In this video, we will discuss **OutputCache**attribute and partial caching in asp.net mvc. **OutputCacheAttribute** is used to cache the content returned by a controller action method, so that, the same content does not need to be generated each and every time the same controller action is invoked. Let us understand this with an example.   
  
   
  
We will be using table **tblEmployee**for this demo. Please refer to [Part 62](http://csharp-video-tutorials.blogspot.com/2013/07/part-62-implementing-search.html) of [MVC Tutorial](http://www.youtube.com/user/kudvenkat/videos?view=1&flow=grid), if you need SQL script to create and populate this table.  
 **Step 1:** Add ado.net entity data model based on table **tblEmployee**. We have discussed doing this several times, in previous sessions of this video series.  
  
   
  
**Step 2:** Add HomeController with the following settings.  
**a)** Controller name = HomeController  
**b)** Template = MVC controller with read/write actions and views, using Entity Framework  
**c)** Model class = Employee  
**d)** Data context class = SampleDBContext  
**e)** Views = Razor  
  
**Step 3:** Modify the **Index()** action method in **HomeController** as shown below. Notice that, we are using **OutPutCache** attribute to cache the content returned by **Index()**action method for 10 seconds.  
[OutputCache(Duration = 10)]  
public ActionResult Index()  
{  
    System.Threading.Thread.Sleep(3000);  
    return View(db.Employees.ToList());  
}  
  
**Step 4:** Modify code in **"Index.cshtml"** view as shown below. The modified code is highlighted in Yellow.  
@model IEnumerable<MVCDemo.Models.Employee>  
<div style="font-family:Arial">  
<p>  
    @Html.ActionLink("Create New", "Create")  
</p>  
<table border="1">  
<tr>  
    <td colspan="4">  
         <b>Employee List retrieved @@ @DateTime.Now.ToString()</b>  
    </td>  
</tr>  
    <tr>  
        <th>  
            @Html.DisplayNameFor(model => model.Name)  
        </th>  
        <th>  
            @Html.DisplayNameFor(model => model.Gender)  
        </th>  
        <th>  
            @Html.DisplayNameFor(model => model.Email)  
        </th>  
        <th></th>  
    </tr>  
  
@foreach (var item in Model) {  
    <tr>  
        <td>  
            @Html.DisplayFor(modelItem => item.Name)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.Gender)  
        </td>  
        <td>  
            @Html.DisplayFor(modelItem => item.Email)  
        </td>  
        <td>  
            @Html.ActionLink("Edit", "Edit", new { id=item.ID }) |  
            @Html.ActionLink("Details", "Details", new { id=item.ID }) |

            @Html.ActionLink("Delete", "Delete", new { id=item.ID })  
        </td>  
    </tr>  
}  
</table>  
</div>  
  
When you navigate to **/Home/Index**, the view ouput is cached for 10 seconds. If you refresh the view, within 10 seconds you will get a cached response. After 10 seconds, the cache expires, code is executed again and the output is cached for another 10 seconds.  
  
**Caching specific portion of a view using ChildActionOnly attribute:**  
**Step 1:** Remove **OutputCache** attribute and the line which calls **Thread.Sleep()**, from the **Index()** action method in **HomeController**. After the changes, the Index() action method should be as shown below.  
public ActionResult Index()  
{  
    return View(db.Employees.ToList());  
}  
  
**Step 2:** Add **GetEmployeeCount**() action method to **HomeController**. Notice that, this method is decorated with **OutputCache** and **ChildActionOnly** attributes.   
// Child actions can be used to implement partial caching,   
// although not necessary. In this case, even if  the ChildActionOnly  
// attribue is removed, a portion of the view will be cached as expected  
[ChildActionOnly]  
[OutputCache(Duration = 10)]  
public string GetEmployeeCount()  
{  
    return "Employee Count = " + db.Employees.Count().ToString() + "@ " + DateTime.Now.ToString();  
}  
  
**Step 3:** Copy and paste the following code, just below the closing table tag in Index.cshtml view.  
<br /><br />  
<b> @Html.Action("GetEmployeeCount") </b>  
  
Navigate to **/Home/Index**. Notice that, everytime you refresh the page, the time in the section of the page that displays employee list changes, but not the time, that displays the employee count. This proves that, only a portion of the view, is cached.