Caching در ASP.NET

۱- مقدمه ای بر Caching در ASP.NET

- Caching عبارت است از قرار دادن حاصل اجرای یک صفحه در یک حافظه سریع جهت دسترسی سریع و استفاده مجدد از آن در ارجاعات بعدی.
- Caching مهمترین فاکتور در ایجاد یک برنامه تحت وب با توانائی و کارائی بالا می باشد.
- محل های Caching عبارتند از Web server, proxy server و client browsers.
- انواع Caching عبارتند:

Output caching

Fragment caching

Data caching

۲- Output Caching

Output Caching چیست؟

- صفحاتی که از Output Caching استفاده می کنند برای بار اول اجرا می شوند و سپس حاصل آن cache می گردد. جهت پاسخگوئی به درخواستهای بعدی برای همین صفحه، از نسخه cache شده استفاده می گردد.
- فواید Output Caching در کاهش محسوس زمان پاسخ دهی سرور و کاهش بار اضافی بر روی CPU در سرور می باشد.
- استفاده صحیح از Output Caching سرعت و کارائی سایت را بطور محسوسی افزایش می دهد.
- از Output Caching می توان در فایل های با پسوند .aspx،asmx. و ascx. استفاده نمود.
- با استفاده از دایرکتیو OutputCache @ در بالای فایلهای فوق بصورت زیر می توان Output Caching را فعال کرد:

```
<%@ OutputCache Duration="600" Location="Any" VaryByParm="none" %>
```

- یا از درون برنامه بصورت زیر هم می توان این کار را انجام داد:

```
Response.Cache.SetExpires(DateTime.Now.AddSeconds(600));

Response.Cache.SetCacheability(HttpCacheability.Public);
```

صفات OutputCaching

- Duration: مدت زمان معتبر بودن cache را مشخص می کند. مقداردهی این صفت الزامی است و مقدار آن بر اساس ثانیه است.

- Location: محل قرار گرفتن cache را مشخص می کند.
  در حالت Server صفحه cache شده در حافظه سرور نگهداری می شود.
  در حالت Downstream صفحه cache شده بر روی proxy server نگهداری می گردد.
  در حالت Client صفحه cache شده بر روی مرورگر کاربر قرار می گیرد.
  در حالت Any صفحه cache شده بر روی هر یک از موارد فوق می تواند قرار گیرد.
  در حالت None صفحه مربوطه عملا در هیچ کجا cache نخواهد شد.

- VaryByParam: نسخه های cache متفاوتی از صفحه مورد نظر براساس پارامترهای موجود در QueryString و Form یا ترکیبی از آنها ایجاد می گردد.

```
<%@ OutputCache Duration="10" VaryByParam="location;count" %>
```

- VaryByHeader: نسخه های cache متفاوتی براساس مقادیر مختلف پارامتر تعیین شده در HTTP header ایجاد می گردد.

```
<%@ OutputCache Duration="60" VaryByHeader="Accept-Language" %>
```

- VaryByCustom: اگر مقدار این صفت کلمه خاص "Browser" باشد، cache مورد نظر براساس نوع و نسخه اصلی مرورگرایجاد خواهد شد. اگر مقدار آن یک رشته دلخواه باشد، آنگاه لازم است که شما متد HttpApplication.GetVaryByCustomString را در فایل Global.asax را بگونه دلخواه بازنویسی کنید.

۳- Fragment Caching

- علاوه بر اینکه شما می توانید تمام یک صفحه را cache کنید، شما حتی می توانید بخشی از یک صفحه را cache کنید. به این عمل Fragment Caching گویند.
- بدین منظور لازم است که شما بخش های مورد نظر را بصورت User Control یا کنترل کاربری ایجاد کرده باشید.
- هر کنترل کاربری دایرکتیو @OutputCache مخصوص به خود را دارا می باشد.
- صفات مورد استفاده در اینجا عبارتند از VaryByParam و VaryByControl.
- امکان تعیین محل cache با استفاده از صفت Location مقدور نمی باشد و محل cache همواره بر روی سرور در نظر گرفته می شود.

- VaryByControl: ششمین صفت موجود در دایرکتیو @Outputcache می باشد. تنها می توان در کنترلهای کاربری از آن استفاده کرد. استفاده از آن سبب می شود تا cacheهای متعددی براساس خواص (properties) کنترل کاربری ایجاد گردد.

جواب این پرسش را تایپ جدید BigInteger داده است. اگر در برنامه فوق نوع متغیر res را به BigInteger تغییر دهیم می توان فاکتوریل اعداد بزرگتری را محاسبه کرد.

البته ذکر این نکته نیز مهم است که استفاده از BigInteger سرعت برنامه را کمی کاهش می دهد. پس باید در استفاده از این تایپ جدید جوانب احتیاط را در نظر گرفته و تنها در صورت نیاز مبرم از BigInteger استفاده کنیم.

```
<%@ Language="C#" %>

<%@ OutputCache Duration="10" VaryByControl="State;Country" VaryByParam="*"%>

<script runat=server>

public String State {

get { return state.Value; }

set { state.Value = State; } }

public String Country {

get { return country.Value; }
```

```
set { country.Value = Country; } }

</script>
```

• می توان بصورت تودرتو از کنترلهای کاربری با قابلیت cache استفاده نمود. در این صورت یک cache سلسله مراتبی بسیار قدرتمند خواهیم داشت. این مساله اگر چه نیاز به هیچ نوع برنامه نویسی خاصی ندارد اما می تواند سبب مصرف حافظه زیادی گردد.
• سعی نکنید بصورت برنامه ای به یک کنترل کاربری موجود در cache دسترسی داشته باشید. در غیر این صورت شما با یک exception برخورد خواهید کرد. زیرا این نوع کنترلها در درخت کنترلهای موجود قرار نمی گیرند.

۴- Data Caching

• با استفاده از data cache می توان داده های برنامه مانند رشته ها، DataSetها و سایر اقلام داده و آبجکت را بصورت زیر cache کرد:

```
Cache ("counter") = mycount.text
```

• اگرچه این مساله مانند استفاده از متغیرهای از نوع Application است، اما بسیار قوی تر و کارآمدتر می باشد.
• هر قسمت از برنامه که از داده های موجود در cache استفاده می کند، باید قادر باشد تا در صورت غیرمعتبر بودن cache، بتواند آن را دوباره بسازد.

```
Public Function GetProductData() As DataSet

If (IsNothing(Cache("ProductData")) Then

Cache("ProductData") = LoadDataSet()

 End If

 Return Cache("ProductData")

End Function
```

• توصیه می شود جهت استفاده از cache همواره از مدل فوق استفاده نمائید.
• جهت داشتن کنترل بیشتر بر cache یا استفاده از امکانات پیشرفته آن از متدهای Cache.Insert و Cache.Add استفاده نمائید.
• با استفاده از متد Cache.Remove می توانید داده مورد نظر را از cache حذف نمائید.