached";
//Sliding Expiration

```
Cache.Insert("SlidingExpiration", cacheData, null,  
System.Web.Caching.Cache.NoAbsoluteExpiration, TimeSpan.FromMinutes(1));
```

6. Explain Data Caching & properties.

- The main aspect of data caching is caching the data source controls.
- Data source controls represent data in a data source, like a database or an XML file.
- These controls derive from the abstract class `DataSourceControl` and have the following inherited properties for implementing caching:
 - **CacheDuration** - sets the number of seconds for which the data source will cache data
 - **CacheExpirationPolicy** - defines the cache behaviour when the data in cache has expired
 - **CacheKeyDependency** - identifies a key for the controls that auto-expires the content of its cache when removed
 - **EnableCaching** - specifies whether or not to cache data