

مرجع آموزشی

**ASP.NET 4**

(جلد 2)



مرجع آموزشی  
**ASP.NET 4**  
(جلد 2)

مهندس محمد اسماعیلی هدی

انتشارات پندار پارس

سرشناسه	:	اسماعیلی هدی، محمد، 1356 -
عنوان و نام پدیدآور	:	مرجع آموزشی ASP.NET 4.0 / تالیف و ترجمه محمد اسماعیلی هدی.
مشخصات نشر	:	تهران: پندار پارس، 1390.
مشخصات ظاهری	:	ج2: 632 ص. منصور، جدول، نمودار.
شابک	:	ریال: 165000-1-62-2989-964-978 دوره: 8-63-2989-964-978
وضعیت فهرست نویسی	:	فیپا
موضوع	:	صفحه‌های سرور فعال
موضوع	:	ای.اس.پی. (پروتکل شبکه کامپیوتری)
موضوع	:	وب--سایت‌ها--طراحی
رده بندی کنگره	:	TK8885/5105 / ص7الف5 1390
رده بندی دیویی	:	005/276
شماره کتابشناسی ملی	:	8426722

### انتشارات پندارپارس



دفتر فروش: انقلاب، ابتدای کارگر جنوبی، کوی رشتچی، شماره 14، واحد 16 [www.pendarepars.com](http://www.pendarepars.com)  
 تلفن: 66572335 - تلفکس: 666926578 همراه: 09122452348  
[info@pendarepars.com](mailto:info@pendarepars.com)



نام کتاب : مرجع آموزشی **ASP.NET 4.0** (جلد 2)

ناشر : انتشارات پندار پارس ناشر همکار: مانلی

تالیف : محمد اسماعیلی هدی

چاپ اول : پاییز 90

شمارگان : 1000 نسخه

طرح جلد : محمد اسماعیلی هدی

لیتوگرافی : ترام سنج

چاپ، صحافی : صالحان، خیام

قیمت : 16500 تومان شابک : 1-62-2989-964-978 دوره: 8-63-2989-964-978



\* هرگونه کپی برداری، تکثیر و چاپ کاغذی یا الکترونیکی از این کتاب بدون اجازه ناشر تخلف بوده و پیگرد قانونی دارد \*

تقدیم به

## پدر و مادر عزیزم

که وجودشان در زندگی برای من امید و انگیزه است.



## فهرست مطالب:

19	11. کاشه کردن اطلاعات
20	11,1. مفهوم کاشه کردن در ASP.NET
21	11,2. کاشه کردن خروجی
21	11,2,1. کاشه خروجی تعریفی
22	11,2,2. کاشه کردن و رشته کوئری
24	11,2,3. کاشه کردن با پارامترهای رشته کوئری خاص
25	11,2,4. کنترل کاشه سفارشی
26	11,2,5. کاشه کردن با کلاس HttpCachePolicy
27	11,2,6. جانشینی Post-Cache و کاشه کردن قطعه‌ای
28	11,2,7. کاشه کردن قطعه‌ای
28	11,2,8. جانشینی Post-Cache
30	11,2,9. پروفایل‌های کاشه
31	11,2,10. پیکربندی کاشه
32	11,2,11. توسعه‌پذیری کاشه خروجی
33	11,2,12. ساخت یک تهیه کننده کاشه سفارشی
36	11,2,13. استفاده از یک تهیه کننده کاشه سفارشی
37	11,3. کاشه کردن داده‌ها
38	11,3,1. اضافه کردن آیتم‌ها به کاشه
41	11,3,2. یک تست ساده کاشه
42	11,3,3. اولویت‌های کاشه
43	11,3,4. کاشه کردن با کنترل‌های منبع داده
44	11,3,5. کاشه کردن با SqlDataSource
46	11,3,6. کاشه کردن با ObjectDataSource
46	11,4. وابستگی در کاشه
47	11,4,1. وابستگی‌های فایل و آیتم کاشه شده
48	11,4,2. وابستگی‌های تراکمی
49	11,4,3. استفاده از متد Callback برای آیتم حذف شده
52	11,4,4. ایجاد وابستگی کاشه
53	11,5. وابستگی کاشه سفارشی
54	11,5,1. وابستگی کاشه سفارشی به صورت پایه
55	11,5,2. یک وابستگی کاشه سفارشی با صف‌های پیغام

59	..... کار با فایل‌ها	12
59	..... کار با سیستم فایل	12,1
60	..... File و Directory کلاس‌های	12,1,1
64	..... FileInfo و DirectoryInfo کلاس‌های	12,1,2
67	..... DriveInfo کلاس	12,1,3
68	..... کار با خاصیت Attributes	12,1,4
70	..... فیلتر کردن فایل‌ها با کاراکترهای عمومی	12,1,5
71	..... دریافت اطلاعات نسخه فایل	12,1,6
73	..... Path کلاس	12,1,7
75	..... یک مرورگر فایل	12,1,8
80	..... خواندن و نوشتن فایل‌ها با STREAM	12,2
82	..... فایل‌های متنی	12,2,1
84	..... فایل‌های باینری	12,2,2
85	..... Upload کردن فایل‌ها	12,2,3
87	..... امنیت فایل‌ها برای کاربران متعدد	12,2,4
87	..... ایجاد نام فایل‌های یکتا	12,2,5
90	..... قفل کردن اشیاء دسترسی به فایل	12,2,6
91	..... فشردن‌سازی	12,2,7
92	..... سریالی کردن داده‌ها	12,3
97	..... استفاده از LINQ	13
98	..... اصول LINQ	13,1
99	..... نحوه کار LINQ	13,1,1
100	..... عبارتهای LINQ	13,1,2
101	..... مفهوم Projection	13,1,3
103	..... فیلتر کردن و مرتب‌سازی	13,1,4
104	..... گروه‌بندی و تجمع	13,1,5
108	..... عبارتهای LINQ در زیر ذره‌بین	13,1,6
109	..... متدهای الحاقی	13,1,7
110	..... عبارتهای لاندا	13,1,8
111	..... عبارتهای چند قسمتی	13,1,9
112	..... DATASET در LINQ	13,2
115	..... استفاده از DataSet دارای نوع	13,2,1



115	..... مقادیر تهی	13,2,2
116	..... در موجودیت‌ها	LINQ.13,3
117	..... تولید مدل داده	13,3,1
118	..... کلاس‌های مدل داده	13,3,2
118	..... کلاس DerivedObjectContext	13,3,3
119	..... کلاس‌های موجودیت	13,3,4
121	..... ارتباط موجودیت‌ها	13,3,5
121	..... ارتباطات یک به چند	13,3,6
122	..... ارتباطات یک به یک	13,3,7
122	..... پرس‌وجوی روال‌های نخیره شده	13,3,8
124	..... بررسی کوئری‌های LINQ در موجودیت‌ها	13,3,9
125	..... فیلتر کردن قدیمی	13,3,10
126	..... بارگذاری داده‌ها	13,3,11
128	..... استفاده از بارگذاری صریح	13,3,12
130	..... کامپایل کوئری‌ها	13,3,13
131	..... عملیات مختلف روی بانک اطلاعاتی	13,4
131	..... درج کردن داده‌ها	13,4,1
132	..... ایجاد کلاس‌های موجودیت جزئی	13,4,2
132	..... وارد کردن موجودیت‌های وابسته	13,4,3
134	..... عملیات به‌هنگام‌سازی	13,4,4
134	..... عملیات حذف	13,4,5
135	..... مدیریت همزمانی	13,4,6
136	..... رسیدگی به تداخل‌های همزمانی	13,4,7
141	..... کنترل ENTITYDATASOURCE	13,5
142	..... نمایش داده‌ها	13,5,1
146	..... گرفتن داده‌های مرتبط	13,5,2
147	..... ویرایش داده‌ها	13,5,3
148	..... معتبرسازی	13,5,4
150	..... استفاده از کنترل QUERYEXTENDER	13,6
150	..... استفاده از یک SearchExpression	13,6,1
152	..... استفاده از عبارت RangeExpression	13,6,2
152	..... استفاده از PropertyExpression	13,6,3

153	..... استفاده از MethodExpression	13,6,4
155	..... استفاده از XML	14
155	..... لزوم استفاده از XML	14,1
156	..... معرفی XML	14,2
157	..... مزایای XML	14,2,1
158	..... شکل مناسب XML	14,2,2
159	..... فضاهای XML	14,2,3
161	..... الگوهای XML	14,2,4
163	..... پردازش XML مبتنی بر STREAM	14,3
163	..... نوشتن در فایل‌های XML	14,3,1
167	..... خواندن فایل‌های XML	14,3,2
170	..... پردازش XML در حافظه	14,4
171	..... کلاس XmlDocument	14,4,1
174	..... کلاس XPathNavigator	14,4,2
177	..... کلاس XmlDocument	14,4,3
177	..... ایجاد XML با XmlDocument	14,4,4
179	..... خواندن XML با XmlDocument	14,4,5
181	..... فضاها (Namespaces)	14,4,6
183	..... جستجو در محتویات XML	14,5
183	..... جستجو با XmlDocument	14,5,1
185	..... جستجوی XPath با XmlDocument	14,5,2
188	..... جستجوی LINQ با XmlDocument	14,5,3
191	..... معتبرسازی محتویات XML	14,6
191	..... یک الگوی پایه‌ای	14,6,1
192	..... معتبرسازی با XmlDocument	14,6,2
194	..... معتبرسازی با XmlDocument	14,6,3
194	..... تبدیل محتویات XML	14,7
195	..... یک شیوه‌نامه پایه	14,7,1
196	..... استفاده از XslCompiledTransform	14,7,2
197	..... استفاده از کنترل Xml	14,7,3
198	..... تبدیل XML با LINQ در XML	14,7,4
200	..... اتصال به داده XML	14,8

200	..... اتصال غیرسلسله مراتبی	14,8,1
202	..... استفاده از XPath	14,8,2
204	..... استفاده از Grid های تودرتو	14,8,3
206	..... اتصال سلسله مراتبی با TreeView	14,8,4
208	..... استفاده از XSLT	14,8,5
209	..... اتصال به محتویات XML از منابع دیگر	14,8,6
210	..... به‌هنگام‌سازی XML از طریق XmlDataSource	14,8,7
210	..... ADO.NET DATASET و XML	14,9
211	..... تبدیل DataSet به XML	14,9,1
213	..... دسترسی به DataSet به صورت XML	14,9,2
219	..... کنترل‌های کاربری	15
219	..... اصول کنترل‌های کاربری	15,1
220	..... ایجاد یک کنترل کاربری ساده	15,1,1
222	..... تبدیل یک صفحه به یک کنترل کاربری	15,1,2
223	..... اضافه کردن کد به کنترل‌های کاربری	15,2
223	..... رسیدگی به رویدادها	15,2,1
224	..... اضافه کردن خصوصیات	15,2,2
226	..... استفاده از اشیاء سفارشی	15,2,3
228	..... اضافه کردن رویدادها	15,2,4
232	..... دسترسی به کنترل‌های وب داخلی	15,2,5
233	..... بارگذاری دینامیک کنترل‌های کاربری	15,3
234	..... Portal Frameworks	15,3,1
236	..... کاشه کردن قسمتی از صفحه	15,4
237	..... VaryByControl	15,4,1
239	..... به اشتراک گذاشتن کنترل‌های کاشه شده	15,4,2
241	..... THEME ها و صفحات MASTER	16
241	..... استفاده از CSS	16,1
242	..... ایجاد یک شیوه‌نامه	16,1,1
244	..... اعمال قوانین شیوه‌نامه	16,1,2
246	..... استفاده از THEME ها	16,2
247	..... فولدرهای تم و Skin ها	16,2,1
249	..... اعمال یک تم ساده	16,2,2

250	..... رسیدگی به تداخل تمها	16,2,3
251	..... ایجاد چند Skin برای یک کنترل	16,2,4
251	..... استفاده از Skin با الگوها و تصاویر	16,2,5
253	..... استفاده از CSS در یک تم	16,2,6
254	..... اعمال تمها از طریق فایل‌های پیکربندی	16,2,7
255	..... اعمال تمها به طور دینامیک	16,2,8
257	..... استانداردهای چینش سایت	16,3
257	..... اصول صفحات MASTER	16,4
258	..... یک صفحه ساده Master	16,4,1
259	..... یک صفحه ساده محتوا	16,4,2
261	..... محتوای پیش‌فرض	16,4,3
261	..... صفحات Master با جدول‌ها و طرح‌بندی CSS	16,4,4
264	..... صفحه Master و مسیرهای نسبی	16,4,5
265	..... اعمال صفحات Master از طریق فایل‌های پیکربندی	16,4,6
265	..... صفحات پیشرفته MASTER	16,5
265	..... تعامل با کلاس‌های صفحه Master	16,5,1
266	..... تنظیم دینامیک صفحه Master	16,5,2
267	..... صفحات Master تودرتو	16,5,3
271	..... هدایت کاربران سایت	17
271	..... صفحات وب با چند نما	17,1
272	..... کنترل MultiView	17,1,1
276	..... کنترل Wizard	17,1,2
276	..... مراحل ویزارد	17,1,3
279	..... رویدادهای ویزارد	17,1,4
281	..... طرح‌بندی، الگوها و سبک‌های ویزارد	17,1,5
284	..... نقشه سایت	17,2
285	..... تعریف نقشه سایت	17,2,1
287	..... اتصال به نقشه سایت	17,2,2
288	..... کنترل SiteMapPath	17,2,3
290	..... نمایش یک قسمت از نقشه سایت	17,2,4
291	..... حذف گره ریشه	17,2,5
291	..... شروع از گره جاری	17,2,6

292	..... شروع از گره مشخص	17,2,7
295	..... اشیاء نقشه سایت	17,2,8
296	..... افزودن اطلاعات سفارشی به نقشه سایت	17,2,9
297	..... ایجاد یک SiteMapProvider سفارشی	17,2,10
298	..... ذخیره کردن اطلاعات نقشه سایت در بانک اطلاعاتی	17,2,11
298	..... ایجاد تهیه کننده نقشه سایت	17,2,12
303	..... اضافه کردن مرتب‌سازی	17,2,13
304	..... اضافه کردن به کاشه	17,2,14
305	..... برش امنیتی	17,2,15
306	..... نگاشت و مسیریابی URL	17,3
306	..... نگاشت URL	17,3,1
307	..... مسیریابی URL	17,3,2
309	..... TREEVIEW کنترل	17,4
310	..... TreeNode	17,4,1
313	..... قرار دادن گره‌ها در هنگام نیاز	17,4,2
315	..... سبک‌های TreeView	17,4,3
316	..... اعمال سبک به انواع گره‌ها	17,4,4
317	..... اعمال سبک به سطوح مختلف گره‌ها	17,4,5
318	..... تصاویر TreeView	17,4,6
319	..... MENU کنترل	17,5
322	..... سبک‌های Menu	17,5,1
324	..... الگوهای Menu	17,5,2
327	..... نشر سایت وب	18
327	..... نصب و پیکربندی IIS	18,1
328	..... نصب IIS 7	18,1,1
329	..... مدیریت IIS 7	18,1,2
331	..... نشر یک سایت وب	18,2
331	..... انتشار با کپی کردن فایل‌ها	18,2,1
335	..... استفاده از انتشار وب	18,2,2
344	..... استفاده از FTP برای انتشار	18,2,3
351	..... مدل امنیت در ASP.NET	19
351	..... ایجاد نرم‌افزار امن	19,1

351	19,1,1	درک تهدیدهای بالقوه
352	19,1,2	خطوط اصلی کدنویسی امن
353	19,1,3	درک نگهبان‌ها
354	19,2	آشنایی با مراحل امنیت
355	19,2,1	تصدیق کاربران
355	19,2,2	صدور مجوزها
357	19,2,3	قابلیت اعتماد و صحت داده‌ها
357	19,2,4	نحوه کار اجزای مختلف باهمدیگر
359	19,3	لایه SECURE SOCKETS
360	19,3,1	گواهینامه‌ها
361	19,3,2	مفهوم SSL
363	19,3,3	پیگر بندی SSL در IIS 7.x
365	19,3,4	پیگر بندی اتصالات برای SSL
366	19,3,5	رمزگذاری اطلاعات با SSL
369	20	تصدیق فرم‌ها
369	20,1	معرفی تصدیق فرم‌ها
371	20,1,1	دلایل استفاده از تصدیق فرم‌ها
371	20,1,2	دلایل عدم استفاده از تصدیق فرم‌ها
372	20,1,3	دلایل عدم پیاده‌سازی تصدیق کوکی
373	20,1,4	کلاس‌های تصدیق فرم‌ها
374	20,2	پیاده‌سازی تصدیق فرم‌ها
374	20,2,1	پیگر بندی تصدیق فرم‌ها
377	20,2,2	ذخیره کردن اعتبارنامه‌ها در web.config
378	20,2,3	لغو دسترسی افراد غیرمجاز
378	20,2,4	ایجاد یک صفحه Login سفارشی
383	20,2,5	خروج از سیستم
383	20,2,6	Hash کردن رمزها در web.config
385	20,2,7	تصدیق فرم‌ها بدون استفاده از کوکی
385	20,2,8	ذخیره اعتبارنامه‌های سفارشی
387	20,2,9	ماندگاری کوکی‌ها در تصدیق فرم‌ها
389	20,3	IIS 7.x و تصدیق فرم‌ها
395	21	عضویت

395	21,1. معرفی API عضویت .....
398	21,2. استفاده از API عضویت .....
400	21,2,1. پیکربندی تصدیق فرم‌ها .....
401	21,2,2. ایجاد منبع داده .....
405	21,2,3. اسکریپت‌های بانک اطلاعاتی در سرویس‌های ASP.NET .....
408	21,2,4. منبع ذخیره‌سازی SQL Server مبتنی بر فایل .....
409	21,2,5. پیکربندی رشته اتصال و تهیه کننده عضویت .....
413	21,2,6. ایجاد و تصدیق کاربران .....
415	21,3. استفاده از کنترل‌های امنیت .....
417	21,3,1. کنترل Login .....
423	21,3,2. الگوها و کنترل Login .....
425	21,3,3. برنامه‌نویسی کنترل Login .....
428	21,3,4. کنترل LoginStatus .....
429	21,3,5. کنترل LoginView .....
430	21,3,6. کنترل PasswordRecovery .....
433	21,3,7. الگوهای PasswordRecovery .....
435	21,3,8. کنترل ChangePassword .....
436	21,3,9. کنترل CreateUserWizard .....
441	21,4. پیکربندی عضویت در IIS 7.X .....
442	21,4,1. پیکربندی تهیه کننده‌ها و کاربران .....
444	21,4,2. استفاده از API عضویت با سایر برنامه‌ها .....
445	21,5. استفاده از کلاس MEMBERSHIP .....
446	21,5,1. گرفتن کاربران از منبع ذخیره‌سازی .....
448	21,5,2. به‌هنگام‌سازی کاربران در منبع ذخیره‌سازی .....
449	21,5,3. ایجاد و حذف کاربران .....
450	21,5,4. معتبرسازی کاربران .....
453	22. تصدیق ویندوز .....
453	22,1. معرفی تصدیق ویندوز .....
454	22,1,1. دلایل استفاده از تصدیق ویندوز .....
455	22,1,2. مشکلات استفاده از تصدیق ویندوز .....
455	22,1,3. مکانیسم‌های تصدیق ویندوز .....
456	22,1,4. تصدیق پایه .....

457	.....	22,1,5	تصدیق درهم شده
458	.....	22,1,6	تصدیق یکپارچه ویندوز
458	.....	22,1,7	تصدیق NTLM
460	.....	22,1,8	تصدیق Kerberos
463	.....	22,2	پیاپی سازی تصدیق ویندوز
463	.....	22,2,1	پیگیری بندی IIS 7.x
465	.....	22,2,2	پیگیری بندی ASP.NET
465	.....	22,2,3	نگاهی عمیق تر به خط لوله IIS 7.x
469	.....	22,2,4	جلوگیری از دسترسی کاربران ناشناس
471	.....	22,2,5	دسترسی به اطلاعات کاربر ویندوز
471	.....	22,2,6	کلاس WindowsPrincipal
473	.....	22,2,7	کلاس WindowsIdentity
475	.....	22,2,8	IdentityReference و اطلاعات نقش
479	.....	23	صدور مجوز و نقش ها
479	.....	23,1	صدور مجوز URL
480	.....	23,1,1	قوانین صدور مجوز
482	.....	23,1,2	کنترل دسترسی کاربران خاص
484	.....	23,1,3	کنترل دسترسی به دایرکتوری های خاص
485	.....	23,1,4	کنترل دسترسی به فایل های خاص
485	.....	23,1,5	کنترل دسترسی نقش های خاص
487	.....	23,2	صدور مجوز فایل
488	.....	23,3	بررسی صدور مجوز در کدنویسی
488	.....	23,3,1	استفاده از متد IsInRole()
489	.....	23,3,2	استفاده از کلاس PrincipalPermission
490	.....	23,3,3	ادغام اشیاء PrincipalPermission
491	.....	23,3,4	استفاده از ویژگی PrincipalPermission
492	.....	23,4	استفاده از API نقش ها در صدور مجوز مبتنی بر نقش
499	.....	23,4,1	استفاده از کنترل LoginView با نقش ها
500	.....	23,4,2	دسترسی به نقش ها با کدنویسی
503	.....	23,4,3	استفاده از API نقش ها در تصدیق ویندوز
504	.....	23,5	صدور مجوز و نقش ها در IIS 7.x
507	.....	23,5,1	صدور مجوز با نقش های ASP.NET در IIS 7.0



509	مدیریت نقش‌های ASP.NET با IIS 7.x	23,5,2
513	پرو فایل‌ها	24
513	مفهوم پرو فایل‌ها	24,1
514	کارایی پرو فایل‌ها	24,1,1
515	نحوه ذخیره‌سازی داده‌ها در پرو فایل	24,1,2
516	پرو فایل‌ها در مقابل کامپوننت‌های سفارشی داده	24,1,3
517	استفاده از SQLPROFILEPROVIDER	24,2
518	ایجاد جدول‌های پرو فایل	24,2,1
521	پیکربندی تهیه کننده	24,2,2
522	تعریف خصوصیات پرو فایل	24,2,3
523	استفاده از خصوصیات پرو فایل	24,2,4
525	سریال کردن پرو فایل	24,2,5
528	گروه‌های پرو فایل	24,2,6
528	پرو فایل‌ها و انواع داده‌های سفارشی	24,2,7
529	ذخیره کردن انواع سفارشی	24,2,8
531	ذخیره‌سازی اتوماتیک	24,2,9
533	API پرو فایل‌ها	24,2,10
536	پرو فایل‌های ناشناس	24,2,11
538	انتقال پرو فایل‌های ناشناس	24,2,12
539	تهیه کننده‌های سفارشی پرو فایل	24,3
540	کلاس‌های تهیه کننده‌های سفارشی پرو فایل	24,3,1
542	طراحی FactoredProfileProvider	24,3,2
544	برنامه‌نویسی FactoredProfileProvider	24,3,3
545	مقداردهی اولیه	24,3,4
546	خواندن اطلاعات پرو فایل	24,3,5
548	به‌هنگام‌سازی اطلاعات پرو فایل	24,3,6
549	تست FactoredProfileProvider	24,3,7
553	رمزنگاری	25
553	رمزگذاری داده‌ها: اهمیت رازداری	25,1
554	فضای CRYPTOGRAPHY در NET	25,2
559	کلاس‌های رمزنگاری NET	25,3
560	الگوریتم‌های رمزگذاری متقارن	25,3,1

---

562	رمزگذاری نامتقارن.....	25,3,2
563	کلاس‌های فشرده رمزگذاری.....	25,3,3
564	ICryptoTransform واسط.....	25,3,4
564	CryptoStream کلاس.....	25,3,5
566	رمزگذاری داده‌های حساس.....	25,4
566	رموز مدیریت.....	25,4,1
569	استفاده از الگوریتم‌های متقارن.....	25,4,2
573	استفاده از کلاس SymmetricEncryptionUtility.....	25,4,3
574	استفاده از الگوریتم‌های نامتقارن.....	25,4,4
577	رمزگذاری داده‌های بانک اطلاعاتی.....	25,4,5
582	رمزگذاری QUERY STRING.....	25,5
582	بسته‌بندی رشته کوئری.....	25,5,1
586	ایجاد یک صفحه تست.....	25,5,2
589	تهیه‌کننده‌های سفارشی عضویت.....	26
589	معماری تهیه‌کننده‌های سفارشی.....	26,1
591	گام‌های اصولی برای ایجاد تهیه‌کننده‌های سفارشی.....	26,2
591	طراحی کلی یک تهیه‌کننده سفارشی.....	26,2,1
593	طراحی و پیاده‌سازی منبع ذخیره سفارشی.....	26,2,2
600	پیاده‌سازی کلاس‌های تهیه‌کننده.....	26,2,3
604	ایجاد کاربران و اضافه کردن آنها به منبع ذخیره‌سازی.....	26,2,4
609	معتبرسازی کاربران در هنگام ورود.....	26,2,5
611	استفاده از درهم‌سازی Salt برای رمز.....	26,2,6
613	سایر توابع تهیه‌کننده.....	26,2,7
616	پیاده‌سازی XmlRoleProvider.....	26,2,8
622	استفاده از کلاس‌های تهیه‌کننده سفارشی.....	26,2,9
624	اشکال‌یابی با کمک WAT.....	26,2,10
625	استفاده از تهیه‌کننده سفارشی با IIS 7.x.....	26,2,11
627	اصطلاحات.....	

# کاشه کردن<sup>1</sup> اطلاعات

منظور از کاشه کردن، ذخیره یک کپی از اطلاعات در حافظه است. برای مثال، ممکن است نتایج حاصل از یک کوئری را در کاشه نگهداری کنید تا در درخواست‌های بعدی کاربر، نیاز به دسترسی به بانک اطلاعاتی نداشته باشید. همچنین، کاشه باعث دسترسی سریع به داده‌های مورد نظر خواهد شد؛ زیرا در این حالت، داده‌ها از حافظه بازیابی می‌شوند. یک نکته بسیار جالب در مورد کاشه، کارایی<sup>2</sup> بالا و مقیاس‌پذیری<sup>3</sup> زیاد آن به طور همزمان است. منظور از کارایی بالا، دسترسی بسیار سریع به اطلاعات است. همچنین، مقیاس‌پذیری در کاشه ارتقاء پیدا کرده است؛ زیرا شما نیاز به عملیات بانک اطلاعاتی زیادی ندارید.

البته، ذخیره کردن اطلاعات در حافظه، همیشه مفید نیست؛ زیرا حافظه سرور محدود است و پر کردن فضای حافظه سرور، باعث کاهش شدید کارایی آن خواهد شد<sup>4</sup>. به همین دلیل، فقط اطلاعاتی در کاشه قرار می‌گیرد که در آینده، به کرات به آنها نیاز پیدا خواهید کرد. همچنین، هنگامی که حافظه پر می‌شود، اطلاعات کم کاربردتر به طور انتخابی از حافظه کاشه حذف می‌شود. این کار باعث می‌شود تا کارایی برنامه‌های وب حفظ شود.

در ASP.NET می‌توانید یک شیء دلخواه، یک صفحه HTML و یا حتی قسمتی از آن را در حافظه کاشه ذخیره کنید. همچنین می‌توانید زمان انقضای داده‌های خود را مشخص کرده و یا رابطه بین آنها را تعیین نمایید. با تعیین این رابطه می‌توانید هنگام ویرایش داده‌ها، اطلاعات موجود در کاشه را به طور اتوماتیک حذف نمایید.

---

<sup>1</sup> Caching

<sup>2</sup> Performance

<sup>3</sup> Scalability

<sup>4</sup> در صورت پر شدن حافظه RAM، اطلاعات به دیسک سخت منتقل می‌شوند.

## 11,1 . مفهوم کاشه کردن در ASP.NET

برخی از توسعه‌دهندگان صفحات وب، ممکن است کاشه کردن را امری زاید تلقی کنند؛ اما در واقع چنین نیست. با کاشه کردن اطلاعات ضروری در یک زمان کوتاه، حتی ممکن است کارایی سیستم تا ده برابر افزایش یابد!

در واقع، ASP.NET از دو نوع کاشه کردن پشتیبانی می‌کند. برنامه‌های وب، باید از هر دو آنها استفاده نمایند؛ زیرا، هر کدام از آنها مکمل دیگری هستند:

- **کاشه کردن خروجی<sup>1</sup>:** این نوع کاشه کردن، ساده‌ترین نوع آنها است. در این حالت، کپی صفحه HTML تولید شده، در کاشه ذخیره شده و هنگام درخواست‌های بعدی، به کاربران دیگر ارسال خواهد شد. بنابراین، نیازی به تولید مجدد صفحه نخواهد بود. این کار، باعث صرفه‌جویی در زمان و افزایش کارایی سرور خواهد شد.
- **کاشه کردن داده‌ها:** این نوع کاشه، به صورت کدنویسی دستی انجام می‌شود. در این حالت، فقط داده‌هایی کاشه می‌شوند که تولید مجدد آنها نیاز به زمان داشته باشد؛ مانند DataSet بازیابی شده از یک بانک اطلاعاتی. سایر صفحات وب می‌توانند با بررسی محتویات کاشه، از اطلاعات آن استفاده کنند. آیتم‌های موجود در کاشه می‌توانند به طور اتوماتیک منقضی شوند.

دو نوع خاص از کاشه، از مدل‌های فوق ناشی می‌شوند که عبارتند از:

- **کاشه کردن قطعه‌ای:** این نوع کاشه، نوع خاصی از کاشه کردن خروجی است. در این حالت، فقط قسمتی از صفحه وب در کاشه ذخیره می‌شود. در این صورت، کنترل‌های کاربر در کاشه ذخیره شده و در درخواست‌های بعدی مورد استفاده قرار می‌گیرند. به عبارت دیگر، تمام کدها و رویدادهای صفحه، به جز کنترل‌های کاربری خاص، اجرا خواهند شد.
- **کاشه کردن منبع داده:** این نوع کاشه کردن، در کنترل‌های منبع داده‌ای مانند SqlDataSource، ObjectDataSource و XmlDataSource ساخته می‌شوند. این نوع از کاشه کردن، از مدل کاشه کردن داده‌ها استفاده می‌کند. تنها تفاوت آن است که در این حالت، نیازی به راه‌اندازی صریح فرآیند ندارید. فقط باید خصوصیات مورد نظر را تنظیم نمایید.

در این فصل، با گزینه‌های مختلف کاشه و انواع آن آشنا خواهید شد.

<sup>1</sup> Output Caching

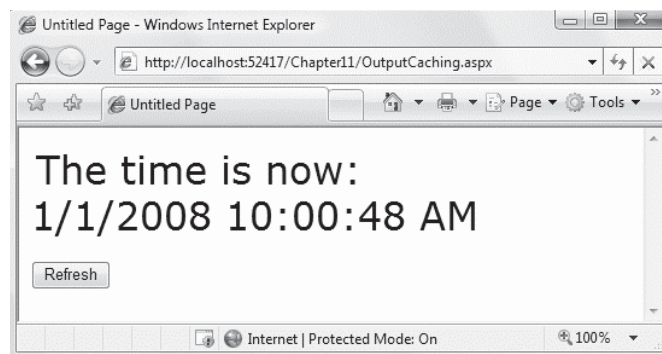
## 11,2. کاشه کردن خروجی

با کاشه کردن خروجی، صفحه HTML تولید شده در حافظه کاشه ذخیره می‌شود. اگر همان صفحه دوباره درخواست شود، کنترل‌ها ایجاد نشده و کدها اجرا نخواهند شد و در عوض، صفحه موجود در کاشه، مجدداً ارسال خواهد شد. به این ترتیب، کارایی برنامه وب افزایش می‌یابد.

■ نکته: یک صفحه ASP.NET ممکن است از منابع استاتیک دیگری مانند تصاویر استفاده کند که به وسیله ASP.NET رسیدگی نمی‌شود. در این حالت، نگران کاشه کردن آنها نباشید؛ زیرا، IIS این آیتم‌ها را به صورت لازم در کاشه ذخیره خواهد کرد.

### 11,2,1. کاشه خروجی تعریفی

برای مشاهده عملی کاشه کردن خروجی، می‌توانید یک صفحه ساده مانند زیر ایجاد کنید که زمان جاری را نمایش می‌دهد.



کد لازم برای ایجاد این صفحه، بسیار ساده است. فقط باید زمان جاری را در یک Label قرار دهید. این کد، به صورت زیر، در داخل رویداد Page.Load قرار می‌گیرد:

```
protected void Page_Load(Object sender, EventArgs e)
{
    lblDate.Text = "The time is now:<br />";
    lblDate.Text += DateTime.Now.ToString();
}
```

شما دو روش برای اضافه کردن این صفحه به حافظه کاشه در پیش دارید. روش متداول، اضافه کردن تگ زیر در بالای فایل `aspx` است. این کد باید در پایین راهنمای<sup>1</sup> Page قرار گیرد:

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

در این مثال، خاصیت `Duration` به `ASP.NET` می‌فهماند که باید صفحه را به اندازه 20 ثانیه در کاشه نگهداری کند. خاصیت `VaryByParam` در بخش بعدی توضیح داده خواهد شد.

هنگامی که صفحه را برای اولین بار اجرا می‌کنید، زمان جاری نشان داده خواهد شد. اگر در فاصله کوتاهی دکمه `Refresh` را فشار دهید، صفحه به‌هنگام‌سازی نخواهد شد؛ زیرا `ASP.NET`، مجدداً صفحه موجود در کاشه را نمایش خواهد داد. بعد از گذشت 20 ثانیه، کپی موجود در کاشه از بین رفته و در هنگام درخواست مجدد، زمان جاری درست نمایش داده خواهد شد.

این زمان کوتاه، در سایت‌هایی که دارای درخواست‌های زیادی است، بسیار مهم خواهد بود. برای مثال، با این کار می‌توانید دسترسی به بانک اطلاعاتی را به سه بار در دقیقه محدود نمایید. به این ترتیب، به ازای هر درخواست، نیاز به اتصال به بانک اطلاعاتی نخواهد بود.

در صورت کمبود حافظه، ممکن است قبل از زمان 20 ثانیه، صفحه وب از کاشه حذف شود. به این ترتیب، به راحتی می‌توانید از کاشه استفاده نمایید؛ بدون آن که نگران کمبود حافظه باشید.

---

■ **نکته:** وقتی دوباره یک صفحه موجود در کاشه را کامپایل می‌کنید، `ASP.NET` آن صفحه را از کاشه حذف خواهد کرد. این کار باعث می‌شود صفحه قدیمی به کاربر نشان داده نشود. در هنگام تست برنامه، بهتر است کاشه کردن را غیرفعال نمایید. در غیر این صورت، هنگام دیباگ کردن صفحه، دچار مشکل خواهید شد؛ زیرا، با وجود کپی آن در کاشه، کد شما اجرا نخواهد شد.

---

## 11,2,2. کاشه کردن و رشته کوئری<sup>2</sup>

یکی از نکات مهم در کاشه کردن، انتخاب یک صفحه برای انتقال آن به حافظه کاشه است. برای نمونه، هنگامی که در ساخت یک صفحه وب از اطلاعات `Session` استفاده می‌شود، نمی‌توانید تمام آن صفحه را در کاشه ذخیره

---

<sup>1</sup> Directive

<sup>2</sup> Query String

کنید. با این حال، می‌توانید قسمتی از آن را کاشه نمایید. مثال دیگر، زمانی است که در ساخت یک صفحه، از اطلاعات صفحه دیگری استفاده می‌شود. معمولاً، این اطلاعات به وسیله رشته کوئری دریافت می‌شود.

در مثال اخیر، خاصیت `VaryByParam` را به `None` تنظیم کنید تا `ASP.NET`، فقط یک کپی از صفحه را در کاشه ذخیره نماید. اگر درخواست صفحه، دارای رشته کوئری در `URL` باشد، `ASP.NET` دوباره همان خروجی موجود در کاشه را ارسال خواهد کرد. توجه کنید که پارامترهای مورد نظر شما می‌تواند در نوار آدرس اضافه شود. همان طور که می‌دانید، پارامترها بعد از علامت `?` قرار می‌گیرند. همچنین، هر پارامتر با علامت `&` از پارامتر دیگر جدا می‌شود.

با این مثال، ممکن است تصور کنید که کاشه کردن، برای صفحات دارای رشته کوئری مناسب نیست؛ اما `ASP.NET` دارای گزینه دیگری هم است. با تنظیم خاصیت `VaryByParam` به مقدار `*` می‌توانید به `ASP.NET` بفهمانید که صفحه وب از رشته کوئری استفاده می‌کند. به این ترتیب، چند کپی مختلف از صفحه در کاشه قرار می‌گیرد:

```
<%@ OutputCache Duration="20" VaryByParam="*" %>
```

هنگامی که صفحه وب را درخواست می‌کنید، اطلاعات رشته کوئری بررسی می‌شود. اگر رشته شما با درخواست‌های قبلی مطابقت داشته باشد، کپی کاشه شده آن صفحه به شما ارسال می‌شود. در غیر این صورت، یک کپی جدید از صفحه ایجاد شده و به حافظه کاشه اضافه می‌گردد. برای درک بهتر این موضوع، به مراحل زیر توجه کنید:

1. یک صفحه را بدون رشته کوئری درخواست کنید تا یک کپی از صفحه (A) را دریافت کنید.
2. صفحه را با پارامتر `ProductID=1` درخواست نمایید. یک کپی از صفحه (B) را دریافت خواهید کرد.
3. یک بار دیگر، صفحه را با پارامتر `ProductID=2` درخواست نمایید. به این ترتیب، کپی دیگری از صفحه (C) را دریافت می‌کنید.
4. دوباره صفحه را با پارامتر `ProductID=1` درخواست نمایید. اگر کاشه B منقضی نشده باشد، همان نسخه به کاربر ارسال خواهد شد.
5. در نهایت، صفحه را بدون رشته کوئری درخواست کنید. اگر نسخه A در کاشه موجود باشد، به کاربر ارسال خواهد شد؛ در غیر این صورت، نسخه جدیدی تولید شده و به حافظه کاشه کپی می‌شود.

خودتان می‌توانید این موضوع را تست کنید. برای تست این برنامه، بهتر است زمان انقضای آن را افزایش دهید.

### 11,2,3. کاشه کردن با پارامترهای رشته کوئری خاص

تنظیم "VaryByParam="\*" اجازه می‌دهد تا صفحات دینامیک را به کاشه منتقل کنید. این روش، برای صفحاتی مانند محصول (Product) بسیار مناسب است؛ زیرا، شناسه محصول به صورت رشته کوئری به این صفحه ارسال می‌شود. به این ترتیب، به ازای هر محصول، یک صفحه در حافظه کاشه قرار می‌گیرد. با این حال، باید زمان کاشه را به چند دقیقه یا بیشتر تنظیم نمایید.

البته، این تکنیک دارای مشکلاتی نیز هست. صفحاتی که دارای محدوده پارامترهای بسیار متنوعی هستند، با این روش به خوبی کار نمی‌کنند. برای مثال، صفحاتی که چند عدد را گرفته و نتایج محاسبه را به کاربر نشان می‌دهند، با این روش، قابل پیاده‌سازی نیستند. دلیل این مساله، مشخص است: احتمال استفاده مجدد از این صفحات بسیار پایین می‌باشد. ضمن آن که با اشغال بی‌رویه حافظه ممکن است باعث حذف سایر اطلاعات موجود در کاشه شوند.

در بسیاری از اوقات، تنظیم VaryByParam به مقدار \* موثر نیست؛ بلکه بهتر است متغیر موجود در رشته کوئری را صریحا معرفی نمایید:

```
<%@ OutputCache Duration="20" VaryByParam="ProductID" %>
```

در این حالت، ASP.NET در رشته کوئری به دنبال ProductID خواهد گشت. بنابراین، درخواست‌هایی با پارامترهای مختلف ProductID، به صورت جداگانه کاشه خواهند شد. همچنین، از سایر پارامترهای رشته کوئری چشم‌پوشی خواهد شد. این کار، مخصوصا زمانی مفید است که رشته کوئری دارای پارامترهای بلااستفاده باشد.

همچنین می‌توان از چند پارامتر مختلف استفاده کرد. در این صورت، پارامترها با علامت ";" از هم جدا می‌شوند:

```
<%@ OutputCache Duration="20" VaryByParam="ProductID; CurrencyType" %>
```

در این حالت، ترکیبات متنوع‌تری از پارامترها در حافظه کاشه ذخیره خواهند شد.

■ **نکته:** درحالت کلی، کاشه کردن خروجی زمانی مفید است که بر مبنای داده‌های موجود در سرور (مانند داده‌های بانک اطلاعاتی) باشد. اگر کاشه کردن بر اساس داده‌های کاربر (مانند داده‌های Session یا کوکی) باشد، سرور کارایی خود را از دست خواهد داد. کاشه کردن خروجی، با رویدادهای مربوط به فرم نیز به درستی کار نخواهد کرد؛ زیرا در این حالت، از رویدادها صرف‌نظر شده و صفحات استاتیک به کاربر ارسال می‌شوند.



## 11,2,4. کنترل کاشه سفارشی

استفاده از پارامترهای متنوع رشته کوئری، تنها راه کاشه کردن نسخه‌های گوناگون از یک صفحه نیست. می‌توان یک روال شخصی ایجاد کرد تا تصمیم بگیرد صفحه کاشه شود و یا آن که یک نسخه جاری به کاربر ارسال گردد. این کد، اطلاعات را بررسی کرده و یک رشته مناسب برمی‌گرداند. سپس، ASP.NET از این رشته برای پیاده‌سازی کاشه استفاده می‌کند. اگر کد شما یک رشته را برای چند درخواست مختلف تولید کند، در این صورت، یک صفحه مناسب در کاشه به کاربر ارسال خواهد شد؛ در غیر این صورت، صفحه جدیدی تولید شده و در کاشه ذخیره می‌گردد.

یکی از روش‌هایی که در کاشه کردن سفارشی استفاده می‌شود، نوع مرورگر است. در این صورت، هر مرورگر مانند Firefox یا IE، صفحات بهینه‌سازی شده خود را دریافت خواهند کرد. برای این کار، ابتدا راهنمای OutputCache را در بالای صفحه‌ای که کاشه خواهد شد، اضافه نمایید. در این حالت، ویژگی VaryByCustom، نوع کاشه سفارشی را مشخص خواهد کرد.<sup>1</sup> در مثال زیر، از نام browser استفاده شده است؛ زیرا، صفحه را بر اساس نوع مرورگر کلاینت، کاشه می‌کند:

```
<%@ OutputCache Duration="10" VaryByParam="None" VaryByCustom="browser" %>
```

سپس، روالی ایجاد کنید که یک رشته سفارشی تولید می‌کند. این روال، باید در فایل global.asax قرار گیرد:

```
public override string GetVaryByCustomString(
    HttpContext context, string arg)
{
    // Check for the requested type of caching.
    if (arg == "browser")
    {
        // Determine the current browser.
        string browserName;
        browserName = Context.Request.Browser.Browser;
        browserName += Context.Request.Browser.MajorVersion.ToString();

        // Indicate that this string should be used to vary caching.
        return browserName;
    }
    else
    {
        return base.GetVaryByCustomString(context, arg);
    }
}
```

<sup>1</sup> این نام، اختیاری است.

تابع `GetVaryByCustomString()` مقدار `VaryByCustom` را به صورت یک آرگومان پاس می‌کند. به این ترتیب می‌توانید تابعی ایجاد کنید که انواع کاشه سفارشی را پیاده‌سازی می‌کند. هر نوع کاشه کردن، از یک نام `VaryByCustom` متفاوت (مانند `BrowserVersion`، `Browser` یا `DayOfWeek`) استفاده می‌کند. تابع `GetVaryByCustomString()` شما، نام `VaryByCustom` را بررسی کرده و یک رشته مناسب برمی‌گرداند.

اصولاً، تابع `GetVaryByCustomString()` مبتنی بر مرورگر ساخته شده بود. به این ترتیب، در متد قبلی نیازی به نوشتن کد ندارید؛ بلکه، متد `GetVaryByCustomString()` رشته مورد نیاز را بر اساس نوع مرورگر و نسخه آن تولید خواهد کرد. اگر بخواهید منطق آن را عوض کنید، باید متد `GetVaryByCustomString()` را مانند مثال قبل، بارگذاری<sup>1</sup> نمایید.

راهنمای `OutputCache` دارای خاصیت سومی به نام `VaryByHeader` نیز هست که اجازه می‌دهد تا نسخه‌های مختلفی از یک صفحه را بر اساس مقدار هدر `HTTP` آن ذخیره نمایید. در اینجا می‌توانید از یک یا چند هدر استفاده کنید که با علامت ";" از هم جدا می‌شوند. این تکنیک بیشتر در سایت‌های چند زبانه استفاده می‌شود که در آن، نسخه‌های مختلف صفحه بر اساس زبان مرورگر کاشه می‌شوند:

```
<%@ OutputCache Duration="20" VaryByParam="None"
  VaryByHeader="Accept-Language" %>
```

### 11,2,5. کاشه کردن با کلاس `HttpCachePolicy`

استفاده از راهنمای `OutputCache` دارای اهمیت بالایی است؛ زیرا دستورات کاشه را از سایر کدهای برنامه جدا می‌کند. همچنین اجازه می‌دهد تا خاصیت‌های مختلف را در یک خط تنظیم نمایید.

با این حال، انتخاب دیگری هم دارید. شما می‌توانید کدی بنویسید که از خاصیت درون‌ساخت `Response.Cache` استفاده می‌کند. این خاصیت، یک نمونه از کلاس `System.Web.HttpCachePolicy` را ارائه می‌کند که به برنامه‌نویس اجازه می‌دهد تا کاشه کردن را برای صفحه جاری فعال نماید. به این ترتیب، با کدنویسی می‌توان تصمیم گرفت که یک صفحه باید در کاشه ذخیره شود یا خیر.

در مثال زیر، صفحه مربوط به نمایش تاریخ، دوباره‌نویسی شده است. در اینجا، کاشه کردن در اولین بارگذاری صفحه فعال می‌شود. فعال کردن کاشه، با استفاده از متد `SetCacheability()` انجام می‌شود. به این ترتیب، صفحه وب، در سرور و کلاینت‌هایی که می‌توانند از نسخه کاشه شده استفاده کنند، ذخیره خواهد شد. متد

<sup>1</sup> Override

`SetExpires()`، زمان انقضای صفحه را مشخص می‌کند. در اینجا، تاریخ انقضا به زمان جاری بعلاوه 60 ثانیه تنظیم شده است.

```
protected void Page_Load(Object sender, EventArgs e)
{
    // Cache this page on the server.
    Response.Cache.SetCacheability(HttpCacheability.Public);

    // Use the cached copy of this page for the next 60 seconds.
    Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

    // This additional line ensures that the browser can't
    // invalidate the page when the user clicks the Refresh button
    // (which some rogue browsers attempt to do).
    Response.Cache.SetValidUntilExpires(true);

    lblDate.Text = "The time is now: <br />" + DateTime.Now.ToString();
}
```

از دیدگاه طراحی، استفاده از کدنویسی برای کاشه کردن ملموس نیست. در عین حال، قرار دادن کدهای کاشه در بین کدهای دیگر، باعث شلوغی و آشفتگی در صفحه می‌شود. به خاطر داشته باشید که کدهای موجود در رویداد `Page.Load`، فقط زمانی اجرا می‌شوند که صفحه مورد نظر در کاشه نباشد.

---

■ **نکته:** باید مطمئن شوید که از خاصیت `Response.Cache` صفحه استفاده می‌کنید نه از خاصیت `Page.Cache`. از خاصیت `Page.Cache` برای کاشه کردن خروجی استفاده نمی‌شود؛ بلکه، دسترسی شما به کاشه داده را فراهم می‌کند. این موضوع در قسمت "کاشه کردن داده‌ها" مورد بررسی قرار می‌گیرد.

---

## 11,2,6. جانشینی *Post-Cache* و کاشه کردن قطعه‌ای

در برخی شرایط، ممکن است متوجه شوید که نمی‌توانید تمام قسمت‌های یک صفحه را در کاشه ذخیره نمایید؛ اما دوست دارید حداقل قسمتی از آن را در کاشه نگهداری کنید. برای انجام این کار، دو راه در پیش رو دارید:

**کاشه کردن قطعه‌ای:** در این حالت، فقط محتویاتی را مشخص می‌کنید که قصد دارید آن را کاشه نمایید. برای این کار، آن را در یک کنترل کاربری قرار داده و سپس خروجی آن را در کاشه ذخیره می‌کنید.

**جانشینی *Post-Cache*:** در این حالت، فقط محتویات دینامیکی را مشخص می‌کنید که نمی‌خواهید اطلاعات آنها در کاشه ذخیره شود. محتویات این قسمت‌ها با استفاده از کنترل `Substitution` تعیین می‌گردد.

با این که پیاده‌سازی کاشه کردن قطعه‌ای، ساده‌تر از روش دوم است؛ اما انتخاب یکی از این دو روش، به حجم داده‌ها بستگی دارد. اگر حجم داده‌ها کم باشد، روش کاشه کردن قطعه‌ای بهتر عمل می‌کند. برعکس، اگر حجم داده‌های دینامیک کم باشد، روش جاننشینی Post-Cache راحت‌تر است. به هر حال، هر دو روش کارایی یکسانی دارند.

### 11,2,7. کاشه کردن قطعه‌ای

برای پیاده‌سازی این روش، نیاز به ایجاد یک کنترل کاربری<sup>1</sup> دارید تا قسمت مورد نظر برای کاشه شدن را پوشش دهد. سپس باید راهنمای OutputCache را به کنترل کاربری اضافه نمایید. نتیجه کار، در کاشه قرار گرفتن محدوده مورد نظر است. جزئیات مربوط به کنترل‌های کاربری، در فصل 15 مورد بحث قرار خواهند گرفت.

کاشه کردن قطعه‌ای، بسیار شبیه کاشه کردن صفحه وب است. در اینجا، یک مشکل وجود دارد. اگر صفحه شما، نسخه کاشه شده کنترل کاربری را دریافت کند، نمی‌تواند با آن تعامل داشته باشد. برای مثال، اگر کنترل کاربری دارای یک سری خصوصیت باشد، نمی‌توان آنها را با کدنویسی تغییر داد.

### 11,2,8. جاننشینی Post-Cache

عموما جاننشینی Post-Cache، با یک متد به نام WriteSubstitution() کار می‌کند که به کلاس HttpResponse اضافه شده است. این متد، یک پارامتر ورودی دریافت می‌کند که به یک متد Callback اشاره می‌نماید. این متد Callback در کلاس Page پیاده‌سازی شده است و محتویات قسمت مورد نظر از صفحه را برمی‌گرداند.

در اینجا یک نکته مهم وجود دارد. وقتی صفحه ASP.NET، قسمت کاشه شده را دریافت می‌کند، به طور اتوماتیک، متد Callback را اجرا کرده و محتوای دینامیک را به دست می‌آورد. سپس آن را به صفحه کاشه شده اضافه می‌کند. حتی اگر صفحه شما هنوز کاشه نشده باشد (مانند اولین باری که صفحه تولید می‌شود)، باز هم متد Callback اجرا خواهد شد.

متدی که محتویات دینامیک را تولید می‌کند، باید از نوع استاتیک باشد. این متد، یک شیء HttpContext را می‌گیرد که حاوی درخواست جاری است. سپس یک رشته با HTML جدید برمی‌گرداند. مثال زیر، یک تاریخ به حالت پررنگ (Bold) را برمی‌گرداند:

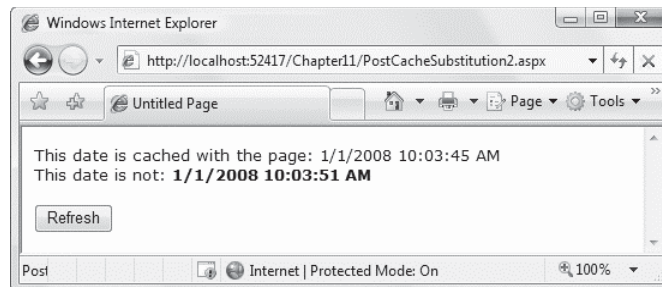
<sup>1</sup> User Control

```
private static string GetDate(HttpContext context)
{
    return "<b>" + DateTime.Now.ToString() + "</b>";
}
```

برای گرفتن این رشته و نمایش آن در صفحه، باید از متد `Response.WriteSubstitution()` استفاده نمایید:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("This date is cached with the page: ");
    Response.Write(DateTime.Now.ToString() + "<br />");
    Response.Write("This date is not: ");
    Response.WriteSubstitution(new HttpResponseSubstitutionCallback(GetDate));
}
```

حال، حتی اگر کاشه کردن را با راهنمای `OutputCache` به این صفحه اعمال کنید، تاریخ دوم به صورت به‌هنگام شده<sup>1</sup>، نشان داده خواهد شد. شکل زیر این موضوع را نشان می‌دهد.



معمولاً وقتی یک صفحه `ASP.NET` را طراحی می‌کنید، اصولاً از شیء `Response` استفاده نمی‌کنید؛ بلکه کنترل‌های وب را به کار می‌برید. این کنترل‌ها، از شیء `Response` برای تولید محتوای خود استفاده می‌کنند. یک مشکل آن است که اگر مانند مثال قبل، از شیء `Response` استفاده کنید، نمی‌توانید محتویات مورد نظر را در جای مناسب از صفحه قرار دهید. تنها راه حل این است که محتویات دینامیک را در یک سری کنترل قرار دهید. به این ترتیب، کنترل می‌تواند هنگام تولید کردن خود، از `Response.WriteSubstitution()` استفاده نماید.

اگر قصد ندارید از کنترل‌های شخصی استفاده کنید، برای به کار بردن جانشینی `Post-Cache`، یک راه میانبر دارید: یک کنترل عمومی `Substitution` که از این تکنیک برای دینامیک کردن محتویات خود استفاده می‌کند. باید کنترل `Substitution` را به متد استاتیکی متصل نمایید که محتویات دینامیک شما را برمی‌گرداند (مانند مثال قبل).

<sup>1</sup> Update