بسم الله الرحمن الرحیم

جمهوری اسلامی ایران
وزارت علوم ، تحقیقات ، فن آوری

دانشگاه پیام نور

طراحی سایت تجارت الکترونیک (بهبازار)

پروژه دوره کارشناسی

در رشته **مهندسی کامپیوتر** گرایش **نرم افزار**

ارائه شده به :

دانشگاه پیام نور مرکز بهشهر

استاد راهنما :

دکتر محمد ابراهیم طالبیان

توسط :

محمد جعفری فوتمی

۱۳۸۶/۱۰/۱۰

پیش گفتار

خوشحالیم از اینکه با وجود تحمل سختی های فراوان در حین تحصیل بالاخره به پایان راه رسیدیم و توانستیم از این مسیر دشوار به سلامت عبور کنیم . امیدواریم این پایان نامه ، پایانی بر تمامی مصائب ما بوده و آغازی باشد برای رسیدن به موفقیت ها و مدارج بالاتر .

با توجه به پیشرفت روز افزون علم کامپیوتر و با توجه به احساس نیاز بشر به سرعت و دقت بیشتر در انجام امور و کارهای روزمره زندگی ، شرکت ها و ادارات خصوصی و دولتی و همه آحاد جامعه به این نتیجه رسیده اند که بدون استفاده از علم کامپیوتر و خدمات الکترونیک قادر به رقابت با سایر رقبای خود نبوده و همچنین با سیستم سنتی نخواهند توانست جوابگوی مشتریان ، ارباب رجوع و نیازهای زندگی خود باشند . بنابراین در عصر حاضر شاهد نفوذ روز افزون علم کامپیوتر و خدمات الکترونیک در این جوامع هستیم .

با توضیحاتی که در بالا به آن ها اشاره شد و با توجه به اینکه در رشته مهندسی کامپیوتر تحصیل می کردیم ، ما نیز تصمیم گرفتیم تا از این حرکت جا نمانده و به قطار سریع السیر پیشرفت و الکترونیکی شدن بپیوندیم . در نتیجه تصمیم به طراحی سایتی گرفتیم که توسط آن خدمات الکترونیک را ارائه کنیم . که این سایت نام بهبازار را به خود گرفت .

در آخر لازم می دانیم تا از زحماتی که دوستان و اساتید گرانقدر برای ما کشیده اند تشکر کنیم و امیدواریم که بتوانیم در راهی که در پیش گرفته ایم با سعی و جدیت به کار خود ادامه دهیم تا بتوانیم زحمات این عزیزان را ارج نهیم .

زمستان ۸۶

محمد شیردل

تقدیم به:

استاد ، راهنما و مشوق همیشگی ما ،

دکتر محمد ابراهیم طالبیان

برای ایشان آرزوی موفقیت و توفیق بیش از پیش را خواستاریم .

و همچنین تقدیم به ،

پدر و مادر عزیزمان

که هیچگاه در راه تحصیل ما را تنها نگذاشتند و همواره مشوق ما بودند .

تقدیر و تشکر :

بدین وسیله از تمامی دوستان و عزیزانی که که ما را در تهیه این پروژه یاری نمودند سپاسگزاریم و برایشان آرزوی توفیق و سلامت را داریم .

همچنین از آقای **مهندس سینا عزیزی** که در بخش طراحی و گرافیک سایت کمک زیادی را به ما نمودند تشکر می کنیم و امیدواریم در زندگی خود موفق باشند . و از اساتید محترم سایت تخصصی **دات نت سورس** (**.NET Source**) نیز به دلیل راهنمایی های وافر و در اختیار گذاشتن تجارت خود ممنونیم .

فهرست مطالب

فهرست شکل ها و جداول

چکیده

امروزه برای ایجاد وب سایت ها و برنامه های کاربردی تحت وب تکنولوژی های زیادی وجود دارد که می توانید یکی از آن ها را انتخاب کنید . ASP.NET تکنولوژی فوق العاده ای است که با استفاده از آن می توانید با کمترین تلاش و زحمت ، وب سایت ها و برنامه های کاربردی تحت وب خود را تولید نمایید . ایجاد برنامه های کاربردی تحت وب هرگز ساده نبوده است ، اما اینک با ASP.NET ساده به نظر می رسد . با این تکنولوژی قدرتمند می توانید پیچیده ترین برنامه های کاربردی تحت وب را ایجاد نمایید .

در این پروژه شما طریقه طراحی وب سایت تجارت الکترونیک بهبازار را به صورتی تقریبا کامل فرا خواهید گرفت . کدنویسی و توضیحات این پروژه برای کسانی است که تا حدی کار با Visual Stadio 2005 ، کدنویسی HTML و CSS و همچنین کار با SQL Server 2000 را می دانند .

در ادامه توضیحاتی در مورد تجارت الکترونیک و پیش نیازهای یک طراحی وب سایت خوب را خواهید خواند . همچنین با تعدادی از کنترل های ASP.NET که در پروژه مورد نیاز است آشنا خواهید شد . با کنترل های اعتبار سنجی در فصل سوم آشنا می شوید . طراحی صفحات سایت بهبازار را در ASP.NET می آموزید ؛ با کنترل های سفارشی کاربر و طریقه طراحی و ساخت کامپوننت ها آشنا خواهید شد . همچنین خواهید دید که چگونه می توانید از طریق سایت خود ایمیل ارسال کنید . کار با بانک اطلاعاتی را می آموزید و ...

به امید اینکه این پروژه راهنمای خوبی برای شما باشد و نیازهایتان را تا حد لزوم بر طرف نماید .

فصل اول

تجارت الکترونیک

- روانشناسی سیستم های کامپیوتری
- اهمیت رنگ در طراحی وب سایت
- چند نکته برای طراحی یک وب سایت خوب
- سایت خود را در موتورهای جستجو ثبت کنید
- چگونه رتبه سایت خود را در موتورهای جستجو بالا ببریم
- راه و رسم رونق تجارت الکترونیک با **Google**

۱-۱- روانشناسی سیستمهای کامپیوتری

آیا تا بحال به این نکته اندیشیده اید، که چه چیزهایی باعث زیبایی یک برنامه کامپیوتری می شود؟

آیا تابحال، به روشهایی برای کاربرپسند کردن یک برنامه و یا یک سایت وب و یا یک نوشته معمولی اندیشیده اید؟ چه روشهایی به ذهن شما خطور کرده است ؟ آیا نیاز به یک گرافیست تنها راه حل مشکل است ؟ آیا اصولا خود کامپیوتر اجزایی از زیباشناختی را دارد؟

با جواب دادن به سوالات مطرح شده ی بالا، می توانیم یک شمای کلی در رابطه با روانشناسی کامپیوتر و سیستمهای اطلاعاتی را کسب نماییم .

تعریف :

چه تعریفی در مورد روانشناسی کامپیوتر و سیستمهای اطلاعاتی می توانیم ارایه نماییم ؟ آیا اصولا مجاز هستیم به رابطه روانشناسی و کامپیوتر بپردازیم؟ آیا می‌توانیم کامپیوتر را یک موجود زنده تصور کنیم ؟

یک تعریف کلی و منحصر به فرد می تواند، خط مشی روانشناسی کامپیوتر و سیستمهای اطلاعاتی را روشن نماید:

به طور کلی، روانشناسی کامپیوتر و سیستمهای اطلاعاتی، علمی است که به بررسی رابطه بین انسان و ماشین می پردازد. در تعریف دیگر، می توانیم آنرا به علمی که رابطه‌ی انسان و ماشین را نزدیک می کند، تشبیه کنیم.

با توجه به تعریف ارایه شده، روانشناسی کامپیوتر و سیستمهای اطلاعاتی را می توان به انواع و دسته های زیر تقسیم نمود :

۱- روانشناسی فضاهای مجازی (Cyber space psychology)
۲- روانشناسی رنگها در کامپیوتر (Color psychology with computer)
۳- روانشناسی کاربر و کامپیوتر (The psychology between computer and user)

مبحثی که در اینجا محوریت بحث ما را تشکیل می دهد، روانشناسی رنگها در کامپیوتر و روانشناسی کاربر و کامپیوتر می باشد.

۱-۲- بررسی اساس روانشناسی کامپیوتر و سیستمهای اطلاعاتی:

به طور کلی همانطور که می دانید، انسان و حیوان هردو در اینکه جهان را احساس می کنند، با یکدیگر مشترک هستند، و حتی برخی از حیوانات دارای سیستم عصبی بس پیچیده‌تری نسبت به انسان می باشند. در این صورت است که می گوییم حیوانات جهان، احساسی کاملتری نسبت به انسان دارند.

از طرف دیگر همانطور که میدانید، ادراکات حسی سرآغاز کسب معرفتهای جدید در انسان می باشد. و همانطور که می دانید، دستگاههای عصبی انسانها با یکدیگر برابر می باشند، پس معرفتهای کسب شده از ادراکات انسانی برای تمام انسانها برابر می باشد. در واقع این انسان است که با کمک نیروی عقل و استنتاج و استدلال خویش برداشتهای مختلفی را نسبت به آن حس دارا می باشد. در مورد تاثیرات برنامه ها و سایتهای کامپیوتری بر روی انسان نیز، روش و مسیر به همان صورت است. به عنوان مثال، سایت yahoo را در نظر بگیرید؛ بسیاری از آن به عنوان یک سایت زیبا نام می‌برند.

علت چیست ؟

پاسخ در یک عبارت است! و آن این است : استفاده‌ی صحیح و بجا از اصول روانشناسی رنگها و روابط متقابل کاربر و کامپیوتر؛ چیزهایی که ما از آنها به عنوان روانشناسی کامپیوتر و سیستمهای اطلاعاتی نام بردیم. پس از آنجا که هر انسانی دارای استدلالها و استنتاجهای خاص خود در زمینه های حسی می‌باشد، پس به همان اندازه نیز، شما با نظرات و آرا مختلف در مورد سایت وب خود و یا برنامه ی کامپیوتری خود مواجه خواهید شد.

پس نیاز مبرم به یک رشته آمارگیریهای دقیق در اینجا احساس می شود. در سال ۱۹۸۶ شرکت آی بی ام، و اپل در یکسری آمارگیریهای مشترک در مورد صفحه کلید مناسب برای کاربران کامپیوتر به نظر واحد صفحه کلید امروزی رسیدند.آمار گیری انجام شده در آن زمینه، تماما بر روی دستان کارمندان دو شرکت بازرگانی در ایلتهای پنسیلوانیا و تکزاس انجام شد و جمعا در حدود ۱۵۰۰ نفر در آن شرکت کردند. در آن آمارگیری فاصله بین انگشتان دست کاربران، و فاصله ی چشمان آنها از دستانشان از میز کارشان و ... اندازه گیری شده بود. ولی، برای راه اندازی یک سایت وب جدید، شما نیازی به اطلاعات در حد و گستردگی آن شرکتها ندارید، زیرا:

۱- شما یک سایت وب شخصی را راه اندازی می کنید
۲- سایت وب شما قرار است که در مورد خاصی تخصص داشته باشد
۳- بازدیدکنندگان آن محدود به منابع عرضه شده بر روی سایت وب شما است
۴- سایت وب شما قرار نیست بخشی از زندگی روزمره اشخاص باشد
۵- و ...

از طرف دیگر اجزای اصولی یک کامپیوتر نیز طراحی و ساخته شده اند. درواقع، همه چیز بستگی به محیط عرضه اطلاعات شما دارد. درواقع کاربران یک محیط عرضه‌ی اطلاعات ، بنا بر ساختار خاص آن محیط دارای استدلال و انتظار یکسانی از نحوهٔ عرضه‌ی اطلاعات دارند.

تعریف یک محیط عرضه‌ی اطلاعاتی :

به طور کلی: محیط پشتیبانی اطلاعات شما را یک محیط عرضه اطلاعاتی می نامند. به عنوان مثال، در بحث اینترنت، محیط جستجوگر وب شما (برای مثال، محیط گرافیکی اینترنت اکسپلورر) را یک محیط عرضه‌ی اطلاعاتی می نامند. اگر شما یک برنامه نویس هستید، محیط سیستم عامل شما، همان محیط عرضه‌ی اطلاعاتی برای شما است.

اساس یک محیط عرضه اطلاعاتی :

اساس یا چارچوب یک محیط عرضه‌ی اطلاعاتی، فقط در زمینه های گرافیکی قابل تعریفند. در زمینه‌ی برنامه نویسی، اساس یک محیط عرضه‌ی اطلاعاتی، چهارچوب رایج سیستم عامل است. به عنوان مثال، در محیط سیستم عامل ویندوز شرکت مایکروسافت، اساس همان پنجره های ویندوز است. یعنی شکل چهارچوب همراه با دکمه های کنترل اندازه ی پنجره . دکمه های کنترل، همان × و – و # می باشند.

حال این به چه معنا است. خیلی ساده است. در واقع این بدان معنا است که از این پس اگر شما برنامه ای در این اساس و چارچوب نوشتید، کاربران آینده‌ی نرم افزار شما، انتظار برخورد با همان دکمه های کنترل را در برنامه ی شما در همان محل متداول دارند. اما در دنیای وب و اینترنت، اساس کاملا متفاوت است.

درواقع این درست است که شما باید پیرو اساس ارتباطها و گرافیکهای استاندارد باشید، ولی قالب کلی را شما به کاربر معرفی می کنید. صفحه اول سایت وب شما حکم معرفی اساس کلی و چارچوب کلی سایت وب شما را دارد.

اگر کمی دقت کنید در می‌یابید که اساس کلیه صفحات **yahoo** به یک شکل است. حداقل آن بحث رنگ زمینه است. به عنوان مثال، شما در کلیه صفحه‌های این سایت، زمینه را به رنگ سفید خواهید یافت.قسمت دیگر، نشان **yahoo** است، که در سر مجموعه ها همگی به یک شکل است.

پس شما در سایت وب خود دنیایی متفاوت را برای کاربر خواهید ساخت، و در این دنیای جدید این ایده و نظر شما است که باید بر استدلالها و استنداجهای کاربر حکومت کند و یا حداقل به آنها جهت بدهد .بحث ما در اینجا راجع به محتوی سایت شما نمی باشد و نخواهد بود. در واقع ما در اینجا راجع به قالب منحصر بفرد برای موضوع سایت و یا برنامه ی شما بحث می کنیم.

۱-۳- متغیرهای روانشناسی کامپیوتر و سیستمهای اطلاعاتی :

برای خلق یک اثر هدایت گر با استفاده از مبحث روانشناسی کامپیوتر و سیستمهای اطلاعاتی ما نیاز به تسلط کافی بر چهار عامل زیر داریم :

۱- رابط منطقی (بر مبنای منطق انسان)
۲- رابط گرافیکی
۳- رابط صوتی و تصویری
۴- هدف مندی

الف) رابط منطقی :

در اینجا محور بحث ما برنامه نوسی نیست، زیرا یک برنامه نویس ملزم به استفاده از منطق ریاضی و در نهایت منطق کامپیوترای در برنامه‌اش است . در اینجا محور بحث ما این است که یک برنامه نویس چه باید بکند که کاربر احساس کند که با موجود هوشمندی در ارتباط است! یک مثال ساده بازی حدس اعداد است .در این بازی، کامپیوتر به تصادف عددی را انتخاب می کند، و سپس کاربر را وادار به حدس و گمان می کند.

ابتدا از کاربر پرسیده می شود، فکر می کنید چه عددی در ذهن من است! فرض کنید عدد مورد نظر ۱۲ باشد. اگر کاربر عدد ۵ را به عنوان مثال وارد کند، بر اساس منطق ریاضی زیر کامپیوتر در جواب به او می گوید، عددی که شما انتخاب کردید کمتر از مقداری بود که من در ذهن دارم !

```
"  Your selected number was less then mine If A<12 Then Print "
```

فکر می کنم، محور اصلی بحث مشخص شد. در اینجا کلمات حول منطق ریاضی هستند که منطق انسانی را می سازند.

ب) رابط گرافیکی:

در دنیای رنگارنگ سیستم عامل‌های امروز، و همچنین گرافیستهای خلاق، هیچکس منکر نقش تصاویر گرافیکی مرتبط با موضوع نمی باشد. هیچکس منکر نقش هدایت کنندهٔ تصویر یک توپ فوتبال در سایت فیفا نمی باشد. محور این مبحث استفاده و یا ایجاد تصاویر مرتبط با موضوع سایت و یا برنامه ی شما است.

ج)رابط صوتی و تصویری :

استفاده‌ی بجا از یک موسیقی آرام بخش و یا هیجان آور در یک سایت و یا برنامه ی کاربردی می تواند کاربر را مشتاقتر و یا متنفر از سایت وب شما سازد. خواندن یک جمله، مثلا خبر یا پیام خوش آمد گویی در ابتدای سایت شما، ایدهٔ بس جالبی است که به مورد اول یعنی منطق کمک شایانی خواهد کرد.

د) هدف :

به هر حال شما از طراحی سایت وب تان هدف خاصی را دنبال می کردید و می کنید. طراحی سایت و یا برنامه ی شما باید بر اساس رسیدن به هدف شما باشد. اگر یک سایت ثبت نام برای مسابقه‌ی فوتبال طراحی می کنید، باید تمام ارکان قابل مشاهده در این سایت ، هدف را که همانا ثبت نام برای مسابقه است را نشان دهد.

روانشناسی کامپیوتر و سیستمهای اطلاعاتی، در سطح گسترده تری با نام ، روانشناسی انسان و ماشین مورد بحث قرار می گیرید، و همه سازندگان و خلاقان به صورت آگاهانه و یا نا آگاهانه از آن بهره می برند.به عنوان مثال، در صنعت خودروسازی امروزی، به بررسی این رابطه انسان (راننده) و اجزای اتومبیل می پردازند و سعی بر آن است که احساس رانندگی را برای راننده لذت بخش‌تر کنند، که امروزه شاهد آن نیز می باشیم.نکته‌ی مهم استفاده بجا و حساب‌شده از نیروی انسانی است. اگر دقت کرده باشید، در کارخانه‌های ساعت‌سازی قدیمی ، کارهای نصب چرخ و دندهٔ ساعتها را به خانمها می دادند. علت ظرافت و دقت بوده است . در اینکه خانمها در سلیقه حرف اول را می زنند، حرفی نیست. پیشنهاد استفاده از خانم گرافیست تحصیل‌کرده می‌تواند یک گزینه مناسب باشد. به هر حال شرکتهای بزرگ تجاری از اینگونه ریزه کاریها بهره جستند و موفق شده اند.

۱-۴- اهمیت رنگ در طراحی وب :

ما در هنگام صحبت با دیگران علاوه بر حرف زدن کارهای دیگری نیز مانند خندیدن ، اشاره کردن ، نگاه کردن ، عصبانی شدن و غیره انجام میدهیم . تمام این کار ها در ارتباط برقرار کردن ما با مخاطب تاثیر دارد و گاهی بسیاری از حرف ها را با چیزی غیر از زبان بیان میکنیم . یک صفحه وب نیز باید بتواند علاوه بر

انتقال مطالب ، احساس طراح وب آن صفحه را نیز بیان کند . این کار را رنگ ها در طراحی وب انجام میدهند . حس خشم ، محبت ، شادی و حتی اعتماد از طریق رنگ ها به بیننده منتقل میشود .

انتخاب بهترین رنگ برای طراحی وب سایت

آیا شما بهترین رنگ را برای وب سایت خود بکار گرفته اید ؟ هنگام انتخاب رنگ در زمان طراحی باید به نکات زیر توجه کنید :

۱- اثر روان شناسی رنگ
۲- قابلیت خواندن متون صفحات سایت
۳- رنگ متمم رنگهای انتخابی برای بک گراند ، گرافیک ها ، لینک ها و متون

لیست زیر مشخصه هایی از رنگ ها را که هنگام طراحی باید مدنظر داشته باشید بیان میکند:
۱- رنگ ها اثر زیادی روی احساسات ما در ۹۰ ثانیه اول دیدن میگذارند .
۲- اثر رنگ میتواند بیننده را ترغیب به خرید یک جنس از شما کند.
۳- رنگ ها رفتار ما را در برابر یک موضوع فقط تشدید نمیکنند ، بلکه اثر خود را کاملا در رفتار ما نشان میدهند .
۴- اثر گذاری رنگ در فرهنگ های مختلف گوناگون است !
۵- هر رنگ به تنهایی یک پیام مخصوص به چشم بیننده میفرستد .
۶-
در فرهنگ آمریکای شمالی رنگ های زیر به احساسات یا مشخصه های زیر مربوطند:

- **سفید :** اشاره دارد به صداقت ، پاکیزگی ، صمیمیت ، ملایمت و معاصر بود ن چیزی . سفید بهترین رنگ برای بک گراند های وب است . در تجارت سفید رنگ خستگی گیر و انرژی بخش است .
- **سیاه :** اشاره دارد به ظرافت ، قدرت ، دلیری ، شهامت ، فریبندگی ، شیطان ، مهارت و باستان . مشکی برای رنگ متن روی یک پس زمینه روشن ایده آل است. این رنگ بعنوان رنگ پس زمینه چشم را خسته میکند .
- **قرمز :** توانایی ، سکس ، هیجان ، احساسات شدید ، سرعت ، خطر و تهاجم . این رنگ از بیننده توجه به خود را طلب میکند . در تجارت بمعنی وام و بدهی است . این رنگ شدید ترین رنگ احساس است و ضربان قلب و تنفس را تسریع میکند .
- **آبی :** امنیت ، اعتماد ، مسئولیت پذیری ، سرما ، ایمان ، وفاداری ، وابستگی وجاه و جلال . آبی دومین رنگ عوام پسند است . در تجارت بمعنای ضمانت مالی است .

- **سبز :** تندرستی ، فراوانی ، حاصلخیزی ، آزادی ، شفا و بهبودی ، طبیعت ، پیشرفت ، حسادت و خونسردی . در تجارت بیانگر مقام و ثروت است. این رنگ در چشم ها راحت تر از همه دیده میشود .
- **قهوه ای :** تاثیر گذاری ، متانت ، توانگری مالی و کمک کننده بودن . قهوه ای رنگ کره خاکی ماست و در طبیعت بسیار فراوان است .
- **خاکستری :** صمیمیت زیاد ، اعتبار و نفوذ و عملی بودن . در تجارت بمعنای سنت گرایی است .
- **صورتی :** ملایمت ، شیرینی ، ظرافت و زنانگی ، خوب بودن ، بی گناهی و پرورش کودک .
- **بنفش :** وقار ، معنویت ، شاهانه بودن ، عیش و نعمت ، دارایی ، اعتبار و نفوذ ، سوگواری و مهارت . در تجارت بزرگ نشان دادن است . بنفش طرفداری از سبک های هنرمندانه است.
- **نارنجی :** سرزندگی و شوخی ، لذت و خوشگذرانی ، تعادل گرمایی ، تشویق کردن ، چالاکی و نیرو ، تحمل و بلند همتی .
- **زرد :** نور خورشید ، گرمی ، خوشی ، نامردی ، ترسویی و حسادت . در تجارت درخواست از نوع روشنفکرانه است و برای تاکید نیز خوب است . زرد باعث افزایش تمرکز شده ، سوخت و ساز را زیاد میکند و سخت ترین رنگ برای چشم هاست .
- **طلایی :** نشانگر گران بودن و پرستیژ است .
- **نقره ای :** سرما ، علمی بودن و اعتبار و پرستیژ است .

پس وقتی میخواهید رنگی انتخاب کنید باید درباره بازار نهایی کار خود باندیشید . چه احساساتی را میخواهید برانگیخته کنید ؟ اول کمی درباره چشم انداز احساسی هدفتان فکر کنید و نیز پیامی که از راه دید میخواهید منتقل کنید . بعد رنگ خود را انتخاب کنید !

رنگ ها و معانی آنها :

رنگ سبز و سفید با هم ترکیب مناسبی بوجود می آورند . اما در ژاپن یک گل میخک صد پر سفید نشان مرگ است و در چین کلاه سبز به معنای خیانت یک زن به شوهرش معنی میدهد . کلاه سبز با میخک سفید نشان زیبایی برای لوگوی یک شرکت نمیتواند باشد . گرچه سبز رنگ آرامش بخش است (بهمین دلیل در بیمارستان ها استفاده میشود) و در ضمن رنگ سبز راحت تر از بقیه رنگ ها در چشم انسان آنالیز میشود . ترکیب های گوناگون رنگ سبز معانی گوناگونی میدهد . ترکیب سبز و زرد کمترین طرفدار را بین بینندگان دارد .

قرمز برای افزایش فشار خون و سرعت ضربان قلب استفاده میشود . افرادی که در محیط قرمز کار میکنند معمولا سریعتر کار میکنند ، اما اشتباهاتشان در کار بیش از دیگران است . این رنگ امیال درونی مثل اشتها ، بی قراری و تنش عصبی را افزایش میدهد . ایجاد یک سایت با دو رنگ آبی کمرنگ و قرمز کمرنگ ایده بسیار ضعیفی است .رنگ قرمز ملایم طولانی ترین طول موج را داراست و آبی ملایم کوتاه ترین را . هنگام

نگاه کردن انسان به این دو رنگ ، لنز چشم برای تنظیم زوم تغییر اندازه میدهد . اما چون فرکانس های رنگی این دو رنگ با هم خیلی تفاوت دارد باعث خستگی چشم و ایجاد سردرد برای بیننده میشود .

سایت هایی که از از سایه های گوناگون آبی یا رنگ آبی و سفید استفاده میکنند بیشتر از بقیه ، مردم پسند بنظر می آیند . چرا ؟ آبی آرامش ، استواری ، امید داشتن و دانایی و بخشندگی را عرضه میدارد . مردم ذاتا به سایت های آبی رنگ سریعتر اعتماد میکنند . متن های آبی رنگ بیشتر در ذهن مردم به یاد میماند . ترکیب آبی و سفید و بنفش نجابت و اصالت را در ذهن انسان میسازد .

خدا را شکر که سایت های زرد رنگ زیاد نیستند . با وجود اینکه رنگ زرد نشان از جمع شدگی و تمرکز است ، اما بسیار سخت و مشکل در چشم آنالیز میشود . رنگ اطاق را زرد کنید: نتیجه این میشود که بچه ها گریه بیشتری میکنند و بزرگتر ها سریعتر عصبانی میشوند. رنگ زرد یک رنگ حسی و چشم فریب است و استفاده آن در میزان کم بسیار جذاب و خوش دید خواهد بود .

بذار کمی درباره نارنجی حرف بزنیم ،نارنجی باعث میشه که اجناس گرون قیمت در نظر افراد مناسب و خوب جلوه کند . نارنجی روشن تر سخت تر در چشم دیده میشود . و به عنوان متن ها و بک گراند صفحه پیشنهاد نمیشود . مقدار کمی نارنجی کمرنگ میتونه کمک کنه که یه سایت شاد و دلچسب بسازید .

عمل و عکس العمل :

رنگ روی حس ما و ادراک ما و عکس العمل ما تاثیر میگذارد . یک بیننده با آگاهی از سایتتان وارد سایت شما شده است ، حال شما باید او را همچنان سر شوق نگه دارید .شما ۸ الی ۱۰ ثانیه وقت دارید تا او را از لحاظ بصری جذب کنید . با استفاده از رنگ شما میتوانید حس خوش آمد گویی و راحتی و اعتماد را به بیننده منتقل کنید . اگر شما گرافیک های یک سایت را تغییر دهید در اصل زبان گفتگوی تصویری آن را عوض کرده اید و بدین سان است که عکس العمل متفاوتی از بازدیدکننده سایت سر میزند.

قرار دادن یک محصول که با آب سر و کار دارد (مثل لباس شنا و ...) در زمینه بنفش و نارنجی قدرت فروش آن را کاهش میدهد . رنگ های بنفش و نارنجی بطور سریع با آب و طبیعت ارتباط ندارند و یک ادراک غلط به بیننده منتقل میکنند . قرار دادن همان محصول در رنگ آبی یا سبز باعث افزایش جذابیت آن محصول میشود .

سایت هایی که اصطلاحا سایت های رنگین کمانی هستند و از همه جور رنگ در هم استفاده کرده اند کمترین زمان بازدید توسط بیننده ها را دارند و بیننده خیلی سریع ازین سایت ها خارج میشود . زیرا چشم برای دیدن رنگ های گوناگون زوم های گوناگونی میکند و زود خسته میشود . (سایت هایی که رنگ سفید

در آنها غالب است و مقدار خیلی کمی از دیگر رنگ ها در آن در بخش های گوناگون دیده میشود جزو این دسته نمیشوند) . هر چه تعدد رنگ ها کم شود مدت زمان ماندن بازدید کننده در صفحه زیاد میشود . یک نکته در استفاده از رنگ های گوناگون اینه که حداکثر از ۵ رنگ استفاده کنید و آنها را همگی از رنگ های گرم یا سرد استفاده کنید و زمینه را سفید بگذارید . این کار چشم ها را خسته نمیکند و سایت های کودکان که میخواهند با استفاده از رنگ های گوناگون محیط شادی را ایجاد کنند اگر ازاین روش استفاده کنند فروش بهتری دارند .

رنگ های گرم و سرد :

رنگ های گرم تشکیل شده اند از : زرد ، نارنجی ، قهوه ای ، زرد–سبز و نارنجی– قرمز . یعنی رنگ هایی که با پاییز در تعامل هستند . بطور کلی رنگ های گرم گرایش به هیجان و تکاپو دارند . بسیاری از مردم مقدار کم از این رنگ ها را می پسندند . بنفش و سبز رنگ های واسط هستند که نه گرم و نه سردند و وابسته به میزان رنگ قرمز یا زردی هستند که در رابطه با رنگ آبی در آنها بکار رفته است . اگر رنگ آبی آن کم تر باشد بیشتر شبیه رنگ های گرم بنظر میرسد .

رنگ های سرد عبارتند از : آبی ، سبز ، صورتی ها ، بنفش ها ، آبی–سبز ها ، سرخابی ها و آبی–قرمز ها . رنگ هایی که بیشتر با بهار و تابستان در تعامل هستند . رنگ های سرد آرامش بخش هستند و محبوبیت بیشتری نزد مردم دارند . طراحی یک سایت با رنگ های گرم و سرد باعث گیج شدن بیننده میشود و باعث میشود سایت شلوغ و بی نظم و غیر قابل اعتماد جلوه کند . طراحان سایت معمولا متوجه نمیشوند که ترکیب رنگ هایشان گرم و سرد است . استفاده از چرخه رنگ ها مفید است . دایره رنگ رنگ های اصلی (آبی و قرمز و زرد) و رنگ های ثانویه (نارنجی و سبز و بنفش) را نشان میدهد . ترکیب دو رنگ اصلی یک رنگ ثانویه میسازد . تمامی رنگ ها از ترکیب سیاه و سفید با رنگ های اصلی بوجود آمده اند .

۱-۵- چند نکته برای طراحی یک وب سایت خوب

۱. در برخی موارد وقتی که کاربران وارد سایت شما می شوند، به علت کم بودن سرعت اینترنت ویا تنظیمات کاربران ممکن است که عکسهای سایت ، در سایت قرار نگیرند و به عبارت دیگر **download** نشوند. بنابراین سعی کنید که در پشت عکسهایی که در صفحه قرار می دهید، توضیحاتی قرار دهید تا در صورت قرار نگرفتن انها در صفحات، کاربران سر در گم نشوند.

۲. صفحات خود را طوری طراحی کنید که دارای حجم کمتری باشند. برای این کار در صفحات خود از عکسهای کمتری استفاده کنید و تا حد امکان از عکسهایی با حجم کمتر استفاده کنید. برای کم کردن حجم عکسهایتان از یک نرم افزار حرفه ای مثل **Photoshop** استفاده کنید.
۳. تا حد امکان در صفحاتتان از **Frame** های کمتری استفاده کنید .**Frame**های زیاد در صفحات ، باعث افزایش حجم ودر نتیجه کاهش سرعت بالا امدن صفحاتتان میشوند.
٤. در صورتی که در فرم وب خود از **CheckBox** ها یا **RadioButton** ها برای دریافت اطلاعات از کاربران استفاده می کنید ، سعی کنید که محدوده قابل کلیک کردن روی اجزای فرمها را افزایش دهید بطوری که کاربر مجبورنباشد دقیقا روی فیلد مورد نظر کلیک کند.
٥. در بسیاری از صفحات دیده میشود که صفحات دارای لینکهای تو در تو وبسیار پیچیده ای هستند. این صفحات فقط باعث سر در گمی کاربران میشوند. برای جلوگیری از سر در گمی کاربران برای سایت خود یک نقشه تهیه کنید و طریقه اتصال لینکها و صفحات را بصورت گرافیکی ترسیم کنید.
٦. برای انکه در طراحی وب سایتها حرفه ای شوید ، سعی کنید سایتهای قوی و حرفه ای را مورد بررسی قرار دهید.
۷. در پایان به این نکته توجه کنید که زیبا بودن یک سایت ، یک کاربر را حداکثر دو یا سه بار به ان سایت جذب می کند و او را به سایت باز می گرداند اما وجود اطلاعات مفید در سایت ، کاربران زیادی را خود جذب خواهد کرد. بنابراین برای قرار دادن اطلاعات مفید در سایت خود تلاش بیشتری کنید

نکات منفی وب سایت ها

وب سایت ها را می توان بسیار زیبا و با استفاده از برنامه فلش ، موسیقی ، انیمیشن و ... ساخت.البته بعضی افراد این ویژگی ها را میپسندند ولی برای بسیاری از کاربران اینها جزو ویژگی های منفی محسوب می شود.

ابتدا باید مخاطبین خود را بشناسید.اگر وب سایت شما دارای محتوایی است که باید توسط بازدیدکنندگان خوانده شود جداً خودداری کنید.سادگی را در ابتدای کارها قرار دهید و محبوبترین سایت های اینترنتی مانند **Yahoo , Google** و **Excite** را همیشه در نظر داشته باشید.در زیر ۲۶ خطای معمول در طراحی صفحات وب آمده :

۱. **معبر ورودی**

منظور یک صفحه معمولا حاوی فلش قبل از صفحه خانه (**Home page**) است.حتی اگر این صفحه برای منظور مفیدی نظیر انتخاب زبان طراحی شده باشد.راه های بهتری نیز برای این کار وجود دارد.

۲. **موسیقی و یا کلیپ تصویری ناخواسته**

قبل از اینکه بازدیدکننده وب سایت مجبور به بارگیری (**Load**) موسیقی یا فیلم شود از او اجازه بگیرید.اگر او مایل به این کار نیست متن جایگزین را در اختیارش قرار دهید.

۳. **متن دارای اسکرول**

استفاده از اسکرول بار برای حرکت در متن مخصوصا در مورد متنهای مهم که باید با دقت خوانده شوند باعث حواس پرتی می شود.

۴. **استفاده از فریم ها در صفحه آغازین**

فریم ها باعث سردرگمی موتورهای جستوجو می شوند و بارگیری صفحه را کند می کنند.به سادگی می توان به جای آنها از جداول استفاده کرد و همان اثرات را به دست آورد.

۵. **بطور کلی استفاده از فریم ها**

چاپ کردن محتوای فریم ها مشکل است،به سختی میتوان به صفحه قبل بازگشت امکان علامت گذاری (**bookmark**) وجود ندارد و مرورگرهای صوتی (**voice browsers**) دچار مشکل می شوند و معمولا فضای کافی برای قراردادن اطلاعات مفید در آنها وجود ندارد.تنها مواقعی که ممکن است مفید باشند که بخواهیم برای محدوده ای از پیام های کوتاه مانند نام و نشانی شعبات استفاده کنیم.

۶. **فرمهای پیچیده**

قانونی که باید رعایت شود این است که همیشه حداقل اطلاعات لازم برای یک هدف معین از کاربر پرسیده شود.پرسیدن سوالات زیاد بازدیدکننده را فراری می دهد.

۷. **قرار دادن تبلیغات به ویژه تبلیغات پاپ-آپ**

هرگز این کار را نکنید مگر اینکه واقعا بابت آنها پول زیادی به شما بدهند.مخصوصا تبلیغاتی که حاوی فلش هستند یا تبلیغات **Pop-up** یا هر نوع دیگر.

۸. **استفاده از Flash**

برای بسیاری از کاربران باعث سردرگمی است و مشاهده متنهای کنار آنها فعالیت ذهنی زیادی می طلبد.

۹. **Plug-in های نا خواسته**

تعداد کمی از بازدیدکنندگان برای دانلود **Plug-in** وقت می گذارند.اگر واقعا وجود آنها را لازم می دانید بهتر است آن را در یک صفحه جداگانه قرار دهید.

۱۰. تصاویر بزرگ

وجود آنها فقط در صفحات هنری قابل توجیه است.حتی در آنجا نیز بهتر است از عکسهای کوچک (بندانگشتی) استفاده کنید و فقط در صورت درخواست بازدیدکننده تصویر بزرگ ارایه شود.

۱۱. استفاده از رنگ های بسیار

مرورگرهای برخی از کاربران فقط برای ۲۵۶ رنگ تنظیم شده است بنابراین ممکن است بعضی از تصاویر وقتی در این حالت قرار گیرند بسیار وحشتناک دیده شوند.بنابراین بهتر است تصاویر کلیدی و مهم دارای رنگهای پیچیده و مختلط نباشند.

۱۲. تصاویر غیرضروری

یک تصویر به اندازه هزاران کلمه ارزش دارد اما تنها هنگامی که به متن کاملا مربوط باشد.

۱۳. تصاویر مورد نیاز

بعضی از بازدیدکنندگان ممکن است برای سرعت بخشیدن به لود صفحه نشان دادن تصاویر را غیرفعال کنند.خوب است در این حالت نیز آنها برداشت مناسبی را از وب سایت به دست آورند.

۱۴. استفاده از تصویر به جای لینک

از تصاویر یا جاوااسکریپت به عنوان لینک استفاده نکنید.لینک های متنی را نیز در قسمت پایینی صفحه قرار دهید.این کار هم به موتورهای جستوجو کمک می کند و هم به بازدیدکنندگان که از تصاویر و جاوااسکریپت استفاده نمی کنند.

۱۵. تصاویر بدون اندازه (unsized images)

اگر تصویری دارای طول و عرض نباشد ممکن است متن پس از لود شدن تصویر پرش پیدا کند و یا تا قبل از لود شدن کامل تصویر نمایش داده نشود.

۱۶. ALT= tags بی فایده

تمام تصاویر باید دارای **ALT= tags** باشند تا بازدیدکننده بداند منتظر دیدن چه چیزی است.اطلاعات جلوی **ALT=** هم باید با معنی و مفید باشد.

۱۷. پس زمینه های مبهم

اگر شما از تصاویر پس زمینه استفاده می کنید باید حجم آنها کمتر از **k8** باشد و با رنگ متن متناسب باشند.بعضی از پرینترها را می توان طوری تنظیم کرد که از پرینت تصاویر پس زمینه همراه متن صرف نظر کنند پس کنترل کنید که متن بدون پس زمینه قابل خواندن باشد.

۱۸. جاوااسکریپت غیر ضروری

با **DHTML** و جاوااسکریپت همه کار می توان کرد اما لود شدن آنها زمان می برد.

۱۹.اخطار در مورد آپگرید مرورگر

امروزه پیغام **Best viewed with** یک پیغام عادی است اما این که ویزیتور چه مرورگری داشته باشد انتخاب شخصی خود اوست و اغلب خوشش نمی آید که از او خواسته شود مرورگرش را آپگرید کند.وب سایت شما باید اجازه دهد که صفحان در مرورگرهای دیگر نیز حتی الامکان به خوبی دیده شود.

۲۰.مشکلات نمایش ۶۴۰

بعضی از صفحات وب دارای عرض ثابتی هستند و هنگامی که با نمایشگرها یا پنجره های نمایش کوچکتر استفاده شوند، یا هنگام چاپ قسمتی از آنها بریده می شود و یا اسکرول افقی پیدا می کنند.متنها باید کوچک شوند تا برای پنجره های کوچکتر هم مناسب باشند.

۲۱.لینک های شکسته

لینک های خارجی ناپدید می شوند و این باعث افسوس است.اما تمام لینک های داخلی باید کار کنند.اگر شما یک یک صفحه را جابه جا می کنید یا نام یک صفحه را تغییر می دهید صفحه قدیمی را با یک انتقال خودکار به صفحه جدید باقی بگذارید.

۲۲.در حال ساخت یا Watch this Space

اگر صفحه ای آماده رویت نشده و در حال ساخت می باشد تا اتمام کامل کار به آن لینکی ندهید.

۲۳.متن به صورت تصویر

متن به صورت تصویر قابلیت کپی شدن،جست و جو و مراجعه به وسیله سرچ را ندارند و لود شدن آنها زمان بیشتری را نسبت به لود شدن متن می برد.پس هرگز متن را به صورت تصویر به کار نبرید

۲۴.سبک های متناقض و متفاوت

تمام صفحات از نظر رنگ،فونت،طرز لینک دادن و غیره باید دارای یک سبک باشند و تمامی عناصر از یک سبک طبعیت بکنند.

۲۵.متن هایی که در وسط قرار می گیرند

اگر به عنوان سر صفحه و تایتل استفاده شوند خوب است ولی در صورتی دیگر نه.زیرا بازدید کنندگان انتظار دارند هر سطر متن از راست شروع شود.

۲۶.صفحات عمیق و تودرتو

برای هر موضوع صفحه ای جداگانه درنظر بگیرید.اگر اجبارا یک صفحه طولانی شود در انتهای صفحه عنوان های فرعی را قرار دهید و از فهرست مطالب به آنها لینک برقرار کنید.

در انتها برای داشتن وب سایتی بهتر اعداد زیر را بخاطر بسپاریم :

۱. حراکثر عرض صفحه ۵۰۰ پیکسل
۲. حداکثر سایز صفحه ۴۰ تا ۶۰ کیلوبایت بهمراه تمامی تصاویر
۳. حداکثر اندازه فایل **HTML,10k**
۴. حداکثر زمان تا لحظه قابل خواندن شدن اطلاعات مهم ۸ ثانیه

۱-۶- سایت خود را در موتورهای جستجو ثبت کنید

سایت خود را در کجا ثبت کنیم ؟

هم اکنون هزاران موتور جستجو و فهرست وجود دارند . ثبت یک سایت در تمامی آن ها زمان بسیار زیادی را می طلبد . خبر خوب این که تنها ۱۰ تا ۱۵ موتور جستجو دارای اهمیت بالا می باشند . ۹۰ درصد ترافیک موتورهای جستجو برای سایت شما از این ۱۰ تا ۱۵ موتور جستجو ایجاد می شود .

نکته : معمولا موتورهای جستجوی مختلف از یک پایگاه داده استفاده می کنند ، لذا اگر آدرس سایت خود را در یکی از پایگاه های داده ثبت کنید ، آن سایت در چندین موتور جستجو دیگر نیز قابل بازیابی است .

در اینجا به لیست کوتاهی از چند سایت مهم برای ثبت آدرس سایت اشاره می کنیم :

فهرست ها

Ask Jeeves : www.ask.com

Look Smart : www.looksmart.com

Mamma : www.mamma.com

Opa Directory : www.dmoz.org

Yahoo ! : www.yahoo.com

موتورهای جستجو

All The Web/Fast : www.alltheweb.com

Altavista : www.altavista.com

Direct Hit : www.directhit.com

Ecite : www.Excite.com

Google : www.google.com

Inktomi : www.hotbot.lycos.com

Parseek : www.parseek.com

JamAsp : www.jamasp.com

موتورهای جستجوی زیر نیز در صورت پرداخت به ازای هر کلیک (PPC) ، امکان ثبت مناسبی را در اختیار قرار می دهند

7Search : www.7search.com

OverTrue(Go to) : www.overtrue.com

Sprinks : www.sprinks.about.com

اگر سایت شما کشور خاصی را هدف گرفته است ، با این که به زبان دیگر غیر از انگلیسی نوشته شده است ، شما می تونید سایت خود را در موتورهای جستجوی محلی مهم و نسخ داخلی موتورهای جستجوی بین المللی ثبت نمایید . اگر مارک تجاری مشهوری را در اختیار داریدیا شرکت شما از شهرت خوبی برخوردار است ، می توانید **RealName** را نیز مورد توجه قرار دهید . موتورهای جستجوی مخصوص صنعت نیز برای شرکت هایی که در این حوزه مشغول به کارند حائز اهمیت هستند .

اگر سایت شما ، سایت بسیار بزرگی نیست ، براحتی و ظرف چند دقیقه می توان آدرس آن را در ۱۵ موتور جستجو و فهرست اصلی به ثبت رساند و نیازی به استفاده از نرم افزارهای گران قیمت ثبت نیست .

چگونه یک آدرس را ثبت کنیم ؟

اضافه کردن یک **URL** به صورت دستی معمولا به راحتی با پر کردن یک فرم کوتاه که دارای آدرس سایت و آدرس پست الکترونیک شماست انجام پذیر است . گاهی شما باید یک دسته بندی هم برای سایت خود انتخاب کنید .

نکته : از آدرس پست الکترونیک معمول خود در زمان ثبت سایت در صفحات **FFA** استفاده ، در غیر اینصورت برای مدتی طولانی تعداد بسیار زیادی پیغام های **Spam** را در این آدرس دریافت خواهید کرد و

باعث مزاحمت شما خواهد شد . یک آدرس پست الکترونیکی جداگانه برای ثبت در صفحات **FFA** ایجاد کنید و از آن استفاده کنید . در هنگام انتخاب دسته بندی و پر کردن فرم اطلاعات ، دقت خود را کاملا بکار بگیرید . به طور معمول شما باید تنها آدرس صفحه اصلی سایت خود را در فهرست ها ثبت کنید .

موتورهای جستجو به شما اجازه می دهند که هر صفحه را تا زمانی که محتوای متفاوتی با بقیه دارد به طور مجزا ثبت نمایید . برای اطمینان از انجام کار به طور مناسب بهتر است در هر روز تنها یک صفحه را ثبت نمایید و به موتور جستجو فرصتی به ۴ تا ۸ هفته بدهید که سایت شما را شاخص دهی نماید . یک صفحه را به طور مکرر ثبت نکنید .

معمولا ثبت تمامی صفحات سایت (روزی یک صفحه) سریع ترین راه برای شاخص دهی تمام سایت است . اما بعضی موتورهای جستجو ممکن است در صورتی که صفحات سایت شما مستقیما ثبت نشده باشند و توسط ربات موتور جستجو یافته شوند ، درجه بالاتری به آن ها اهدا کند . سعی کنید تمامی زیر صفحات را به صفحه اصلی پیوند دهید و تنها صفحه اصلی را ثبت نموده و ۲ تا ۳ ماه صبر کنید . پس از آن تنها صفحاتی که توسط روبات یافته نشده اند را دستی ثبت نمایید . ثبت روزانه یک صفحه به عنوان **span** تلقی نخواهد شد و به سرعت بخشیدن شاخص دهی صفحات کمک می کند .

ثبت مجدد

اگر پس از گذشت چند هفته ، هنوز سایت شما لیست نشده است ، احتمالا درخواست ثبت شما گم شده است ، سعی کنید سایت خود را مجددا ثبت کنید . اما پیش از آن مطمئن شوید که سایت شما مشکلی ندارد که از شاخص دهی جلوگیری به عمل آورد . پس از ثبت اولیه شما باید بین ۶ تا ۸ هفته صبر کنید تا تمامی موتورهای جستجو فرصت شاخص دهی داشته باشند . پس از آن کنترل نمایید و ببینید اگر سایت بخوبی شاخص دهی نشده بود یا درجه مناسبی نداشت مجددا ثبت نمایید .

برای پیدا کردن سایتهایی که می توانید سایت خود را در آن ها ثبت نمایید می توانید دنبال کلمه **"AddURL"** در اینترنت بگردید .

۱-۷- چگونه رتبه سایت خود را در یک موتور جستجوگر بالاتر ببریم و ترافیک بیشتری جذب کنیم ؟

هیچ کس نمی تواند انکار کند که سایت های تجارت الکترونیک موفق ، بر اساس ترافیک و بازدید کننده زیاد خودشان بوده که این موفقیت را کسب نموده اند . بنابراین در این بخش سعی داریم تا چند راه ساده را برای افزایش ترافیک سایت به شما معرفی کنیم .

اطمینان دارم که تبلیغات زیادی را درباره یک روش جدید ؟ جادویی ؟ راهی برای افزایش سریع ترافیک سایت خود و بدست آوردن بالاترین رتبه در موتورهای جستجوگر ؟ آن هم در یک شب ؟ شنیده و خوانده اید . و امیدوار هستم بدانید که چرا از این همه علامت سوال استفاده نموده ام . بله " ریشخند و طعنه ! " .

با خواندن این بخش دیگر لازم نیست مبالغ ۲۹ دلار تا ۹۷ دلاری که مدعیان این روش ها از شما خواهند گرفت را پرداخت کنید . رمز معجزه آسای بزرگ اینست : یک وبلاگ راه اندازی کرده و آن را توسعه دهید . همین ! نقطه .

زمانی که شما یک **E-Book** ، ۶۰ صفحه ای (که پر است از مطالب تکراری ، لینک و تبلیغ سایت ها و محصولات مختلف و خروارها نظرخواهی که اتفاقا همه هم راضی هستند) را مطالعه می کنید ، مفهوم و منظور اصلی همین یک کلمه است . یک وبلاگ ، که به خوبی در مورد آن تبلیغ شده و کاملا در جهت فعالیت اقتصادی آنلاین شماست .

خب ، آیا مؤثر است ؟

بله ، مطمئنا موثر است . این افراد ممکن است زیاد تبلیغ کنند ولی کمتر اشتباه می گویند .

بنابراین ، آشنایی با دلایل کلیدی که "چرا" یک وبلاگ در افزایش رتبه موتور جستجوگر و جذب ترافیک بالا برای سایت موثر است ، مهم می باشد .

اول ، یک ولاگ ، ذاتا دارای مطالبی است که بصورت دوره ای بروز رسانی می شوند . موتورهای جستجو همیشه جذب مطالب تازه می شوند ، چون کاربران به دنبال چنین مطالبی هستند .

دوم ، ابزار **RSS** یا **Atom** بصورت مستقیم مطالب درج شده شما را در قالب "اخبار" یا "**news feed**" به موتور جستجوگر ارسال می کند . بنابراین ، هربار که وبلاگ خود را بروز رسانی می کنید ، بدون تحمل دردسرهای تسلیم سایت به موتور جستجوگر و تبلیغ آن ، رتبه بالاتری را در موتور جستجوگر کسب می نمایید .

سوم ، مشترکان خبرنامه یک وبلاگ ، بصورت خودکار ایمیل های این وبلاگ را بدون آن که توسط نرم افزارهای متعدد فیلتر شود در صندوق پستی خود دریافت می کنند .

خوب به نظر می رسد ؟! البته ، چون شما هنوز از " شش دلیل اشتباه بودن استفاده از وبلاگ " برای کسب و کار آنلاین خود خبر ندارید :

۱) این صفحات ترافیک هدفمند زیادی را به سمت خود جلب می کنند . کاربران آنلاین ممکن است عناوین مطالب وبلاگ شما را بخوانند و با هدف تبلیغاتی شما آشنا شوند ولی از اصلا از سایت شما بازدید نکنند .

۲) بسیاری از سیستم های میزبان ، وبلاگ شما را وابسته به میزبان ها ، خدمات دهندگان و یا بسته های نرم افزاری دیگر می کنند . اطلاع داشته باشید که چندین میزبان بلاگر بزرگ در چند سال اخیر ورشکسته و بسته شده اند .

۳) اکثر شرکتهای بلاگر امروزی اقدام به درج آگهی و تبلیغات در صفحات وبلاگ شما می کنند . از آن جایی که اینگونه تبلیغات بسیار حساس به نوع متن (**context-sensetive**) هستند شما ممکن است در بلاگ خود برای رقبایتان هم تبلیغ کنید و هم پهنای باند بسازید .

۴) وبلاگ باید بصورت دائم بروز رسانی شود در غیر اینصورت رتبه خود را از دست می دهد . شما واقعا چند وقت یکبار فرصت کافی برای بروز رسانی وبلاگ خود دارید ؟ و چه مقدار وبلاگ شما می تواند به موضوع تجاری و کاری شما نزدیک باشد ؟

۵) هنوز مشکلات قانونی متعددی در ارتباط با تجارت و تبلیغات مبتنی بر وبلاگ وجود دارند که باید حل شوند . در کشور آمریکا هنوز برخی قوانین ، باعث ایجاد محدودیت در عملکرد وبلاگ ها شده اند . ممکن است بصورت خیلی غیر منتظره بخاطر حرف یا نوشته خود مسوول شناخته شده و جریمه شوید .

۶) به لطف تبلیغات بسیار گسترده ای که برای وبلاگ ها صورت پذیرفته است ، بصورت متوسط در هر روز نزدیک به ۴۰۰۰۰ وبلاگ جدید ساخته می شوند که اغلب آن ها فقط دارای قالب وبلاگ بوده و هیچ محتویاتی ندارند . البته زمان زیادی طول نخواهد کشید تا موتورهای جستجوگر یاد بگیرند اینگونه بلاگ ها را از فهرست نتایج خود حذف کنند .

اجازه دهید به دلیل شماره یک افزایش ترافیک سایت شما توسط وبلاگ مراجعه کنیم : " بروز رسانی مطالب به صورت دوره ای ! " ببینید ، این وبلاگ نیست که معجزه می کند ، بلکه به روز رسانی مرتب مطالب است . و نکته دیگر اینکه بروز رسانی دوره ای مطالب به معنای داشتن وبلاگ نیست !

یک میانبر محرمانه وجود دارد : صفحه نخست سایت خود را بصورت مرتب بروز رسانی کنید ! چگونه ؟ یک طرح ساده تغییر یا جایگزینی مطالب تازه و مربوط را در نظر بگیرید . مطمئن باشید که در هیچ جای دیگری در سطح وب در مورد این موضوع صحبت نشده است . تا آن جایی که من می دانم از ابتدای سال ۲۰۰۶ تا بحال ، بصورت یک راز باقی مانده است .

فقط کافیست بصورت دوره ای مطالب صفحه اول (**home page**) سایت خود را بروز رسانی کنید . چندین بند یا پاراگراف مختلف بنویسید که به توضیح جنبه های مختلف فعالیت شما می پردازند . چند صفحه اول (**index**) بسازید و در هر کدام از چیدمان مختلف با استفاده از پاراگراف های مختلف استفاده کنید . در ابتدای پاراگراف به معرفی یک سرویس بپردازید ، در ادامه آن را تشریح کنید و جزییات آن را بیان کنید و در پایان نیز به معرفی سرویس و خدمات دیگر بپردازید . مثلا اینکه " ما محصولات دیگری نیز مرتبط با ... ارایه کرده ایم که ... "

حتی ممکن است آغاز و پایان بندها یکسان باقی بمانند ولی قسمت توضیحات فرق کنند ، باز هم سودمند خواهد بود .

نکته : می توانید محتویات صفحه نخست می تواند بر اساس رویدادها یا مواردی که بصورت فصلی مورد توجه کاربران است تغییر یابد .

توجه : بندها و پاراگراف های مختلف در صفحه باید حتما دارای مطالب متفاوتی باشند . شما نمی توانید یک مطلب را به چند شکل در یک صفحه بیان کنید . حالا که دارید وقت می گذارید و زحمت می کشید ، کار را درست انجام دهید .

وقتی که چند صفحه اول (**index**) مختلف ساختید ، هر چند روز یکی از آن ها را جایگزین کنید . چه اتفاقی می افتد ؟ خب ، وقتی که نرم افزار **spider** یا کاوشگر موتورهای جستجوگر به سایت شما مراجعه می کند ، با مطالب جدید در صفحه نخست مواجه می شود . بار دیگر هم با مطالب متفاوت بروز رسانی شده روبرو می شود . اینجاست که نرم افزار کاوشگر به فکر می افتد (البته نه ، هیچ نرم افزاری فکر نمی کند ، آن ها فقط چند تابع ریاضی هستند و همین رمز موفقیت این تکنیک می باشد) تا این سایت را زیر نظر بگیرد و بیشتر به این سایت سر بزند ، با مطالب (بین خودمان بماند ، منظور **index**) جدید و بروز رسانی شده ملاقات می کند . آیا نرم افزار آنچه را که امروز یافته با مطالبی که در پنج کاوش قبل یافته مقایسه می کند ؟ بله ، ولی برای این کار زمان خیلی زیادی لازم است .

نکته مهم : درست مانند وبلاگ نویسی باید این کار را بصورت دستی و غیر خودکار انجام دهید . استفاده از کدهای جاوا و سیستم تغییر صفحه یا مطالب خودکار هیچ سودی نخواهد داشت . البته وقتی که شما مطالب را از قبل آماده داشته باشید ، تغییر صفحه یا مطالب کار وقت گیر نخواهد بود . بجای اینکه هر روز نگران فراهم نمودن مطالب جدید برای وبلاگ خود باشید ، بسادگی می توانید مطالب از قبل آماده شده را در صفحه جایگزین کنید .

در طی چند ماه شما به کاوشگر موتور جستجو آموخته اید که باید بیشتر به سایت شما سر بزند . و اگر سایت شما مطالب قابل ثبت دیگر داشته باشد ، این مطالب شروع به پیدا کردن راهی به رتبه های بالا در نتایج جستجو خواهند کرد . هر مطلبی که به سایت خود بیافزایید سریعتر ایندکس و ثبت می شود . و از آنجایی که سایت شما به طور مداوم مورد بازدید موتورهای جستجوگر قرار دارد ، برای هر تبلیغ در کوتاه مدت بازدید کنندگان بیشتری خواهید داشت . و اینجاست که به مرحله دوم طرح می رسیم .

مرحله دوم داشتن یک تبلیغ یا سفارش روز یا هفته است . چندین محصول یا تبلیغ یا خدمات را به صورت دوره ای در صفحات اول در لابه لای مطالب دیگر درج کنید . می توانید این مرحله را همزمان با مرحله اول انجام دهید . و مطالب را همراه با تبلیغات یا پیشنهادات فروش مختلف در صفحات تغییر دهید . با این کار تعداد **index** بیشتری خواهید داشت . و احتمال پیدا شدن نمونه های تکراری یا مشابه کاهش می یابد .

نکته کلیدی : این راهبرد بر اساس اصل ساده کردن ایجاد و تغییر دوره ای مطالب مربوط به سایت می باشد . صفحه اول شما همیشه در دید بوده و یک پیام می رساند . ولی در هر بار به شکل متفاوتی در برابر جستجوگر ظاهر می گردد . فوق العادست . این روش مزیت منحصر بفردی دارد و آن هم بهینه کردن بازدید **spider** یا کاوشگر موتورهای جستجوگر به قلب وب سایت است نه اینه آن ها را به سمت یک وبلاگ دور افتاده بکشاند .

آیا این روش می تواند به اندازه راه اندازی یک وبلاگ مرتب با مطالب بروز رسانی شده سودمند باشد ؟ خیر ! این روش مزیت برخورداری از گزینه **news feed** یا **RSS** موجود در بلاگرها که یکی از دلایل موفقیت وبلاگ می باشد را ندارد . و دیگر اینکه شانس دوبار حضور آنلاین شما را که می تواند دو فرصت تبلیغاتی و شانس برخورد با بازار هدف باشد را از شما می گیرد . اما حتی اگر این روش به اندازه نصف تاثیر یک وبلاگ ، سودمند باشد (که البته تاثیر بیشتری دارد) ، فقط ۲٪ تلاش یک وبلاگ را خواهد داشت . و فرصت خوبی برای شما جهت ایجاد و تبلیغ حرفه و سایت خود فراهم می کند .

هدف ، بدست آوردن حداکثر نتیجه خوب با حداقل تلاش است .

۸-۱- راه رسم رونق تجارت با Google

در این بخش ، برآوردی از تجارت در **Google** و خدماتی که به منزله سکوی پرتاب کاربران است ، ارائه شده است . اکثرا با صورت پرکاربرد **Google** آشنایید : موتور جستجو ، ولی **Google** چیزی فراتر از یک موتور جستجو است . **Google** شرکتی بر مبنای فناوری است که بر اساس برنامه های تجاری خود شناخته می شود .

هدف هر طراح سایت

سایت ها و فعالیت های تجاری گوناگون و بی شماری بر روی اینترنت شکل گرفته است . سایت هایی مانند فروشگاه های اینترنتی ، آژانس های مسافرتی ، مجلات ، پورتال ها و غیره ، اما تمام این سایت ها ، در ۳ هدف اساسی خود با هم مشترکند :

۱- **حضور در وب**

راه اندازی یک وب سایت ، مانند قرار دادن یک بیلبورد در بیابان است . هیچ کس آن را نخواهد دید . حضور در نتایج جستجوی **Google** ، به معنی بازدید کنندگان بیشتر است و هر چه در این فهرست جایگاه شما بالاتر باشد ، شانس بیشتر برای بیشتر دیده شدن دارید .

۲- **افزایش بینندگان**

افزایش بینندگان ، در واقع به معنی گسترش دیده شدن است . اما دیده شدن به تنهایی کافی نیست ، حضور در واقع با قرار گرفتن در نتایج جستجوی **Google** به دست خواهد آمد ، ولی آن چه که هدف شما را از طراحی سایت ارضا خواهد کرد ، ترافیک بینندگان است و این اتفاق زمانی خواهد افتاد که کاربران در لیست نتایج جستجوی **Google** ، روی نام سایت شما کلیک کنند .

۳- **جهت دادن به بازدید کننده**

بازدید بالا و ترافیک ، ممکن است هدف تعدادی از طراحان سایت باشد ، اما بیشتر طراحان سایت های اینترنتی هدف های بالاتر دیگر را از طراحی وب سایت خود انتظار دارند ، هدف هایی مانند پر کردن یک فرم مخصوص توسط بازدید کننده ، ملحق شدن به لیست نامه های الکترونیک سایت ، خرید کردن از سایت و ... در واقع برآورده شدن این هدف ، اصلی ترین دستاورد یک طراح سایت است .

Google بهترین ابزاری است که برای رسیدن به دو هدف اول به شما کمک خواهد کرد . حتی اگر موقعیت شما در لیست نتایج جستجوی **Google** به اندازه کافی به شما کمک نمی کند ، برنامه تبلیغاتی **Google** یا همان **AdWords** برای افزایش بازدید کنندگان سایت شما بسیار مؤثر خواهد بود .

Google و وب سایت شما

برنامه های تبلیغاتی **Google** موسوم به **AdWords** و **AdSense** برای هر دارنده وب سایتی فرصت بسیار خوبی به شمار می آید . این برنامه های تبلیغاتی ، سه هدف اساسی را در کسب موقعیت و گسترش فعالیت تجاری دنبال می کند :

۱- قرار گرفتن در لیست جستجوی Google

قرار گرفتن در موقعیت بهتر در لیست نتایج جستجوی **Google** به عنوان گام اول است . ممکن است بسیاری از طراحان و دارندگان وب سایت ، بدون اینکه اقدام به تبلیغ سایت خود کنند ، تنها بر روی بهبود وضعیت سایت خود در نتایج جستجو ، متمرکز شوند .

۲- **AdWords**

AdWord یا برنامه تبلیغاتی ، در نتایج جستجوی **Google** در واقع ترافیک بازدید سایت شما را بسیار افزایش خواهد داد . **Adword** همان تبلیغاتی است که در نتایج جستجود در بالا و سمت راست صفحه دیده می شود . آگهی دهنده ها ، تنها در صورتی که کاربران **Google** روی آن ها کلیک کنند ، بهای آن را خواهند پرداخت . این برنامه تبلیغاتی برای شرکت ها یا دارندگان سایت هایی مفید است که نمی خواهند روی بالابردن رتبه ی سایت خود در نتایج جستجوی **Google** متمرکز شوند و به دنبال یک راه سریع تر برای قرار گرفتن در صفحات اول نتایج جستجو می گردند .

۳- **AdSense**

AdSense یا برنامه مشارکت تبلیغاتی **Google** ، راهی برای کسب درآمد در سایت شما است . طراحان و دارندگان سایتی که در این برنامه مشارکت می کنند ، با قرار دادن تعدادی از تبلیغات **Google** در سایت خود ، می توانند درآمد حاصل از این آگهی ها را با **Google** سهیم شوند . این تبلیغات توسط **Google** آماده شده و در سایت شما قرار می گیرد و به ازای هر بیننده ای که از سایت شما دیدن کرده و روی این تبلیغات کلیک می کند ، **Google** درآمد هر بار کلیک روی این آگهی را با شما شریک می شود. مشارکت در این طرح ، برای طراحان سایت هیچ هزینه ای ندارد و مشترکان این طرح ، برای تغییر برخی از صفحات و یا کل سایت اختیار کامل دارند .

به جرأت می توان گفت که سه امکانی که شرح داده شد ، هر کدام معادل یک فعالیت اقتصادی تمام عیار است . درک نحوه ی صحیح عملکرد این سه فعالیت و توانایی شما در به خدمت گیری آن ها ، می تواند در استفاده ی صحیح از هر کدام از سرویس ها به شما کمک کند . فاکتورهایی را که در این زمینه ، باعث بهبود عملکرد شما خواهند شد ، می توان به این صورت خلاصه کرد .

۱- بهینه سازی سایت

بهینه سازی سایت ، یک فعالیت مداوم و روزمره است که نیاز به مهارت های حرفه ای به روز کردن سایت ، هوش و زیرکی و اشتیاق برای بهترین بودن دارد .

بهینه سازی سایت بهترین راه ، برای بالا رفتن در نتایج جستجوی **Google** و بهبود بازدید کنندگان سایت است .

۲- تبلیغات

اگر سایت شما به راستی خوب کار می کند ، به این معنی که شما اطلاعات مفید ، محصولات قابل فروش ، برنامه و یا خدمات اساسی ارائه می کنید . بهینه سازی مختصری در سایت ممکن است مستلزم پیمودن مسیر بسیار طولانی برای شما باشد . در این صورت ، اگر از قابلیت ها و توانایی های سایت خود اطمینان دارید و فکر می کنید که سایت شما ، کشش بینندگان زیاد را بدون بهینه سازی سایت دارا است ، در این صورت تبلیغات گزینه مناسبی است .

AdWords ، می تواند به شما کمک کند . این برنامه می تواند خریداران و مصرف کنندگان زیادی را به سایت شما سرازیر کند ، و شما به ازای هر فرد که وارد سایت شما خواهد شد ، هزینه تبلیغات را پرداخت خواهید کرد ، و البته اینکه ، این بازدید به خرید یا استفاده از سرویس ها منجر شود ، به سایت شما بستگی خواهد داشت .

۳- کسب درآمد

اگر شما محصولی را در سایت خود نمی فروشید ، و درآمدی از سایت خود ندارید و می خواهید به وسیله ی سایتتان کسب درآمد کنید ، در این صورت **AdSense** این امکان را به شما می دهد . **AdSense** راهی برای عضو شدن در برنامه ی مشارکتی تبلیغاتی **Google** است . در درآمد با **Google** شریک شوید . این سیستم قرار دادن آگهی ها در سایت و کسب درآمد به وسیله آن به کسب درآمد مجهول نیز معروف است . زیرا در واقع ، این شما نیستید که کسب درآمد می کنید ؛ بلکه این سایت شما است که کار میکند ، و درآمد دارد .

۱-۹- چگونه Google را محدود کنید ؟

حتی طراحان و مالکان سایت ها نیز گاهی مایلند **Google** را در برخی از بخش های تجارت خود محدود کنند .

صفحات کاملا شخصی که برای مشاهده دوستان ساخته می شود ، یا صفحات نیمه خصوصی که برای بازدید عده خاصی در نظر گرفته شده اند ، نباید در لیست جستجو وارد شود . همچنین صفحاتی از یک سایت کامل که در دست ساخت و بهینه سازی اند ، بهتر است در نتایج جستجو دیده نشود .

با یک فرآیند بسیار ساده ، می توانید آدرس تعدای از صفحات و یا کل سایت را از لیست جستجو حذف کنید . توضیحات ذیل ، به شما کمک می کند که **Google** را در خزش به سایتتان محدود کنید .

منحرف کردن خزنده

۱-۱۰- فایل Robot.txt

کلید منحرف کردن خزنده ی **Google** از سایتتان فایلی به نام **Robot.txt** است که با عنوان **"Robot Exclusion Protocol"** نیز شناخته می شود .

فایل **robot.txt** ، فایل بسیار کوچکی است که باید آن را در دایرکتوری ریشه سرور دامین تان قرار دهید . این فایل را می توانید به راحتی در **NotePad** یا هر ویرایشگر متنی دیگر ایجاد کنید و به فرمت **Ascii** درآورید . بهتر است از برنامه **Word** برای این کار استفاده نکنید ، ولی اگر مایلید از این نرم افزار جهت ساختن فایل **robot.txt** استفاده کنید ، به خاطر داشته باشید که فرمت فایل را حتما به صورت **txt** در آورید و سپس از انتقال آن ، به سرور به شکل باینری که پیش فرض برنامه های **FTP** است ، مطمئن شوید .

ما در سایت بهبازار نیز از فایل **robot.txt** استفاده کرده ایم که در ادامه همراه با توضیحاتی به آن خواهیم پرداخت .

برنامه **robot.txt** دو دستورالعمل دارد .

- **User Aqeyt** : این دستورالعمل ، مشخص می کند که خزنده موتور جستجو ، باید از دستورات فایل **robot.txt** پیروی کند (برنامه های خزنده به صورت پیش فرض در ابتدا فایل **robot.txt** را جستجو می کنند) .
- دستورات استثناگر **DisAllow** : دستوراتی هستند که نشان می دهند ، خزنده ی موتور جستجو ، به کدام قسمت های سایت نباید دسترسی پیدا کند .

اطلاعاتی که در این قسمت در اختیارتان قرار می گیرد ، نحوه ی اثر و چگونگی تولید فایل **robot.txt** است . در صورتی که مایل به کسب اطلاعات بیشتر درباره **robot** های خزنده ی جستجوگر هستید ، به آدرس زیر مراجه کنید :

www.robots.txt.org

سوالات معمولی که در این مورد پرسیده می شود ، نیز در این صفحه گردآوری شده است :

www.robotstxt.org/wc/faq.html

در ذیل کدهای مربوط به فایل robot.txt در سایت بهبازار را مشاهده می کنید :

```
User-agent: *
Disallow: /User/
Disallow: /Admin/
```

این دستوراتی است بسیار معمولی برای فایل **robot.txt** که به ترتیب خط اول به معنی این است که دستورات شامل تمام خزنده های و کاوشگرهای می شود .

و در خط دوم دایرکتوری به نام **User** با دستور **DisAllow** اجازه مشاهده را از خزنده می گیرد . و در خط بعد نیز مانند خط دوم اجازه مشاهده دایرکتوری **Admin** از خزنده گرفته می شود .

توجه کنید که صفحاتی اجازه دسترسی آن های توسط خزنده محدود شده است که در واقع تنها کاربران و مدیر سایت حق مشاهده آن را دارند و در آن ها اطلاعات حساس ذخیره شده است ، به همین دلیل نباید خزنده موتور جستجو آن ها را در دیتابیس خود ایندکس کند و به معرض نمایش همگان بگذارد !

در فایل **robot.txt** شما می توانید دستورات حرفه ای تر دیگری نیز به کار گیرد که توسط آن دستورات نام خزنده مورد نظر را تعیین کنید و تنها اجازه دسترسی آن خزنده بخصوص را سلب کنید و یا دستوری بنویسید که اجازه نمایش و جستجوی عکس های موجود در سایت را از خزنده بگیرید ، البته باید توجه کنید که این عکس ها شامل لوگو و آرم سایت شما نیز خواهد شد !

۱-۱۱- مستثنی کردن صفحات به وسیله تگ Meta

برخی مواقع حذف صفحات در لیست جستجو ، به روش تگ **Meta** کاری ساده تر از استفاده از فایل **robot.txt** است . در صورتی که کدهای **HTML** خود را به صورت دستی نوشته اید و از نرم افزارهایی مانند **Dreamweaver** و **FrontPage** استفاده نکرده اید ، استفاده از تگ **Meta** برای مستثنی کردن تعدادی از صفحات ساده تر است .

گرچه می توانید ، هم از **robot.txt** و هم از تگ **Meta** استفاده کنید . البته برای برخی از خزنده ها ، فرمان تگ **Meta** تعریف نشده است ، اما در مورد **Google** بدون نگرانی می توانید آن را به کار ببندید .

برای حذف صفحه ی بخصوص ، از نتایج جستجو ، خط فرمان زیر را در ابتدای کدهای **HTML** صفحه ی خود تایپ کنید :

```
<meta name="robots" content="noindex, nofollow">
```

به دو دستور **noindex** و **nofollow** دقت کنید . اولین دستور **Google** را از لیست کردن محتویات صفحه منع می کند و دومین دستور ، اجازه بررسی لینک های صفحه را به خزنده ی **Google** نمی دهد .

در صورتی که تنها مایلید محتویات صفحه در نتایج جستجو نیاید و لینک های صفحه پیگیری شوند دستور **nofollow** را حذف کنید .

همچنین می توانید به جای کلمه ی **robots** از **googlebot** استفاده کنید تا از این طریق تنها خزنده **Google** را از لیست کردن سایت تان مستثنی کنید .

۱-۱۲- پرهیز از حافظه Google Cache

در صورتی که ، در لینک **Cached** در نتایج جستجوی **Google** کلیک کنید ، مشاهده می کنید که صفحات بسیار سریعتر از معمول ظاهر خواهند شد .

در واقع ، این صفحات در حافظه **Google** ذخیره شده اند و مربوط به آخرین کاوش **Google** هستند که ممکن است در زمانی که شما از آن ها بازدید می کنید ، تغییر کرده باشند .

این امکان برای بازدیدکنندگان **Google** بسیار مناسب است . زیرا در صورتی که سایتی را مشاهده می کنند به سرعت به روز نمی شود ، امکان دسترسی سریع به سایت مربوطه را دارند .

این امکان صاحبان سایت ها را چندان خوشحال نمی کند . زیرا اولا **Google** با این کار به حریم قوانین مالکیت تجاوز کرده ، زیرا برای در اختیار داشتن نسخه ای از سایت اجازه نگرفته است (البته **Google** به درخواست صاحب سایت ، این نسخه را حذف می کند) .

ثانیا زمانی که یک طراحی سایت ، صفحه ای را تغییر می دهد ، مایل است که صفحه تغییر کند و در صورتی که خطایی در صفحه ی قدیمی وجود داشته باشد ، مایل نیست که بازدید کنندگان دوباره آن را مشاهده کنند .

با استفاده از دستور **Meta** زیر ، از ذخیره ی صفحه ی مورد نظرتان در حافظه می توانید جلوگیری کنید :

```
<meta name="googlebot" content="noarchive">
```

در صورتی که مایل نیستید ، صفحه ی شما به وسیله ی هیچکدام از خزنده ها ذخیره گردد ، به جای **GoogleBot** از **robots** استفاده کنید .

```
<meta name="robots" content="noarchive">
```

۱-۱۳- بهبود رتبه بندی در نتایج جستجو به وسیله ی برقراری لینک

نحوه ی رتبه نتایج جستجوی **Google** بسیار محرمانه ، اسرار آلود و برای کسانی که در روی **Web** تجارت می کنند ، بسیار مهم است و این موضوع ، به بزرگی یا کوچکی تجارت و یا شرکت ارتباطی ندارد . حتی شرکت ها و غول های بزرگی مانند **Amazon.com , Ebay.com** و **Yahoo!** نیز از رتبه بندی مناسب خوشحال خواهند شد .

تقریبا برای تمام شرکت ها مشاهده شدن در **Google** به معنی تجارت موفق تر و رتبه بندی مناسب به معنی مشاهده شدن بیشتر است .

رتبه بندی **Google** حاصل الگوریتم های خاص و پیچیده ی این شرکت است . گرچه فرمول این رتبه بندی منتشر نشده است ، اما با سعی و خطا و اطلاعیه های خود **Google** ، تا حدی در مورد این رتبه بندی اطلاعات به دست آمده است .

میزان اهمیت هر سایت نزد **Google** را رتبه بندی سایت مشخص می کند ، به این معنی که مهمترین سایت در نتایج جستجو ، ابتدا ظاهر می شود . نکته ی بعدی در اینجاست که ، کدامیک از صفحات سایت شما با کلمات کلیدی در ارتباط است و در نتایج جستجو خواهد آمد ، هر چه تعداد این صفحات هم بیشتر باشد ، شما در مقابل سایت های مشابه خود ، شانس بیشتر خواهید داشت .

Google همواره در تلاش است که عادل باقی بماند و موفق هم بوده است . نتایج جستجو بر اساس محبوبیت و شایستگی است . هر سایتی چه کوچک و چه بزرگ می تواند با برقراری لینک و بهینه سازی رتبه های مناسبی را اشغال کند .

لینک های ورودی و رتبه بندی

یکی از کلیدهای کسب موقعیت بهتر در نتایج جستجو ، برقراری لینک های ورودی از سایت های دیگر است . رتبه بندی سایت ، کاملا بر اساس الگوریتمی است که **Google** آن را محرمانه نگه داشته است . اما **Google** خود اعلام کرده است که ، تعداد لینک هایی که به سایت وارد می شوند ، مهمترین فاکتور مستقل در رتبه بندی سایت ها است . دو فاکتور اساسی در تلاش برای به دست آوردن رتبه ی بهتر ، برقراری لینک های ورودی به سایت و بهینه سازی سایت است .

هر دوی این فاکتورها ، به بازدیدکننده ی بیشتر و تجارت موفق تر می انجامد . در عمل هر چه راه های متعددی برای خزنده ی **Google** به سمت سایتتان فراهم شود ، رتبه بندی سایتتان در لیست نتایج جستجو بالاتر خواهد رفت .

نقش برقراری لینک ورودی

زمانی که یک سایت ، تعدادی لینک ورودی برقرار کرده است ، تا آن جا که توسط خزنده ی **Google** مورد کاوش قرار گیرد ، ممکن است خزنده ی **Google** در جستجوی خود تنها یکبار با این سایت برخورد کند . اما به نسبت ، زمانی که تعداد لینکها مثلا به ۱۰۰۰۰۰ لینک ورودی افزایش یابد ، خزنده به احتمال بسیار زیاد ، و به دفعات متعدد به این سایت برخورد خواهد کرد .

از نظر **Google** این سایت ، سایتی پربیننده است . بنابراین در نتایج جستجو مکان بالاتری را اشغال خواهد کرد . البته رتبه بندی صرفا بر اساس فاکتور یاد شده نیست و فاکتورهای بسیار زیادی در محاسبات گوگل وارد می شوند .

۱-۱۴- بهینه سازی سایت برای Google

بهینه سازی برای موتور جستجو یا **SEO** هم ساده و هم پیچیده است . ساده از آن جهت که قواعد پایه برای بهینه سازی بسیار روشن است و پیچیده از آن جهت که ، علاوه تنظیم تگ ها و صفحات ، شما باید بتوانید بازدید کننده ها را به آنچه می خواهید وادار کنید .

SEO ممکن است به طور مستقیم با درآمد زایی ارتباطی نداشته باشد . بهبود موقعیت سایت در نتایج جستجوی **Google** برای هر سایتی حتی سایت هایی که محصولی نمی فروشند ، مفید است .

بهینه سازی فریب دادن خزنده ی **Google** نیست . بهینه سازی یک بازی بدون بازنده است : بازدیدکننده ، سایت مورد قبول تری را مشاهده می کند ، سایت در نتایج جستجو رتبه ی مطلوب تری دارد و صاحب سایت ، از این موضوع منتفع خواهد شد .

بهینه سازی قبل از ساخت سایت

بهینه سازی را باید قبل از طراحی و ساخت سایت و حتی هنگام انتخاب موضوع آغاز کرد . بسیاری از طراحان ابتدا آمار دقیقی از کلمات کلیدی جستجو شده به دست می آورند ، سپس با توجه به نیاز خواست بازار ، اقدام به طراحی سایت و ثبت دامین مربوط می کنند .

به صورت نظری بهینه سازی سایت شامل گام های زیر است :

۱- به دست آوردن درک صحیح از سایت

درک صحیح از سایت به این معنی است که ، شما باید هدف از طراحی سایت را کاملا درک کرده باشید . یک سایت می تواند هداف متعددی داشته باشد ، اما فراموش نکنید که این اهداف باید کاملا وابسته و مربوط به هم باشند .

۲- مشخص کردن کلمات کلیدی

مشخص کردن کلمات کلیدی بسیار حیاتی است . کلمات کلیدی می توانند یک کلمه ی تنها یا مجموعه ای تشکیل دهنده ی یک عبارت باشند . البته در صورتی که ، از عبارتی کلیدی استفاده می کندی ، باید بسیار کوتاه باشد .

به عنوان مثال در سایت بهبازار از کلمات کلیدی زیر استفاده شده است :

کامپیوتر ، تجارت الکترونیک ، صنعت ، مقالات تجارت الکترونیک ، آموزش ، شبکه ، اینترنت ، نیازمندیها ، نیازمندیهای مازندران و گلستان ، کاریابی ، استخدام و ...

۳- ثبت دامین

دامین را با توجه به کلمات کلیدی خود انتخاب کنید . مانند سایت بهبازار که با توجه به اینکه این سایت یک نوع بازار می باشد که تجارت الکترونیک در آن انجام می شود نام دومین نیز بر همین اساس انتخاب شده است و سعی شده که این نام کوتاه نیز باشد .

نام سایت بهبازار (**www.BehBazar.com**) می باشد که ۸ کاراکتر است .

۴- طراحی سایت

در طراحی سایت ، به خزنده جستجوگر نیز توجه داشته باشید . که در قسمت های قبلی در مورد آن بحث شد .

۵- طراحی محتوا

طراحی محتوا یک پروسه ی مدوام در نگهداری سایت است که از زمان طراحی آغاز می شود . که البته ماهیت سایت بهبازار به گونه ای است که این سایت به خودی خود و توسط کاربران به روز خواهد شد و در واقع یک سایت کاملا پویاست .

۶- بهینه سازی محتوا به وسیله کلمات کلیدی

جای دادن کلمات هر صفحه در متن ، همان صفحه ی کار را برای بازدیدکنندگان و موتور جستجوی **Google** آسان می کند .

۷- استفاده از تگ در سایت

استفاده از تگ در سایت ، به معنی استفاده از کلمات کلیدی در تگ هایی است که خزنده ی **Google** با آن ها برخورد می کند .

تعداد بسیار کمی از طراحی سایت ، از ابتدا به این نکات توجه دارند . عده ای از طراحان سایت پس از کامل شدن سایت ، به فکر بهینه سازی می افتند . ولی همواره تصحیح اشتباهات ، سخت تر از اجتناب کردن از آن هاست . اما فارغ از آن که در کجای مسیر ایستاده اید ، همواره بهینه سازی در محتوا ، استفاده از تگ های مناسب ، طراحی کلمات کلیدی ، همواره به رونق گرفتن سایت می انجامد .

گروه طراحان بهبازار از ابتدا در فکر کسب بالاترین رتبه در نتایج جستجو در موتورهای جستجو بوده اند و به همین دلیل چندین فاکتور را برای این منظور در نظر گرفته اند که البته تعدادی از این فاکتورها در بالا به آن اشاره شد .

یکی از مواردی که برای این منظور در بهبازار گنجانده شده است قسمت است که مقالاتی در مورد تجارت الکترونیک که مربوط زمینه کاری سایت نیز می باشد هست . این مقالات که خود آن ها دارای مقادیر زیادی کلمات کلیدی در متن خود هستند هم به کاربران کمک مفید در برآورد نیاز آن ها می کند و هم آن ها را به سمت سایت ما از طریق جستجو روانه می کند . و چه بسا بسیاری از کاربران که از این طریق وارد سایت ما شده و پس از مشاهده سایت به مشتریان ما پیوسته اند !

فصل دوم

معرفی کنترل های ASP.NET

- کنترل **Label**
- کنترل **TextBox**

- کنترل های دکمه ای
- کنترل **Image**
- کنترل های رادیویی
- کنترل **DropDownList**
- کنترل **HyperLink**
- کنترل **FileUpload**

کلاس کنترل های وب در فضای نام **System.Web.UI.WebControls** قرار دارند . تمام کنترل های وب از کلاس پایه **WebControl** به ارث برده می شوند . این کلاس خواصی دارد که اغلب کنترل های وب آن ها را دارا می باشند . به عنوان مثال ، هر یک از کنترل های **Label** و **TextBox** خاصیتی به نام **BackColor** دراند که رنگ پس زمینه آن ها را تعیین می کند .

۲-۱- کنترل **Label**

این کنترل برای تعیین متن های ثابت به کار می رود . به عنوان مثال برای نمایش پیام ها یا عناوین از این کنترل استفاده می شود . یک خاصیت مهم این کنترل **Text** است که متن مربوط به آن کنترل را مشخص می کند و فاقد هر گونه متد و رویداد است .

۲-۲- کنترل **TextBox**

این کنترل برای دریافت اطلاعات به کار می رود . به عبارت دیگر ، متن قابل تغییر در این کنترل قرار می گیرد . این کنترل ۳ نوع است :

- ورود متن

- ورود کلمه عبور
- متن چند خطی

نوع اول معمولا برای دریافت ورودی های یک خطی به کار می رود . نوع دوم برای دریافت کلمه عبور به کار می آید . ویژگی کلمه عبور این است که هنگام ورود آن ، نباید به نمایش درآید ، بلکه کاراکتر خاصی مثل * یا . به جای کاراکتر ورودی قرار می گیرد . نوع سوم برای ورود متن های طولانی و چند خطی به کار می رود . خواص **Columns** و **Rows** برای تعیین ابعاد کنترل **TextBox** برحسب کاراکتر به کار می روند . اما با استفاده از خواص **Width** و **Height** مربوط به کلاس پایه **WebControl** نیز می توان ابعاد را برحسب واحدهایی مثل **Pixel** تعیین کرد .

۲-۳- کنترل های دکمه ای

تعدادی از کنترل های در **ASP.NET** وجود دراند که به صورت دکمه عمل می کنند . این کنترل ها عبارتند از :

- **Button**
- **ImageButton**
- **LinkButton**

هر چند که هر یک از این دکمه ها کار خاصی را انجام می دهند ، ولی اثر همه ی آن ها تحویل فرم به سرور است .

۲-۳-۱- کنترل Button

با کلیک این کنترل ، فرم به سرور تحویل داده می شود . کنترل **Button** هم می تواند به عنوان دکمه **Submit** برای تحویل فرم عمل کند و عمل می تواند به عنوان یک دکمه فرمان (**Command**) عمل کند . در حالت اول از رویداد **Click** و در حالت دوم از رویداد **Command** استفاده می گردد که خاصیت **CommandName** نامی برای دکمه انتخاب می شود . به این ترتیب می توان در برنامه از چند دکمه استفاده کرد و با استفاده از رویداد **Command** تشخص داد که کدام دکمه کلیک شده است .

۲-۳-۲- کنترل ImageButton

این کنترل تا حدی شبیه کنترل **Button** است ، با این تفاوت که می تواند تصویری را نمایش دهد و با کلیک کردن بر روی تصویر ، به رویدادهای آن پاسخ داد .

۲-۳-۳- کنترل LinkButton

عملکرد این کنترل مثل عملکرد کنترل **Button** است ، با این تفاوت که به یک پیوند تبدیل می شود . وقتی بر روی این پیوند کلیک شود ، تمام اطلاعات موجود در فرمی که این کنترل بر روی آن قرار دارد به سرور منتقل می شود . خواص ، رویدادها و متدهای این کنترل دقیقا مثل کنترل **Button** است ، با این تفاوت که متنی که در خاصیت **Text** این کنترل وارد می شود ، به صورت یک پیوند ظاهر می شود .

۲-۴- کنترل Image

این کنترل می تواند تصویری را به صفحه وب بیاورد . تفاوت این کنترل با کنترل **ImageButton** این است که در کنترل **Image** نمی توان کلیک کردن ماوس را تشخیص داد ، ولی همان طور که دیدید ، در کنترل **ImageButton** با استفاده از رویدادهای **Click** و **Command** می توان کلیک کردن ماوس را بر روی آن تشخیص داد .

۲-۵- کنترل های رادیویی

دو کنترل رادیویی به نام های **RadioButton** و **RadioButtonList** برای ایجاد گزینه هایی به کار می روند که فقط یکی از آن ها قابل انتخاب است .

۲-۵-۱- کنترل RadioButton

این کنترل یک دکمه رادیویی را در صفحه ایجاد می کند . بدیهی است که تنها وجود یک دکمه رادیویی ، مشکلی را حل نمی کند . بلکه مجموعه ای از دکمه های رادیویی باید ایجاد شوند تا هر کدام یک گزینه را تعریف کند . بنابراین هر مجموعه از دکمه های رادیویی از طریق خاصیت **GroupName** در یک دسته قرا می گیرند که در هر دسته فقط یک گزینه قابل انتخاب است . بنابراین ، ممکن است چند دسته از دکمه های رادیوی در صفحه داشته باشیم که هر دسته دارای یک نام باشند و در هر دسته یک گزینه قابل انتخاب است .

بدیهی است که یکی از نکات مهم در بکارگیری دکمه های رادیویی ، تشخیص گزینه انتخاب است . این کار به دو روش انجام می گیرد :

- استفاده از خاصیت **Checked** هر یک از دکمه های رادیویی .
- استفاده از رویداد **CheckedChanged** .

اگر بخواهید پس از تغییر وضعیت هر گزینه ، فرم به سرور تحویل داده شود ، باید خاصیت **AutoPostBack** تمام گزینه ها را **True** کنید .

۲-۵- کنترل RadioButtonList

این کنترل می تواند حاوی چند دکمه رادیویی باشد . به عبارت دیگر ، این کنترل شامل چند کنترل **RadioButton** است که به طور خودکار در یک گروه قرار می گیرند و فقط یکی از آن ها قابل انتخاب است . وقتی چند دکمه رادیویی بخواهند در ی گروه قرار گیرند ، باید خاصیت **GroupName** آن ها یک مقدار داشته باشد . اما وقتی از **RadioButtonList** استفاده می شود ، دکمه های رادیویی به طور خودکار در یک گروه قرار خواهند گرفت .

نکات مهم مربوط به کنترل **RadioButtonList** عبارتند از :

- اضافه کردن دکمه های رادیویی به لیست .
- یافتن دکمه رادیویی انتخاب شده .
- تعیین ظاهر لیست دکمه های رادیویی .

۲-۶- کنترل DropDownList

این کنترل نوعی لیست بازشونده را نشان می دهد که در آن واحد فقط یک گزینه آن نمایش داده می شود و با کلیک کردن فلش موجود در آن می توان تمام گزینه های لیست را مشاهده کرد . فقط یک گزینه از آن نیز قابل انتخاب است . امتیاز این لیست این است که فضای اندکی را از صفحه نمایش اشغال می کند .

۲-۷- کنترل HyperLink

برای ایجاد پیوند از کنترل **HyperLink** استفاده می شود . این کنترل فاقد هرگونه رویداد و متد است . پیوندی که توسط این کنترل ایجاد می شود ، می تواند به صورت متن یا تصویری باشد .

حتما در بسیاری از سایت دیده اید که وقتی پیوندی را کلیک می کنید ، صفحه جدید می تواند در پنجره فعلی باز شود یا می تواند در پنجره جدیدی قرار گیرد . اگر بخواهید در پنجره فعلی باز شود ، خاصیت **Target** را برابر **_self** و اگر بخواهید در پنجره جدیدی قرار گیرد ، خاصیت **Target** را برابر با **_blank** قرار دهید .

۲-۸- کنترل FileUpload

یکی از نکات مهم این است که سایت شما امکان پذیرش فایل های دیگران را داشته باشد . به عنوان مثال ، اگر بخواهید در سایت خود آلبومی از عکس های خاص را ایجاد کنید ، باید به کاربران اجازه دهید فایل های تصویر را به سایت شما ارسال کنند . البته ، این کار می تواند برای سایت شما خطرناک باشد ، زیرا ممکن است افرادی از این وضعیت سوء استفاده کنند و فایل ها و برنامه های خطرناکی را به سایت شما ارسال کنند . البته اگر پروژه را به خوبی پیکربندی کنید ، می توانید ضمن پذیرش فایل های کاربران ، امنیت را نیز در سیستم خود فراهم کنید .

کنترل **FileUpload** این امکان را برای شما فراهم می کند که بتوانید فایل هایی را از کاربران سایت خود دریافت کنید .

فصل سوم

کنترل های اعتبارسنجی (Validation Controls)

- کنترل **RequireFiledValidator**
- کنترل **RegularExpression**
- کنترل **CompareValidator**
- کنترل **RangValidator**
- کنترل **ValidationSummary**
- کنترل **CustomValidator**

۳-۱- کنترل های اعتبارسنجی

در بعضی از صفحات وب که در سایت بهبازار طراحی شده است کنترل هایی برای دریافت اطلاعاتی مثل نام کاربری ، کلمه عبور و یا ایمیل وجود دارند . در بسیاری موارد لازم است محتویات این کنترل ها قبل از ارسال به سرور ارزیابی شوند . به عنوان مثال مشخص شود که فیلدی حاوی مقدار است یا خیر ، آیا مقدار فیلدی در شرایط خاص صدق می کند یا خیر ، آیا فیلدی با فیلد دیگری یا با یک مقدار ثابت برابر است یا خیر . بررسی این موارد و موارد مشابه آن را اعتبارسنجی می گویند . اعتبارسنجی داده ها در طرف مشتری (قبل از تحویل به سرور) از اهمیت ویژه ای برخوردار است . کنترل های اعتبارسنجی را به صورتی ساده در این بخش مورد بررسی قرار خواهیم داد .

۳-۲- فعال و غیر فعال کردن اعتبارسنجی سمت مشتری

کنترل های اعتبارسنجی از کتابخانه اسکریپتی **JavaScript** استفاده می کنند که هنگام نصب **ASP.NET** ، بر روی سرور نصب می گردد . این کتابخانه در فایل **WebUIValidation.js** قرار دارد . همه ی کنترل های اعتبارسنجی خاصیتی به نام **EnableClientScript** دارند که اگر مقدارش

True باشد ، آن کنترل اعتبارسنجی عمل خواهد کرد ، ولی اگر مقدارش **False** باشد ، آن کنترل عمل نخواهد کرد .

۳-۳- فرآیند اعتبارسنجی

برای اعتبارسنجی فیلدها از طریق کنترل های اعتبارسنجی مراحل زیر انجام می شود :

۱. صفحه ای در اختیار کاربر قرار می گیرد که فیلدهای ورودی را پر می کند .
۲. پس از پر کردن فرم ، دکمه ای را کلیک می کند تا صفحه به سرور تحویل داده شود .
۳. هر کنترل دکمه ای ، خاصیتی به نام **CauseValidation** دارد .

a. اگر این خاصیت برابر با **False** باشد ، **ASP.NET** کنترل ها را اعتبارسنجی نمی کند و صفحه را به سرور تحویل می دهد .
b. اگر این خاصیت **True** باشد (درحالت پیش فرض) ، وقتی کاربر دکمه تحویل فرم را کلیک کرد ، فرم به طور خودکار اعتبارسنجی می شود .

۳-۴- کنترل RequiredFieldValidator

این کنترل می تواند تعیین کند کنترلی از فرم ، مثل **TextBox** دارای مقدار است یا خیر . به عنوان مثال ، فرض کنید فرم دارای یک کنترل **TextBox** است و قبل از تحویل فرم به سرور ، حتما باید مقدار داشته باشد . اگر کاربر بدون وارد کردن مقداری در این فرم ، دکمه **Submit** را کلیک کند ، این کنترل اعتبارسنجی ، پیام خطایی را صادر می نماید .

توجه کنید که برای هر کنترلی که می خواهید اعتبار آن را بررسی کنید ، باید یک کنترل اعتبارسنجی به آن نسبت دهید . اگر پس از اعتبارسنجی ، بخواهیم مشخص کنیم که آیا نتیجه اعتبارسنجی هر کنترل مثبت است یا خیر ، باید خاصیت **IsValid** همان کنترل اعتبارسنجی را بررسی کنیم . مقدار **True** به معنای مثبت بودن و مقدار False به معنای منفی بودن اعتبارسنجی است . نکته مهم این است که اگر اعتبارسنجی تمام کنترل ها مثبت باشد ، خاصیت **Page.IsValid** (خاصیت **IsValid** مربوط به صفحه) **True** خواهد بود .

خاصیت **Display** برای تعیین چگونگی نمایش پیام متن به کار می رود . اگر مقدارش **Static** باشد ، وضعیت فعلی صفحه نمایش برای چاپ پیام حفظ می شود . مقدار **Dynamic** موجب جابه جایی کنترل های صفحه و نمایش مناسب پیام خطا می شود.

خاصیت **InitialValue** نیز نیاز به توضیح دارد . به عنوان مثال ، ممکن است مقدار اولیه ای را دریک فیلد **TextBox** قرار دهیم و با استفاده از کنترل اعتبارسنجی انتظار داشته باشیم که کاربر مقدار متفاوتی را وارد کند ، وگرنه اعتبارسنجی مثبت نخواهد بود . این مقدار اولیه در فیلد **InitalValue** قرار خواهد گرفت .

برای ذکر مثالی در مورد این کنترل صفحه ی ورود به بخش مدیریت در سایت بهبازار را در نظر بگیرید . در این صفحه دو **TextBox** برای وارد کردن نام کاربری و کلمه عبور وجود دارد که در هر دو مورد از کنترل **RequiredFieldValidator** استفاده شده است چرا که برای وارد شدن به بخش مدیریت به هر دوی این ها احتیاج می باشد و باید حتما کاربر آن ها را وارد کند .

شکل ۳-۱ - **RequiredFieldValidator**

همانطور که در شکل نیز ملاحظه می کنید در صورت خالی بودن دو مورد ایمیل و کلمه عبور پیغام خطایی به کاربر نمایش داده می شود که از وی تقاضا می شود این مقادیر را وارد کند . با اضافه کردن این کنترل ها به صفحه مورد نظر کدهای **HTML** زیر به صورت اتوماتیک اضافه می شود :

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="Email" ErrorMessage="شود وارد"
Font-Names="Tahoma" Font-Size="7pt">
</asp:RequiredFieldValidator>
```

۳-۵- کنترل RegularExpressionValidator

با استفاده از این کنترل اعتبارسنجی می توانیم مقدار موجود در یک فیلد را با یک عبارت منظم مقایسه کنیم . این کنترل برای اعتبارسنجی آدرس های پست الکترونیکی ، نام کاربر و کلمه عبور ، آدرس های وب

، شماره تلفن ، کدپستی و طول اطلاعات وارد شده به کار می رود . بسیاری از خواص و متدهای این کنترل با کنترل **RequiredFieldValidator** مشترک است .

عبارت منظم

عبارات منظم ، ابزار پیشرفته ای برای تطبیق الگوها است . در ویژوال استودیو نت ، برنامه نویسان می توانند عملیات جست و جو و جایگزینی را از طریق عبارات منظم انجام دهند . کل یک زبان را می توان یک عبارت منظم دانست . تمام عبارات منظم از دو نوع کاراکتر تشکیل شده اند که عبارت اند از لیترال ها و متاکاراکترها . لیترال ها کاراکتر تعریف شده ی خاصی اند . به عنوان مثال کاراکتری مثل ***"x"*** را جست و جو می کنید ، دقیقا همان کاراکتر را می یابید . متا کاراکترها مثل علائم × و ؟ در سیستم عامل **DOS**هستند که هر کاراکتری به جای آن قرار می گیرد . به عنوان مثال ، منظور از ×.× تمام فایل های موجود در دیسک است . نقطه یک لیترال و × متاکاراکتر است . عبارات منظم دیگر نیز وجود دارند . به عنوان مثال **\s** نشان دهنده فضای خالی (مثل **blank** و **tab**) است . **\d** ارقام را نشان می دهد .

به عنوان مثالی از کنترل **RegularExpressionValidator** مثال قبل را در نظر بگیرید . در مثال قبل از کاربر به جای نام کاربری ، ایمیل او دریافت می شود . برای اعتبارسنجی ایمیل و اطمینان از اینکه کاربر به درستی ایمیل خود را وارد کرده است از کنترل **RegularExpressionValidator** استفاده می کنیم و کدهای **HTML** زیر را خواهیم داشت .

```
<asp:RegularExpressionValidator
ID="RegularExpressionValidator2" runat="server"
ControlToValidate="Email"
ErrorMessage="اشتباه ایمیل" Font-Names="Tahoma" Font-Size="7pt"
ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)*" ValidationGroup="Login">اشتباه ایمیل
</asp:RegularExpressionValidator>
```

۳-۶- کنترل CompareValidator

این کنترل می تواند محتویات یک فیلد را با مقدار یک فیلد دیگر یا با یک مقدار ثابت مقایسه کند . علاوه بر این برای تعیین نوع داده ی وارد شده نیز به کار می رود . به عنوان مثال اگر بخواهیم مشخص کنیم که آیا کاربر تاریخ ، شماره یا مبلغ معتبری را وارد کرد ، از این کنترل استفاده می کنیم . یکی دیگر از مواردی که در آن از این کنترل می توان استفاده کرد مقایسه تکرار کلمه عبور با کلمه عبور می باشد .

برای اینکه نمونه ای از این مورد را ببینید ، صفحه زیر را که صفحه ثبت نام کاربران در سایت بهبازار می باشد در نظر بگیرید :

شکل ۳-۲- **CompareValidator**

همانطور که در این صفحه ملاحظه می کنید در هنگام دریافت اطلاعات از کاربر دو گزینه به نام های کلمه عبور و تکرار کلمه عبور وجود دارد . تکرار کلمه عبور برای تایید کلمه عبور وارد شده توسط کاربر می باشد تا کاربر از کلمه عبور وارد شده خود از این طریق اطمینان یابد . در اینجا برای **TextBox** دوم (تکرار کلمه عبور) از کنترل **CompareValidator** استفاده می کنیم تا آن را با **TextBox** کلمه عبور مقایسه کنیم و در صورت یکی بودن دو مقدار فرم برای ارسال به سرور تایید می شود .

کنترل **CompareValidator** دارای دو خصوصیت **ControlToCompare** و **ControlToValid** می باشد که خصوصیت اول نشان دهنده کنترلی است که می خواهید مقایسه با آن صورت گیرد و خصوصیت دوم نشان دهنده کنترل مورد بررسی است . کدهای **HTML** ایجاد شده توسط این کنترل را در زیر مشاهده می کنید :

```
<asp:CompareValidato  ID="CompareValidator1" runat="server"
ControlToCompare="Pass" ControlToValidate="RepeatPass"
 ErrorMessage="نیست صحیح تکرار" Font-Names="Tahoma" Font-
Size="7pt">نیست صحیح تکرار
</asp:CompareValidator>
```

۳-۷- کنترل RangeValidator

این کنترل بررسی می کند که آیا مقدار یک فیلد (کنترل) ، در محدوده خاصی قرار دارد یا خیر . مقدار مورد مقایسه ممکن است عددی ، تاریخ ، رشته ای یا کاراکتری باشد . بسیاری از خواص و متدهای این کنترل شبیه کنترل **RequiredFieldValidator** است . به عنوان مثالی برای استفاده از این کنترل می توان به موردی اشاره کرد که می خواهیم در هنگام دریافت سن افراد ، سن آن ها در محدوده معینی مثلا ***15 – 30*** سال باشد .

۳-۸- کنترل ValidationSummary

فرض کنید فرمی با ۵۰ فیلد (کنترل) داشته باشید و بخواهید اعتبار هر کدام از این فیلدها را با استفاده از کنترل های اعتبارسنجی که تا کنون بررسی شدند ، بسنجید . چاپ پیام های خطای مربوط به هر کدام از این فیلدها در نقاط مختلف صفحه نمایش ، نه تنها صفحه نمایش را از زیبایی خارج می کند ، بلکه مشاهده پیام مربوط به فیلد خاص ، مستلزم حرکت در صفحه نمایش است .

با استفاده از کنترل **ValidationSummary** می توان خطاهای اعتبارسنجی را در نقطه خاصی از صفحه نمایش ، به کاربر نشان داد . حتی می توان آن ها را در کادر جداگانه در صفحه نمایش چاپ کرد . توجه داشته باشید که خاصیت **Text** و **ErrorMessage** کنترل اعتبارسنجی برای نمایش پیا در کنترل **ValidationSummary** به کار می رود .

کنترل **ValidationSummary** ، خطاها را به صورت یک لیست علامت دار (**Bulleted**) نمایش می دهد . می توانیم چگونگی نمایش پیام های خطا را عوض کنیم ، به طوری که به صورت لیست بدون علامت یا به صورت پاراگراف نمایش دهیم . این کار با خاصیت **DisplayMode** انجام می گیرد .
علاوه بر این ، برای اینکه پیام های خطا را در کادر جداگانه ای در صفحه نمایش نشان دهید ، خاصیت **ShowMessageBox** را **True** کنید . اگر می خواهیم پیام ها فقط در کادر جداگانه ظاهر شوند و نه درخود صفحه ، مقدار خاصیت **ShowSummary** را **False** تعیین کنید .

۳-۹- کنترل CustomValidation

کنترل های اعتبارسنجی که تا کنون مورد بررسی قرار گرفتند ، اعتبارسنجی را به شکل معینی انجام می دهند . اما گاهی ممکن است به اعتبار سنجی خاصی نیاز داشته باشید که اعتبارسنجی های آماده ، پاسخگوی نیاز شما نباشد . به عنوان مثال ، ممکن است بخواهید یک فیلد عددی را طوری دریافت کنید که همواره مضربی از ۳ باشد . برای این نوع اعتبارسنجی ، کنترل های اعتبارسنجی موجود مناسب نیستند . برای این منظور از نترل **CustomValidation** استفاده می شود . بسیاری از خواص و متدهای این کنترل همانند سایر کنترل های اعتبارسنجی است . اعتبارسنجی از طریق این کنترل را اعتبارسنجی سفارشی گویند .
یکی از مواردی که اعتبارسنجی سفارشی می تواند مورد استفاده قرار گیرد ، ثبت نام کاربران در یک سایت است . معمولا سایت ها برای پر کردن یک فرم ثبت نام ، از کاربر می خواهند که یک نام کاربری یا آدرس ایمیل را وارد کند . برای تشخیص اینکه نام کاربری یا ایمیل منحصر بفرد است یا خیر ، باید در بانک اطلاعاتی وب سایت جستجو شود . این کار را می توان با استفاده از کنترل **CustomValidate** انجام داد . به طوری که می توانیم زیر برنامه ای بنویسیم که اعمال مورد نظر ما را انجام دهد . این اعتبارسنجی می تواند در طرف سرور یا طرف مشتری انجام گیرد .

۳-۱۰- جلوگیری از اعتبارسنجی

معمولا در هر فرم علاوه بر دکمه **Submit** که فرم را به سرور تحویل می دهد ، دکمه دیگری به نام **Cancel** وجود دارد که کاربر می تواند با کلیک کردن بر روی آن ، از فرم صرف نظر کرده به صفحه دیگری برود . این نوع دکمه در فرمی که حاوی کنترل های اعتبارسنجی است ، با فرم های دیگر متفاوت است . مسئله این است که اسکریپت های اعتبارسنجی طرف مشتری می توانند مانع از اجرای زیربرنامه هایی شوند که به دکمه **Cancel** اختصاص دارند .
برای حل این مسئله ، از خاصیت **CauseValidation** مربوط به دکمه **Button** ، **LinkButton** و **ImageButton** استفاده می شود . این خاصیت می تواند اعتبارسنجی را فعال یا غیرفعال کند .

فصل چهارم

طراحی صفحات سایت

- صفحات ایستا (**Static**)
- صفحات پویا (**Dynamic**)
- صفحات اصلی و محتوی (**MasterPage**)
- استفاده از **CSS** در طراحی سایت

۴-۱- طراحی صفحات سایت

وقتی طرحی وب سایت را شروع می کنید ، در ابتدا بهتر است کارهایی که می خواهید وب سایت شما بتواند انجام دهد را مشخص کنید . مثلا وب سایت بهبازار ، که سایتی تجاری می باشد و در خود آگهی ها را جای می دهد ، باید بتواند آگهی ها را به صورت طبقه بندی به نمایش بگذارد . همچنین باید بینندگان بتوانند در سایت ثبت نام به عمل آورند و کاربر سایت شوند و از این طریق آگهی های مورد نظر خود را در سایت به ثبت برسانند . همچنین باید برای راحتی کار بینندگان آگهی ها قسمتی را برای جستجوی آگهی ها در نظر بگیرید ، تا بیننده برای پیدا کردن آگهی مورد نظر خود وقت زیادی را صرف نکند .

بعد از مشخص کردن کارهایی که می خواهید انجام دهید ، باید صفحات سایت را مشخص کنید . به طور مثال برای ثبت نام ، صفحه ی ثبت نام کاربر جدید را در نظر می گیرید . و یا صفحه ای را برای نمایش جدیدترین آگهی ها و صفحه ای دیگر برای جستجوی آگهی مد نظر قرار می دهید . پس از مشخص کردن صفحات مورد نظر باید نوع صفحات خود را مشخص کنید . ما در کل دو نوع صفحه خواهیم داشت :

۱. صفحات ایستا .

۲. صفحات پویا .

۴-۲- صفحات ایستا

این صفحات ، صفحاتی را شامل می شوند که در آن ها هیچ تراکنشی صورت نمی گیرد و با هیچ پایگاه داده ای در ارتباط نخواهیم بود و تنها یک متن یا تصویر ثابت را بدون هیچ جابجایی و تغییر به نمایش درمی آوریم . این صفحات تنها جهت اطلاع رسانی می باشند و عموما صفحاتی را شامل می شوند که خیلی به ندرت تغییر می کنند . به طور مثال در سایت بهبازار صفحات درباره ما و همچنین صفحه ی تعرفه آگهی ها از این نوع صحفات هستند . ما در این صفحات تنها اطلاعاتی را به کاربران و بینندگان می دهیم که تقریبا تا مدت ها بدون تغییر هستند .

در صفحات ایستا از بانک های اطلاعاتی به هیچ وجه استفاده نمی شود . و متون کاملا ثابت و بدون تغییر می باشند .

۴-۳- صفحات پویا

صفحات پویا به صفحاتی اطلاق می شود که در تعامل با بینندگان می باشند و یا محتویات آن ها در هر بار لود شدن تغییر می کند . این صفحات عموما با دیتابیس در ارتباط می باشند و اطلاعاتی را ذخیره می کنند و یا اینکه اطلاعات ذخیره شده قبلی را به نمایش می گذارند .

به عنوان مثالی از اینگونه صفحات می توان صفحات تماس با ما و یا آگهی های جدید را در سایت بهبازار از این نوع صفحات برشمرد .

۴-۴- صفحات اصلی و محتوی

هر سایتی از داشتن ظاهری زیبا و یکنواخت سود می برد . در اینترنت بندرت می توان سایتی را یافت که از داشتن چینش عمومی یک سایت منحرف شده باشد . چینش عمومی معمولا شامل موارد زیر است :

۱. یک سرآمد (**Header**) مشترک و سیستم منو برای کل سایت .
۲. نواری در سمت چپ صفحه که شامل گزینه های ناوبری سایت است .
۳. پاورقی (**Footer**) که اطلاعات کپی رایت را فراهم می کند و همچنین منوی دیگری برای تماس با رئیس و مدیر سایت .

این عناصر با اینکه ویژگی های ضروری را فراهم نمی کنند ، بر روی همه صفحات قرار می گیرند . ولی ظاهر و چینش یکنواخت این عناصر به کاربر اطمینان می دهد که هنوز در همان سایت هستند . اگر چه این گونه ظاهر را می توان با فایل های **HTML** ایجاد کرد ، ولی **ASP.NET 2.0** ابزار قدرتمندتری همچون صفحات اصلی (**Master**) و محتوی (**Content**) را فراهم نموده است .

صفحه **Master** چینش کلی مورد استفاده در کلیه صفحات سایت را تعریف می کند . صفحه **Master** بعنوان پدر کل سایت ، چینش سایر صفحات را کنترل کرده و مشخص می کند که سرآمد هر صفحه به چ بزرگی باشد ، ابزار ناوبری سایت در کجای صحفات قرار گیرند ، و متنی که در پاورقی صفحات نشان داده می شود چه باشد . صفحه **Master** حاوی برخی از محتویاتی که در صفحات سایت وجود دارند ، می باشد . بنابراین استاندارد متن پاورقی کپی رایت را می توان در این صفحه قرار داد و همچنین آرم (**Logo**) اصلی سایت را نیز می توان در بالای این صفحه منظور کرد . پس از تعریف ویژگی های استاندارد برای صفحه **Master** ، می توانید تعدادی جانگهدار (**Place Holder**) به آن اضافه کنید . جانگهدارها مناطقی بر

روی صفحه هستند که جاهایی که محتویاتشان از صفحه ای به صفحه دیگر تغییر می کند را تعریف می کنند .

صفحه محتوی (**content**) صفحه ای مبتنی بر صفحه **Master** است . این صفحه جائیست که محتویات هر صفحه سایت را اضافه می کنید و این محتویات از صفحه ای به صفحه دیگر متفاوت است . صفحه محتوی شامل متن ، **HTML** و کنترل هایی در برچسبهای **<asp:content>** می باشد . وقتی صفحه محتوی مورد درخواست واقع می شود ، محتویاتش با یک کپی از صفحه **Master** ترکیب می شود بطوریکه محتویات خاص تعریف شده در صفحه محتوی در جانگهدارهای مخصوص موجود در صفحه **Master** قرار می گیرند . سپس کل بسته برای مرورگر ارسال می شود . مطابق شکل زیر :

شکل ۴-۱- صفحات اصلی و متحوی

۴-۵- استفاده از CSS در طراحی صفحات سایت

برگه های سبک آبشاری (CSS – Cascading Style Sheet)

مفهوم برگه های سبک آبشاری چند سالی است که مورد استفاده قرار می گیرد (در ابتدا در دسامبر سال ۱۹۹۶ توسط W3C پیشنهاد شد) ، طراحی خوب هر برنامه کاربردی تحت وب ، بستگی به این دارد که **CSS** آن تا چه اندازه خوب تعریف شده باشد .با استفاده از برگه سبک ، می توانید هر نوع عنصری که می خواهید بر روی صفحه ظاهر شود را تعریف کنید . همچنین قادر هستید تعاریفی را برای سبک های خاصی که می توانید آن را انتخاب و بر روی عناصر خاصی از صفحه اعمال نمایید ایجاد کنید . مثلا می توانید مشخص کنید ، هر نمونه از برچسب ** ** باید شامل متن با رنگ **navy** باشد . و یا اینکه می توانید یک کلاس (class) سبک تعریف کنید و آن را برای هر یک از برچسب های یا سایر عناصر موجود بر روی صفحه بکار ببرید . در زیر ، برگه سب آبشاری سایت بهبازار را مشاهده می کنید :

```
body
{

}
p {
     font-family:Tahoma;
```

```
	font-size: 8pt;
	text-align: center;
}
a:link {

	text-decoration: underline;
	text-decoration: none;
	color:#0000a8;
}
a:visited{

	text-decoration: none;
	color:#660033;
}
a:hover {
	text-decoration: none;
	color: #ff0000;
}
a:active {
	color: #cc0000;
	text-decoration: none;
}
```

در کدهای بالا که در برگه سبک بهبازار قرار دارند برای تگ های P که نشان دهنده پاراگراف ها در صفحه می باشند و همچنین برای تگ های a که لینک های صفحه را مشخص می کنند کد نویسی شده است . در مورد دستورات و خصوصیات CSS در ضمیمه راهنمای سریع CSS همگی آورده شده اند و اما در مورد باقی موارد .

a:link - این قسمت خصوصیات لینک های صفحه را مقدار دهی می کند .

a:visited - این بخش مشخص کننده لینک هایی که توسط بیننده وب سایت دیده شده است و حداقل یک بار بر روی آن ها کلیک شده است . و خصوصیات لینک های بازدید شده را مشخص و مقدار دهی می کند .

a:hover - این حالت برای موقعی است که نشانگر ماوس روی لینک ها قرار می گیرد .

a:active - این حالت هنگامی اتفاق می افتد که نشانگر ماوس روی لینک قرار دارد و کلیک چپ ماوس نگه داشته شده است .

استفاده از کدهای CSS در صفحه به دو صورت مورد استفاده قرار می گیرد . یکی به صورت یک فایل مجزا می باشد که کدهای CSS را در فایلی با پسوند .css ذخیره کرده و سپس در صفحه مورد نظر آن را

فراخوانی می کنیم . همان کاری که ما در سایت بهبازار انجام داده ایم . برای فراخوانی این فایل از کد زیر در صفحه استفاده می گردد .

```
<link href="cssName.css" rel="stylesheet" type="text/css" />
```

که این کار بسیار ساختیافته تر ، ساده تر است و همچنین دراین روش از کدهای کمتری استفاده خواهد و از یک فایل CSS می توان در چندین صفحه استفاده کرد .

و اما روش دوم روشی است که در خود تگ های صفحه از دستور style استفاده کنیم . که این روش باعث زیاد شدن کدهای صفحه نسبت به روش قبل خواهد شد . برای مثال از این روش کدهای زیر را در نظر بگیرید :

```
<span style="color: #212142">جديد هاي آگهي</span></a>
<span style="color: #212142"> | 
</span>
```

همانطور که ملاحظه می کنید از دستور **style** در تگ **span** استفاده شده است و خصوصیت **color** مقدار دهی شده است .

برای اعمال یکی سبک بر روی یک عنصر لازم نیست کاری را بر روی خود عنصر انجام دهید تا زمانی که صفحه شما می داند که از کجا باید اطلاعات مربوط به سبک را پیدا کند ، سبک بطور خودکار اعمال می گردد . اما برای مشخص کردن کلاس خاصی که باید برای سبک بندی به کار روی ، از مشخصه **class** استفاده می شود که در این مورد نام کلاس در تگ مربوطه ذکر خواهد شد .

فصل پنجم

فایل های استاندارد برای برنامه های کاربردی ASP.NET 2.0

- فایل **Web.config**
- فایل **Global.asax**

۵-۱- فایل های استاندارد برای برنامه های کاربردی ASP.NET 2.0

ASP.NET 2.0 از دو نوع فایل استفاده می کند که در کلیه سایت های **ASP.NET** مشترک بوده و برای نگهداری اطلاعات پیکربندی و کدهای بکار رفته در کل سایت می باشند که به ترتیب عبارتند از : فایل **Web.config** و فایل **Global.asax** .

- فایل **Web.config** حاوی تنظیمات پیکربندی سایت است . مثلا جهت هر مشکلی که در سایت برای کاربران بوجود می آید ، می توان استاندارد دلخواهی را برای صفحه خطا (**error page**) مشخص نمود تا برای کاربران نمایش داده شود .
- فایل **Global.asax** حاوی کدهایی است که کلیه رویدادهای اتفاق افتاده توسط هر یک از صفحات سایت را اداره می کند . مثل کدی که هر دفعه ، کاربری برای نخستین بار به سایت وارد می شود ، اجرا می گردد.

۵-۲- Web.config فایلی که تنظیمات کل سایت را نگهداری می کند

فایل **Web.config** مقادیری که در کل سایت بکار می رود را ذخیره می کند . این فایل دارای ساختار **XML** بوده و در ریشه سایت قرار دارد . گره ها ، اطلاعات موجود در سه ناحیه اصلی را نگهداری می کنند :

- تنظیمات برنامه کاربردی برای پذیرش ویژگی مورد استفاده در حین پیاده سازی .
- رشته های اتصال (**connection strings**) که حاوی مقادیری هستند که در زمان خواندن یا نوشتن از/ به منبع داده مورد استفاده قرار می گیرند .

- تنظیما **System.Web** و **System.Net** که مابقی چیزها را نگهداری می کنند .

تنظیمات **System.Web** به چند زیر مقوله دیگر تقسیم می شود :

- ماژول های **HTTP** که برای اجرای کد ، صفحه را به سایر صفحات نشان می دهند .
- روتین های اشکالزدایی که در هنگام ترجمه باید فعال باشند .
- تکنیک احراز هویت .
- مدیر یا ناظر نقش که می تواند فعال یا غیرفعال باشد .
- مجوز یا عدم مجوز شناسائی بی نام (**anonymous identification**) .
- تنظیمات اداره کردن خطا .
- فایل داده **SiteMap** مورد استفاده ناوبری و منوها .
- پروفایل داده مورد استفاده جهت شناسایی کاربران .
- تنظیمات **E-Mail** برای پروتکل **SMTP** .
- تعریف فضاهای نام (**namespaces**) برا شناسایی محل اشیاء موجود در اشیاء بزرگتر .

System.Net فقط تنظیماتی را برای مقاصدمان ه یک سری مقادیر برای ارسال **E-Mail** است ، را نگهداری می کند. محتویات این فایل را می توانید به دو روش تغییر داده و اصلاح کنید : روش نخست ویرایش دستی آن در **VWD** است که زیاد جالب نیست . روش دوم استفاده از ابزار نظارتی وب سایت موجود در **ASP.NET** می باشد که می توانید آن را در **VWD** بکار ببرید . می توانید به منوی اصلی **VWD** رفته و از گزینه **WebSite** آیتم **ASP.NET Configuration** را انتخاب کنید . توسط یکسری از جعبه های دیالوگ می توانید مقادیری را که می خواهید **VWD** بطور خودکار در فایل **Web.config** تغییر دهد (بدون باز کردن فایل) را تنظیم کنید .

توضیحات بعدی که در مورد ساختار فایل **Web.config** می آید ، نگاهی به قسمت های مختلف فایل **Web.config** برنامه سایت بهبازار خواهد بود . این فایل که ساختار آن همانند فایل های **XML** می باشد (که حاوی برچسبهای باز و بسته که هر کدام برای یک خصوصیت هستند و گاهی اوقات نیز برای گره های فرزند هستند) . تنظیمات پیکربندی برنامه کاربردی با افزودن گره ها و ویژگی های مناسب انجام می شود . متونی که بین کاراکترهای خاص <!-- --> قرار دارند ، توضیحات (**comments**) هستند که می توانید با اضافه کردن آن ها در بخش های مختلف فایل ، دیگران را در فهمیدن بخش های مختلف کمک کنید .

در زیر کدهای فایل **Web.config** در سایت بهبازار را مشاهده می کنید :

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can
use the
    web admin tool to configure settings for your application.
Use
    the Website->Asp.Net Configuration option in Visual
Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
<configuration
xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">


     <appSettings/>
     <!--
          define the connection string to the database
     -->
     <connectionStrings>
          <add name="DB-name" connectionString="Data
Source=(local);Initial Catalog=DB-name;Integrated
Security=True" providerName="System.Data.SqlClient"/>
     </connectionStrings>
     <system.web>
          <!--
            Set compilation debug="true" to insert debugging
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.
        -->
            <compilation debug="true">
                <assemblies>
                     <add assembly="System.Design,
Version=2.0.0.0, Culture=neutral,
PublicKeyToken=B03F5F7F11D50A3A"/>
      </assemblies>
    </compilation>
          <!--
            The <authentication> section enables configuration
            of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->
            <authentication mode="Windows"/>
            <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error
pages
```

```
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404"
redirect="FileNotFound.htm" />
        </customErrors>
        -->

    <httpHandlers>
      <add verb="GET"
      path="FtbWebResource.axd"
      type="FreeTextBoxControls.AssemblyResourceHandler,
FreeTextBox" />
    </httpHandlers>

     </system.web>
 <system.net>
  <mailSettings>
   <smtp from="email-name">
    <network host="server-name" password="password"
userName="username" />
   </smtp>
  </mailSettings>
 </system.net>
</configuration>
```

در بخش ابتدایی فایل **Web.config** بهبازار که در زیر مشاهده می کنید :

```
<?xml version="1.0"?>
<!--
    Note: As an alternative to hand editing this file you can
use the
    web admin tool to configure settings for your application.
Use
    the Website->Asp.Net Configuration option in Visual
Studio.
    A full list of settings and comments can be found in
    machine.config.comments usually located in
    \Windows\Microsoft.Net\Framework\v2.x\Config
-->
```

برخی از بخش های کد فوق به صورت پیش فرض به کلیه فایل های جدید **Web.config** اضافه می شوند . خط اول شامل تعریف **XML** است که نشان می دهد فایل **Web.config** از استاندارد **XML** پیروی می کند . بخش بعدی شامل توضیحات مفصلی است که به شما یادآوری می کند تا بجای ویرایش دستی کد ، از ابزارهای نظارتی استفاده کنید . بخش آخر نیز گره ریشه فایل را مشخص می کند . گره

<configuration> شامل کلیه گره های فرزندی که حاوی تنظیمات مربوط به محتویات ذخیره شده در سایت هستند ، خواهد بود .

بخش بعدی شامل تنظیمات برنامه کاربردی دلخواهی است که می تواند در تغییر روشی که برنامه کاربردی نمونه برای محیط های مختلف اجرا می شود ، مفید باشد . بخش بزرگی که مابین **<!--**و **-->** قرار دارد ، توضیحاتی از جانب **VWD** برای برنامه نویسان و جزء تنظیمات واقعی نمی باشد .

بخش بعدی که رشته های اتصال است ، حاوی مجموعه اطلاعاتی درباره منابع داده است . این رشته معمولا شامل اطلاعات احراز هویت است که می توانید آن را برای اتصال کد و برنام خود به داده های ذخیره شده در بانک اطلاعاتی استفاده کنید .

```
<!--
          define the connection string to the database
-->
<connectionStrings>
          <add name="DBBehBazar" connectionString="Data
     Source=(local);Initial Catalog=DBBehBazar;Integrated
     Security=True" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

تنظیمات بعدی در **system.web** مربوط به مقدار ترجمه (**compiletion**) است . وقتی این مقدار **true** باشد **ASP.NET 2.0** خروجی را برای صفحه فراهم خواهد کرد تا هر مشکلی که در طی ساخت صفحه ایجاد شده است را نشان می دهد . این ویژگی در زمان پیاده سازی سایت مفید است ولی در زمان گسترش سایت باید آن را **false** کرد .

```
<system.web>
<!--
     Set compilation debug="true" to insert debugging
     symbols into the compiled page. Because this
     affects performance, set this value to true only
     during development.
-->
<compilation debug="true">
<assemblies>
     <add assembly="System.Design, Version=2.0.0.0,
     Culture=neutral, PublicKeyToken=B03F5F7F11D50A3A"/>
</assemblies>
</compilation>
<!--
     The <authentication> section enables configuration
     of the security authentication mode used by
     ASP.NET to identify an incoming user.
-->
```

بخش بعدی برای احراز هویت می باشد که به صورت زیر است :

```
<!--
    The <authentication> section enables configuration
    of the security authentication mode used by
    ASP.NET to identify an incoming user.
-->
<authentication mode="Windows"/>
```

بخش بعدی که در ارتباط با اداره کردن خطاهاست . **ASP.NET** را می توان بگونه ای تنظیم کرد که در صورت بروز مشکل ، صفحه خطای دلخواهی را به کاربر نشان دهد . کد زیر در فایل **Web.config** برای این منظور به کار می رود . که در صورت بروز خطا در صفحات سایت کاربر به صفحه **GenericErrorPage.htm** ارجاع داده می شود .

```
<!--
     The <customErrors> section enables configuration
     of what to do if/when an unhandled error occurs
     during the execution of a request. Specifically,
     it enables developers to configure html error pages
     to be displayed in place of a error stack trace.

<customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
<error statusCode="403" redirect="NoAccess.htm" />
<error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
-->
```

آخرین گروه تنظیمات در **System.net** قرار دارد . با استفاده از آن ها می توان در مواقعی که کاربر کلمع عبورش را فراموش کرده است و تقاضای ارسال آن با پست الکترونیک را دارد ، بطور خودکار آن را برای وی **E-Mail** کرد . بدون توجه به اینکه **e-mail** چگونه ایجاد می شود ، باید پروتکل **SMTP** را بصورت زیر در **Web.config** تعریف کنید :

```
<system.net>
  <mailSettings>
   <smtp from="email">
          <network host="server-name" password="password"
          userName="username" />
   </smtp>
  </mailSettings>
 </system.net>
```

و بالاخره همانند هر فایلی **XML** هر برچسبی که باز می شود باید با برچسبی بسته بشود . بنابراین تنظیمات فایل با دو برچسب زیر به پایان می رسد :

```
</system.web>
</configuration>
```

همانطور که دیدید ، فایل **web.config** حاوی تنظیمات گسترده ای است که کلیه صفحات می توانند به آن رجوع کنند . بدین ترتیب از مشخ کردن مکرر اطلاعات تکراری در هر صفحه ای که به آن ها نیاز دارد ، خلاصه می شوید .

۵-۳- Global.asax فایلی که کدهای کل سایت را نگهداری می کند

فایل **web.config** حاوی مقادیر و فایل **Global.asax** حاوی کدها می باشند . فایل **Global.asax** نیز همانند فایل **Web.config** در ریشه سایت قرار می گیرد .

کد موجود در فایل **Global.asax** در یکی از سه مورد زیر اجرا می شود . اولین مورد زمانی است که برنامه کاربردی کلا آغاز یا متوقف شود . دومین مورد زمانی است که هر کاربر استفاده از سایت را آغاز یا متوقف می کند . سومین مورد در پاسخ به رویدادهای خاصی است که می تواند بر روی هر صفحه ای اتفاق افتد مثل ورود یک کاربر یا بروز یک خطا . هر کدام از اینها بعنوان یک رویداد شناخته می شود . وقتی هر کدام از این رویدادها اتفاق می افتد ، **ASP.NET** به فایل **Global.asax** اطلاع می دهد و با ویرایش **Global.asax** می توانید کدهای مورد نظرتان را در آن قرار دهید تا در پاسخ به آن رویداد اجرا شوند .

فصل ششم

کنترل های کاربر (User Control)

۶-۱- کنترل های کاربر (User Control)

کنترل های کاربر در فایل هایی مجزا با پسوند **.ascx** ذخیره می شوند . هر گاه فایلی با این پسوند را ملاحظه کردید ، بدانید که با کنترل های کاربر سر و کار دارید . برای ایجاد یک کنترل کاربر باید یک شبه دستور **@Register** را در بالای **WebForm** اضافه کنید تا از طریق آن بتوانید کنترل کاربر را پیدا کنید :

```
<%@ Register TagPrefix="uc" TagName="UCStatic"
Src="UCStatic.ascx" %>
```

باید برچسب جدیدی را برای مشخص نمودن جایی که کنترل کاربر بر روی صفحه قرار خواهد گرفت ، اضافه کنید . این برچسب از عبارت **TagPrefix** که به دنبال آن یک کالن (:) ، **TagName** و **ID** و سپس ویژگی آشنای **runat="server"** می آید تشکیل شده است :

```
<uc:UCStatic ID="UCStatic1" runat="server" />
```

در نهایت باید خود کنترل کاربر را در یک فایل **.ascx** مجزا مشخص کنید . برخلاف **Web Form** ها در اینجا نیازی نیست تا برچسبهای اضافی **<html>** و **<body>** را مشخص کنید ، زیرا محتویات این کنترل به **<body>** حاوی صفحه اصلی اضافه خواهد شد . در واقع ، کل چیزی که باید داشته باشید ، خود کنترل هایی هستند که می خواهید آن ها را ضمیمه کنید . به طور مثال کدهای زیر کنترل کاربر آمار سایت را در سایت بهبازار نشان می دهد :

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="UCStatic.ascx.cs" Inherits="UserControl_UCStatic" %>
```

```
<table cellpadding="0" cellspacing="0" style="border-right:
#99cc00 1px; border-top: #99cc00 1px; border-left: #99cc00
1px; border-bottom: #99cc00 1px; width: 730px;" dir="ltr">
    <tr>
        <td dir="rtl" style="height: 25px; background-color:
#f7f6f3;
            text-align: center; border-top: #99cc00 1px solid;
border-left: #99cc00 1px solid; width: 85px; border-bottom:
#99cc00 1px solid;" valign="middle">
            <asp:Label ID="lblVisitorAllDay" runat="server"
Font-Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="0000000000"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 105px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: left" valign="middle">
            <asp:Label ID="Label5" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="ها کننده بازدید کل :"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 75px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: center" valign="middle">
            <asp:Label ID="lblVisitorToday" runat="server"
Font-Names="Tahoma" Font-Size="9pt"
                ForeColor="#212142"
Text="000000"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 100px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: left" valign="middle">
            <asp:Label ID="Label3" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="امروز کننده بازدید :"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 70px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: center" valign="middle">
            <asp:Label ID="lblAdCount" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="000000"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 80px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: left" valign="middle">
            <asp:Label ID="Label1" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="ها آگهي کل :"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 80px; border-bottom: #99cc00 1px solid;
```

```
            height: 25px; background-color: #f7f6f3; text-
align: center" valign="middle">
            <asp:Label ID="lblDate" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="00/00/0000"></asp:Label></td>
        <td dir="rtl" style="border-top: #99cc00 1px solid;
width: 40px; border-bottom: #99cc00 1px solid;
            height: 25px; background-color: #f7f6f3; text-
align: left" valign="middle">
            <asp:Label ID="lable2" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="تاریخ :"></asp:Label></td>
        <td dir="rtl" style="border-right: #99cc00 1px solid;
border-top: #99cc00 1px solid;
            width: 95px; border-bottom: #99cc00 1px solid;
height: 25px; background-color: #f7f6f3;
            text-align: right" valign="middle">
             <asp:Label ID="Label2" runat="server" Font-
Names="Tahoma" Font-Size="9pt" ForeColor="#212142"
                Text="سایت روزانه آمار :"></asp:Label></td>
    </tr>
</table>
```

این کنترل سپس می تواند در هر جایی از صفحات وب که شبه دستور **@Register** را در آن جا قرار داده اید و برچسبی برای کنترل اضافه کرده اید ، چفت شده و باقی بماند .

۶-۲- ایجاد یک کنترل کاربر

۱. در **Solution Explorer** بر روی وب سایت راست – کلیک کرده و گزینه **Add New Item** را انتخاب کنید . **WebUserControl** را انتخاب کنید ، و نامی که مورد نظرتان است را در بخش **Name** وارد کنید .
۲. اینک باید در صفحه ی جدیدی که باز شده است کنترل های مورد نظر خود را به صفحه اضافه کنید و کدهای مورد نیاز را بنویسید . مانند کدهایی که در بالا برای کنترل **UCStatic** در سایت بهبازار نوشته شده است .
۳. اکنون کنترل کاربر مورد نظر آماده استفاده در سایت با استفاده همان دستوراتی که در بالا توضیح داده شد می باشد .

به همین سادگی می توانید کنترل های کاربر خود را تعریف کنید و در هر جایی در صفحات سایت خود از آن ها به کرات استفاده نمایید . این کار مزیتش این است که از نوشتن کدهای تکراری برای قسمت های یکسان سایت جلوگیری می شود . به عنوان مثال منوی مدیریت در سایت بهبازار نیز یک کنترل کاربر است

که با یک بار تعریف این کنترل در تمامی صفحات مدیریت کاربر استفاده می گردد ، و بدین طریق از کدنویسی تکراری و ملال آور جلوگیری می شود .

۶-۳- کنترل های مرکب

دیدید که چگونه می توان کنترل های کاربر را داخل صفحات قرار داد ، اما کار به همین جا خاتمه نمی یابد ، می توانید کنترل های کاربر را در داخل کنترل های کاربر دیگر قرار دهید و آن ها را بصورت تودرتو (**nested**) استفاده کنید . این نوع کنترل ها را کنترل های مرکب (**composite Controls**) می نامند . سناریوئی را در نظر بگیرید که در آن به گروهی از کنترل های کاربر مثل یک کنترل **News** و یک کنترل **Login** و یک کنترل **Fixture Viewer** نیاز دارید . بجای اینکه هر بار ارجاع های جداگانه ای را برای این کنترل ها مشخص کنید ، می توانید آن ها را در یک کنترل کاربر قرار داده و بجای آن ها از آن استفاده کنید . بدین ترتیب قابلیت استفاده مجدد از کدها در برنامه نیز بهبود می یابد .

فصل هفتم

طراحی کامپوننت در ASP.NET

در این بخش سعی می کنیم ساده ترین شکل ایجاد و نحوه استفاده از کامپوننت ها را شرح دهیم تا هم دلیل استفاده از کامپوننت ها مشخص شود و هم ساده ترین مثال برای کامپوننت ها ذکر شده باشد .

۷-۱- پیش نیازها :

- آشنایی با نحوه ایجاد صفحات وب
- تسلط کافی بر زبان **C#**
- تسلط کافی بر **HTML**

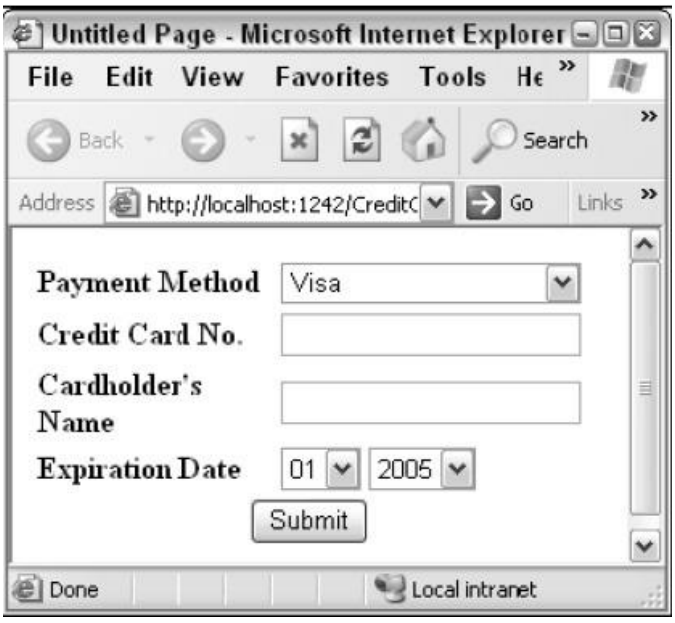
۷-۲- یک مثال :

کدهای **HTML** زیر را در نظر بگیرید :

```
<html xmlns=http://www.w3.org/1999/xhtml>
<head>
<title> Component Design Example </title>
<body>
<form method="post" action="Defualt.aspx" id="form1">
<table id="ccf" style="font-weight:bold;">
</td>    <strong>Payment Method</strong> <td> <tr>
</tr>
</td>   <strong>Credit Card No.</strong> <td> <tr>
```

```
</tr>     </td>     <input type="text"/> <td>
</td>     <strong>CardHolder's Name</strong>  <td> <tr>
</tr>     </td>     <input type="text"/> <td>
</td>   <strong>Expiration Date</strong> <td> <tr>
<select> <td>
<option value="1">01</option>
...
</select>
<select><option value="2005">2005</option>
</tr>     </td> </select>
<td colspan="2" align="center"> <tr>
</td>  <input type="submit" value="submit" />
</tr>
</table>
</form>
</body>
</html>
```

با اجرای کدهای فوق صفحه زیر را خواهیم داشت :



شکل ۷-۱- طراحی کامپوننت

خوب ، تا اینجا هر چه بود **HTML** ساده بود و هیچ چیز اضافه ای نداشت . به نظر شما اگر به جای نوشتن این همه کد **HTML** یک خط کد بنویسیم و بتوانیم از این کامپوننت بارها استفاده کنیم کار درستی کرده ایم ؟

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">

        <custom:creditcardform1 runat="server" id="ccf" />

    </form>
</body>
</html>
```

اگر جواب شما مثبت است ادامه مطلب را بخوانید !

کنترل **CreditCardForm1** یک جعبه است که کدهای مطرح شده در مثال را در خود به صورت کپسوله شده در آورده است . **CreditCardForm1** یک کنترل سفارشی است که از کلاس استاندارد پایه **ASP.NET** بنام **CreditCardForm1 Control** مشتق شده است . تمام **Server Control** ها در **ASP.NET** به طور مستقیم یا غیر مستقیم از کلاس **Control** مشتق شده اند . در واقع مشتق شدن یک **Server Control** از این کلاس است که به آن مفهوم **Server Control** را می دهد .

این کلاس تمام امکانات پایه ای لازم برای یک **Server Control** را در **ASP.NET** فراهم می کند . یکی از این متدهای پایه ، متدی به نام **Render** است . این متد در جایی که یک **Server Control** کدهای **HTML** خود را ایجاد می کنند کار می کند .

دقت کنید کلمه **Render** در زمانی که **Browser** ها کدهای **HTML** را دریافت می کنند به معنی نمایش می باشد (**Display**) . اما در اینجا به مفهوم ایجاد می باشد (**Generate**) .

در زمان اجرا صفحه محتوای کامپوننت یک نمونه از کلاس **HtmlTextWriter** را ایجاد و به متد **Render** آن **Server Control** ارسال می کند . در کلاس **Server Control** ایجاد شده یک متد بنام **Write** مسئولیت خروجی دادن کدهای **HTML** مورد نیاز را بر عهده دارد .

مثال :

```
<table style="width:287px;height:124px;">
```

```
writer.Write("<table style='width:287px;height:124px;'>");
```

گام نخست :

- یک پروژه جدید وب ایجاد کنید .
- سپس به آن یک فایل کلاس با هر نام دلخواهی اضافه کنید .

از این مرحله به بعد برای ساختن یک کامپوننت ۳ مرحله در پیش داریم :

- نوشتن (به دست آوردن) کد **HTML** کامپوننت مورد نظر .
- نوشتن کلاس مشتق شده ای از کلاس پایه **Control** .
- **Override** کردن متد **Render**

گام دوم :

کدهای زیر را به طور کامل در فایل کلاس ایجاد شده کپی کنید :

```
sing System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

/// <summary>
/// Summary description for Class1
/// </summary>
```

```
namespace CustomComponents
{
    public class CreditCardForm1 : Control
    {
        protected override void Render(HtmlTextWriter writer)
        {
            writer.Write("<table style='width:287px;height:124px;border-width:0;'>");
            writer.Write("<tr>");
            writer.Write("<td>Payment Method</td>");
            writer.Write("<td>");
            writer.Write("<select name='PaymentMethod' id='PaymentMethod'
style='width:100%;'>");
            writer.Write("<option value='0'>Visa</option>");
            writer.Write("<option value='1'>MasterCard</option>");
            writer.Write("</select>");
            writer.Write("</td>");
            writer.Write("</tr>");
            writer.Write("<tr>");
            writer.Write("<td>Credit Card No.</td>");
            writer.Write("<td><input name='CreditCardNo' id='CreditCardNo'
type='text' /></td>");
            writer.Write("</tr>");
            writer.Write("<tr>");
            writer.Write("<td>Cardholder's Name</td>");
            writer.Write("<td><input name='CardholderName' id='CardholderName'
type='text' /></td>");


            writer.Write("</tr>");
            writer.Write("<tr>");
            writer.Write("<td>Expiration Date</td>");
            writer.Write("<td>");
            writer.Write("<select name='Month' id='Month'>");
            for (int day = 1; day < 13; day++)
            {
                if (day < 10)
                    writer.Write("<option value='" + day.ToString() + "'>" + "0" +
                    day.ToString() + "</option>");
                else
                    writer.Write("<option value='" + day.ToString() + "'>" +
                    day.ToString() + "</option>");
            }
            writer.Write("</select>");
            writer.Write(" ");
            writer.Write("<select name='Year' id='Year'>");
            for (int year = 2005; year < 2015; year++)
            {
                writer.Write("<option value='" + year.ToString() + "'>" +
                year.ToString() + "</option>");


               }
               writer.Write("</select>");
               writer.Write("</td>");
               writer.Write("</tr>");
               writer.Write("<tr>");
               writer.Write("<td align='center' colspan='2'>");
               writer.Write("<input type='submit' value='Submit' />");
               writer.Write("</td>");
               writer.Write("</tr>");
               writer.Write("</table>");
               base.Render(writer);
        }
    }
}
```

سپس پروژه را یک بار **Build** کنید ، اکنون شما دارای یک کامپوننت هستید .

۷-۳- چگونه از این کامپوننت استفاده کنیم ؟

گام سوم :

در هر صفحه که نیاز به استفاده از این کامپوننت را دارید ، کد زیر را به قسمت Source آن اضافه کنید .

```
<%@ Register TagPrefix="custom" Namespace="CustomComponents" %>
```

از این پس هر گاه در این صفحه بخواهیم از این کامپوننت استفاده کنیم کافی است به شکل زیر عمل کنیم :

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<%@ Register TagPrefix="custom" Namespace="CustomComponents" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    <custom:CreditCardForm1 runat="server" ID="ddo"></custom:CreditCardForm1>

    </div>
    </form>
</body>
</html>
```

فصل هشتم

کار با داده های بانک اطلاعاتی و مقید سازی داده ها

- اتصال به بانک اطلاعاتی
- بازیابی رکوردها از جداول بانک اطلاعاتی
- اضافه کردن رکورد به بانک اطلاعاتی
- اصلاح رکوردها
- حذف رکوردهای جدول
- استفاده از رویه های ذخیره شده
- بهبود کارآیی صفحه با جمع آوری رشته های اتصال
- تفاوت های اصلی **DataSet** و **DataReader**

۸-۱- کار با داده های بانک اطلاعاتی و مقید کردن داده ها

در این قسمت با نحوه انجام عملیاتی آشنا می شوید که در مقوله بانک اطلاعاتی معمول و متداول هستند و همچنین در سایت بهبازار از آن ها استفاده شده است . ابتدا با نحوه نمایش و بازیابی رکوردهای بانک اطلاعاتی ، افزودن رکورد جدید ، اصلاح رکوردهای موجود و حذف رکوردهای مورد نظر آشنا می شوید .

برای انجام تمام این اعمال می توانیم هم از میدان **System.Data.SqlClient** استفاده کنیم و هم از میدان **System.Data.OLEDB** که ما در سایت بهبازار از میدان اولی استفاده کرده ایم . وقتی با **SQL Server** سروکار داریم ، باید میدان **System.Data.SqlClient** را به صفحه خود اضافه کنیم .

ولی وقتی با سایر سیستم های مدیریت بانک اطلاعاتی (از قبیل **Access** یا **Oracel**) کار می کنیم ، باید میدان **System.Data.OLEDB** را بکار ببریم .

۸-۲- اتصال به بانک اطلاعاتی

برای دسترسی به بانک اطلاعات ابتدا باید با آن ارتباط برقرار کنیم ؛ یعنی به آن وصل شویم . نحوه ارتباط باز هم بستگی به این دارد که بخواهیم به کدام نوع بانک اطلاعات وصل شویم . ما در سایت خود از بانک اطلاعاتی **SQl Server 2000** استفاده کرده ایم بنابراین از طریق زیر با این بانک ارتباط برقرار می نماییم :

```
connectionString="Data Source=(local);Initial Catalog=DB-
name;Integrated Security=True"
providerName="System.Data.SqlClient"
```

```
ConnectionStringSettings settings;
settings = ConfigurationManager.ConnectionStrings["DB-name"];
```

```
con = new SqlConnection(settings.ToString());
```

همانطور که ملاحظه می کنید ابتدا مقدار **ConnectionSring** را در فایل **Web.config** تعریف می کنیم و سپس از این مقدار در فرم های وب خود استفاده می کنیم . مقدار متغیر con در آخرین خط یک اتصال به بانک اطلاعاتی را تعریف می کند و هنگام استفاده از این اتصال باید این اتصال را ابتدا باز کنیم که از طریق دستور زیر این کار را انجام می دهیم .

```
Con.Open();
```

۸-۳- بازیابی رکوردها از جداول بانک اطلاعاتی

یکی از دستورات **SQL** پر استفاده در صفحات **ASP.NET** ، دستور **Select** است . این دستور ، برای بازیابی رکوردهایی به کار می رود که با شروط (یا معیارهای) از پیش تعیین شده ای مطابقت می کنند . رسم الخط استفاده از دستور **Select** به طور کلی چنین است :

```
SELECT  column1, column2, …
FROM  tablename1, tablename2, …
WHERE  search condition
```

برای مثالی از این دستور ما تابعی را استفاده می کنیم که توسط آن دیتاگریدهای خود را ، از داده های بانک اطلاعاتی پر می نماییم . کدهای زیر نشان دهنده این تابع می باشند :

```
Sql = "SELECT * FROM  ContactUs ";
```

علامت × یک کاراکتر عمومی است که جایگزین تمام ستون ها می شود و اگر از **WHERE** استفاده نکنیم ، تمام سطرها از جدول بازیابی می شوند .

به طور کلی ف در اجرای یک دستور **SELECT** در صفحات **ASP.NET** خود ، این چهار مرحله را باید پی گرفت :

۱. با بانک اطلاعاتی ارتباط برقرار کرد .
۲. دستور **SELECT** مناسب را تشکیل داده و در یک شیء **Command** قرار داد .
۳. با استفاده از متد **ExecuteReader()** دستور را اجرا کرده و نتیجه را به صورت یک **DataReader** یا **DataSet** تحویل گرفت .
۴. با استفاده از **DataReader** یا **DataSet** خود به پردازش اطلاعات مشغول شد .

مثالی از این مورد را در زیر مشاهده می کنید که در تابع **BindToGridView()** ، اطلاعات از بانک اطلاعاتی خوانده شده و به یک **DataSet** و همچنین یک **GridView** انقیاد می شود :

```
private void BindToGridView()
{
    try
    {
        Sql = "SELECT * FROM  ContactUs ";

        ds = new DataSet();
        da = new SqlDataAdapter(strSql, con);
        da.Fill(ds, "ContactUs");

        GridView.DataSource = ds;
        GridView.DataBind();
    }
    catch
```

```
        {
            Response.Redirect("ErrorPage.aspx");
        }
    }
```

در این تابع اطلاعات از جدول **Contact** خوانده شده و در **DataSet** ذخیره می شوند و پس از آن به **GridView** انقیاد می شوند . در صورتی که این مراحل با خطا مواجه شوند کاربر به صفحه **ErrorPage.aspx** ارجاع داده می شود .

۸-۴- اضافه کردن رکورد به بانک اطلاعاتی

با استفاده از دستور **Insert** ، در **SQL** می توان رکوردهای جدیدی را به جدول بانک اطلاعاتی خود اضافه کرد. رسم الخط این دستور به صورت زیر می باشد :

```
INSERT  tablename (column1, column2, . . .)
VALUES  (value1, value2, . . .)
```

یعنی برای درج رکورد جدید به یک جدول ، اسامی ستون ها و مقادیر متناظر با هر ستون را در جلوی دستور **Insert** می نویسیم . برای نمونه ای از این دستور کدهای زیر را از سایت بهبازار برای قسمت تماس با ما در نظر بگیرید :

```
string Sql = "INSERT INTO ContactUs "
                    + " (Name,Email,Comment,Date,Time)"
                    + "VALUES(N'"
                    + Name.Text.Trim() + "',N'"
                    + Email.Text.Trim() + "',N'"
                    + Comment.Text.Trim() + "',N'"
                    + Date() + "',N'"
                    + Time() + "')";
```

برای اجرای یک دستور **Insert** درصفحات **ASP.NET** مراحل زیر را باید دنبال کرد :

۱. با بانک اطلاعاتی ارتباط برقرار کنید .
۲. دستور **Insert** مورد نظر را تشکیل داده و در کلاس **Command** قرار دهید .
۳. با استفاده از متد **ExecuteNonQuery()** دستور مزبور را اجرا کنید .

در زیر تابعی برای درج نظرات ارسال شده توسط بینندگان سایت بهبازار را مشاهده می کنید :

```
private void Insert()
{
    try
    {
        string Sql = "INSERT INTO ContactUs "
               + " (Name,Email,Comment,Date,Time)"
               + "VALUES(N'"
               + Name.Text.Trim() + "',N'"
               + Email.Text.Trim() + "',N'"
               + Comment.Text.Trim() + "',N'"
               + Date() + "',N'"
               + Time() + "')";

        da = new SqlDataAdapter(Sql, con);

        if (con.State == ConnectionState.Closed)
            con.Open();

        da.InsertCommand = new SqlCommand(Sql, con);
        da.InsertCommand.ExecuteNonQuery();

        con.Close();
    }
    catch
    {
        lbl.Text = "آمده بوجود اطلاعات ذخیره در مشکلي !";
        lbl.Visible = true;
        Response.Redirect("ContactUs.aspx");
    }
}
```

۸-۵- اصلاح رکوردها

برای اصلاح و تغییر رکوردهای موجود یک جدول در بانک اطلاعاتی از فرمان **UPDATE** در **SQL** استفاده می شود . رسم الخط یک فرمان پایه **UPDATE** به صورت زیر است :

```
UPDATE tablename SET column1 = value1,column2 = value2 ,...
WHERE search condition
```

اصلاح یک جدول با تعیین مقادیر جدید برای ستون هایی از جدول انجام می شود که سطرهای آن ها در شرط خاصی می گنجند . مثلا فرض کنید جدولی دارید به نام **Authors** که یک ستون به نام **au_lnam** دارد . دستور زیر هر جا برای این ستون مقدار **Bennet** ثبت شده باشد ، آن را به **Smith** تغییر می دهد :

```
UPDATE  Authors  SET  au_lname = 'Smith'
WHERE  au_lname = 'Bennet'
```

اصلاح رکوردی که اصلا وجود ندارد ، تولید خطا نخواهد کرد .

برای اجرای یک دستور **UPDATE** در صفحات **ASP.NET** خود ، مراحل زیر را باید دنبال کنید :

۱. با بانک اطلاعاتی مربوطه ارتباط برقرار کنید .
۲. دستور **Update** مورد نظر را تشکیل داده و در یک **Command** قرار دهید .
۳. با استفاده از متد **ExecuteNonQuery()** دستور را اجرا کنید .

۸-۶- حذف رکوردهای جدول

برای حذف اطلاعات از یک بانک اطلاعات ، از دستور **DELETE** در **SQL** استفاده می شود . رسم الخط نوشتن یک دستور **DELETE** پایه به این قرار است :

```
DELETE  tablename  WHERE  search condition
```

اگر مثلا بخواهیم تمام سطرهایی از جدول **Authors** را پاک کنیم که فیلد **au_lname** آن ها حاوی مقدار **Bennet** باشد ، از چنین دستوری استفاده می کنیم :

```
DELETE  Authors  WHERE  au_lname = 'Bennet'
```

در زیر مثالی از این دستور که در سایت بهبازار برای حذف نظرات استفاده شده است را ملاحظه می کنید :

```
string SqlTemp = "DELETE FROM ContactUs WHERE ContactUsNo = "
+ GridView.DataKeys[e.RowIndex].Value;

SqlDataAdapter da = new SqlDataAdapter(Sql, con);
da.DeleteCommand = new SqlCommand(Sql, con);

con.Open();
da.DeleteCommand.ExecuteNonQuery();
con.Close();
```

این دستورات نظر مورد نظر را از جدول نظرات که دارای شماره خاصی است حذف می کند .

۸-۷- استفاده از رویه های ذخیره شده

اجرای یک دستور **SQL** از درون صفحات **ASP.NET** به دو صورت امکان پذیر است . روش اول این است که دستور مورد نظر را مستقیما در کد صفحه نوشته و اجرا نماییم . اما روش دوم این است که آن دستور را

در قالب یک «رویه ذخیره شده » بسته بندی کرده و سپس آن رویه را از داخل خود صفحه به اجرا درآوریم .

ساخت رویه های ذخیره شده زحمت بیشتری نسبت به اجرای مستقیم دستورات **SQL** دارد ، اما در عوض در وب سایت هایی که وابستگی زیادی به بانک اطلاعات دارند ، از این طریق می توانیم به کارایی بسیار بهتر و سریع تری دست یابیم . همان که می دانید ، سرور بانک اطلاعات هنگام اجرای دستورات **SQL** آن ها را **Parse** ، **Compile** و **Optimaze** می کند ؛ یعنی قبل از اینکه یک دستور **SQL** واقعا شروع به کار روی بانک اطلاعات بکند ، مدت زمان قابل توجهی را بابت این سه عمل از دست می دهد . اما وقتی دستور خود را در قالب رویه ذخیره شده به سرور دهیم ، این دستور فقط یکبار **Parse** ، **Compile** و **Optimize** می شود و در اجراهای بعدی از نسخه کامپایل و بهینه شده آن استفاده خواهد شد .

یکی دیگر از مزایای استفاده از رویه های ذخیره شده این است که می توانیم چندین دستور را در قالب یک رویه ذخیره کنیم و آن ها را گروهی به اجرا درآوریم . مثلا ، با ذخیره چند دستور **Update** که هر کدام جدول متفاوتی را مورد اصلاح قرار می دهند ، می توانیم از طریق رویه ذخیره شده رکوردهای چند جدول را یکباره اصلاح کنیم .

استفاده از قابلیت های برنامه ای زبان **Transact-SQL** هم زمانی امکان پذیر است که از رویه های ذخیره شده استفاده کنیم . همان طور که می دانید ، زبان **Transact-SQL** قابلیت کار با متغیرها ، دستورات شرطی ، حلقه ها و توابع را دارد ؛ یعنی با این زبان می توانیم یک مینی برنامه پیشرفته در قالب رویه های ذخیره شده بنویسیم .

در ضمن ، استفاده از رویه های ذخیره شده این فرصت را به ما می دهد که صفحه **ASP.NET** خود را وابسته به جداول بانک اطلاعاتی نکنیم . به عبارت دیگر ، اگر به هر دلیلی لازم شد فرمت و ساختار داخلی جداول مورد استفاده در برنامه خود را تغییر دهیم ، دیگر لازم نیست به سراغ کد صفحه برویم ، بلکه می توانیم با اصلاح رویه های ذخیره شده ، برنامه خود را با این ساختار جدید جداول سازگار کنیم .

برای مثالی از استفاده رویه های ذخیره شده در صفحات وب خود ، نمونه ای را از سایت بهبازار ذکر می کنیم . کدهای زیر برای تشخیص هویت کاربر و صحت کلمه عبور و نام کاربری وی به کار می روند که در صفحه ورود به سایت به کار رفته اند . و شما مشاهده می کنید :

```
private string VerifyPassword(string Email, string Password)
{
    try
    {
```

```
        cmd = new SqlCommand("VerifyPassword", con);
        cmd.CommandType = CommandType.StoredProcedure;
        parmReturn = cmd.Parameters.AddWithValue("@return",
SqlDbType.Int);
        parmReturn.Direction = ParameterDirection.ReturnValue;
        cmd.Parameters.AddWithValue("@Email", Email);
        cmd.Parameters.AddWithValue("@Password", Password);
        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
        return cmd.Parameters["@return"].Value.ToString();
    }
    catch
    {
        Response.Redirect("ErrorPage.aspx");
        return "";
    }

}
```

تابع فوق برای تشخیص صحیح یا غلط بودن نام کاربری و کلمه عبور وارد شده توسط کاربر به کار می روند . این تابع در صفحه ای که برای ورود به سایت نوشته شده است بکار می رود و همانطور که مشاهده می کنید در این کدها هیچ دستور **SQL** ای استفاده نشده است و برنامه عاری از این کدهاست .

دستورات **SQL** استفاده شده برای این منظور در یک رویه ذخیره شده که در بانک اطلاعاتی قرار دارد نوشته شده است . که رویه ذخیره شده مذکور را در زیر می بینید :

```
ALTER PROCEDURE [dbo].[verifyPassword]
(
  @Email  nvarchar( 100) ,
  @Password  nvarchar( 20 )
)
AS
DECLARE @Found nvarchar( 100 )
SELECT @Found = Email
  FROM UB
  WHERE Email = @Email
  AND Password = @Password
IF @Found IS NOT NULL
  Return 0
ELSE  IF Exists ( SELECT Email
    FROM UB WHERE Email = @Email )
    Return 2
  ELSE
    RETURN 1
```

۸-۸- بهبود کارآیی صفحه با جمع آوری رشته های اتصال

ایجاد ارتباط با بانک اطلاعات عملی زمان بر و پر هزینه برای سیستم مدیریت بانک اطلاعات است و عملکرد صفحه **ASP.NET** را کند می کند . به علاوه ، بانک اطلاعات با تعداد محدودی متقاضی می تواند ارتباط برقرار کند ، و هر ارتباط نیازمند میزان حافظه معینی است .

اگر قصد دارید به صدها کاربری که به وب سایت شما سر می زنند همزمان سرویس بدهید ، ایجاد ارتباط جداگانه با بانک اطلاعات برای هر کاربر می تواند تاثیری منفی بر عملکرد و کارایی وب سایت شما بگذارد .

خوشبختانه برای جلوگیری از وقوع چنین حالتی راه حل جالبی وجود دارد که به «انبار کردن اتصالات» یا **Connection Pooling** موسوم است . در این روش ، مجموعه ای از «کانکشن ها» همواره به حالت **open** نگه داشته می شوند تا بتوان آن ها را میان چند کاربر به اشتراک گذاشت . وقتی کانکشن جدیدی را درخواست بکنیم ، یک کانکشن فعال از انباره حذف می شود . و هنگامی که کانکشن را می بندیم (**close** می کنیم) ، دوباره به انباره برمی گردد .

برای استفاده بهینه از این روش باید دو نکته را در صفحات **ASP.NET** خود رعایت کنید . اولا ف باید دقت کنید هر وقت ارتباطی را با بانک اطلاعات برقرار می کنید ، دقیقا از همان رشته اتصال استفاده کنید . فقط کانکشن هایی که با رشته اتصال یکسان برقرار می شوند در یک انباره جمع می شوند . حتی تفاوت های بسیار کوچک در رشته های اتصال باعث اختلاف می شوند . کانکشن ها فقط زمانی انبار می شوند که با رشته هایی برقرار شوند که دقیقا کاراکتر به کاراکتر یکسان باشند . به همین دلیل ، بهتر است رشته اتصال خود را در یک مکان ایجاد کنید و در تمام صفحات خود از همان رشته اتصال استفاده نمایید . مثلا ، می توانید رشته اتصال خود را در فایل **web.config** قرار دهید و هر وقت خواستید ارتباطی را با بانک اطلاعات برقرار کنید ، آن را از این فایل استخراج کنید . یک راه دیگر این است که رشته اتصال را در متغیر حفظ وضعیت **Application** قرار دهید .

برای استفاده بهینه از روش انبار کردن اتصالات باید به یک نکته دیگر هم دقت داشته باشید : همیشه ارتباط را در اولین فرصت بعد از پایان کار خود قطع کنید . اگر خودتان صراحتا اتصال را با استفاده از متد **close()** قطع نکنید ، این اتصال هیچ وقت به انباره برنمی گردد .

استفاده از شیوه انبار کردن کانکشن ها یک مزیت دیگر هم دارد و آن این است که می توانیم پارامترهایی اضافی به رشته اتصال اضافه کنیم که در عملکرد انباره اثر دارند . مثلا می توانیم اندازه حداقل و حداکثر انباره را تعیین کنیم یا حتی فرآیند انبار کردن کانکشن ها را کلا غیرفعال کنیم .

پارامترهای اضافی که در این صورت می توانیم به رشته اتصال **SQL Server** اضافه کنیم به قرار زیر هستند :

- **Connection Lifetime** – اتصال را بعد از گذشت مدت زمانی مشخص (برحسب ثانیه) نابود می کند . مقدار پیش فرض بر این پارامتر **0** است که نشانه این است که اتصال هیچ وقت نباید نابود شود .
- **Connection Reset** – معلوم می کند وقتی کانکشن به انباره برگشت **reset** بشود یا نه . پیش فرض **true** است.
- **Enlist** – معلوم می کند کانکشن به طور خودکار در بطن تراکنش جاری گنجانده شود یا نه . پیش فرض **true** است.
- **Max Pool Size** – تعداد حداکثر کانکشن هایی که در یک انباره تنها مجاز هستند . پیش فرض ۱۰۰ است .
- **Min Pool Size** – تعداد حداقل کانکشن هایی که در یک انباره تنها مجاز هستند . پیش فرض **0** است .
- **Pooling** – تعیین می کند اساسا سیستم **Connection Pooling** فعال باشد یا نه . پیش فرض **True** است .

۸-۹- تفاوت های اصلی DataSet و DataReader :

DataSet	DataReader
با تمام **Data Provider** ها کار می کند و شی **DataAdapter** توسط متد **Fill** آن را از اطلاعات پر می نماید .	برای هر **Data Provider** یک شیء جداگانه دارد ، برای مثال : **SqlDataReader, OLDBDataReader, ODBCDataReader**
مقادیر بازیابی شده هم خواندنی و هم قابل تغییر	مقادیر بازیابی شده فقط خواندنی می باشند .

هستند .	
هر زمان و در هر جایی می توان به داده ها دسترسی داشت .	مقادیر بازیابی شده **Forward_Only** می باشند ، یعنی هنگامی که از سطر خاصی گذشتیم ، قابل برگشت نیست ، در این حالت باید **Close** شود و دوباره از داده ها پر شود .
کلیه مقادیر مورد نظر را از منبع داده دریافت نموده و در یک لحظه در حافظه قرار می دهد .	اطلاعات را در یک ارتباط مستقیم ارائه می کند . به عبارت دیگر در هر لحظه یک سطر از اطلاعات در حافظه جای می گیرد .
از منابع حافظه و **IIS** بیشتری استفاده می نماید تا کل اطلاعات را بازیابی نماید . در اینجا باید **Connection** مربوطه **Open** شود تا متد **Fill** اجرا گردد .	از منابع حافظه و کمی هم از **IIS** استفاده می نماید . به دیتابیس متصل می باشد تا زمانی که **Connection** بسته نشده باشد .
توسط نام ، ستون ها ارجاع داده می شوند ، همچنین می توان معین نمود که ستون مربوط به کدام سطر ارجاع داده شود .	توسط نام و شماره ، ستون ها در آن ارجاع داده می شوند .
شامل مجموعه ای **DataTable** ها می باشد . کلید اصلی ممکن است برای هر **DataTable** قرار گیرد و ارتباطات و شروط ممکن است در بین آن ها برقرار شود .	مفاهیم کلید اصلی ، شرط ، **View** و سایر مفاهیم دیتابیس های رابطه ای را درک نمی کند، به استثنای سطر و ستون .
می توان اطلاعات را در دیتاست تغییر داده و تغییرات را در منبع داده ثبت کرد .	نمی توان منبع داده را بروز رسانی نمود .
می تواند توسط چندین منبع داده **Fill** شود .	فقط می تواند به یک منبع داده متصل شود .

جدول ۸-۱- مقایسه **DataSet** و **DataReader**

فصل نهم

انتقال مقادیر بین صفحات و حفظ حالت در ASP.NET

(State Management)

- حفظ حالت سمت کلاینت (**Client Side**)
- حفظ حالت سمت سرور (**Server Side**)
- چگونگی تصمیم گیری بین انتخاب حالت های مختلف

فرمهای وب **ASP.NET**، مدل برنامه‌نویسی رویدادگرای شگرفی را برای توسعه‌گران فراهم می‌کنند. این موضوع طراحی سرتاسر برنامه کاربردی شما را ساده می‌کند ولی مسائل و مشکلات خاص خود را نشان می‌دهد. برای مثال، در **ASP** کلاسیک شما به آسانی می‌توانید مقادیر را با استفاده از **POST** از یک صفحه **ASP** به صفحه‌ای دیگر ارسال نمایید. اما اگر می‌خواهید در مدل فرمهای وب (یا همان مدل برنامه‌نویسی **ASP.NET**) برنامه‌نویسی کنید، همان چیز در **ASP.NET** ممکن نمی‌باشد. اما راههایی برای غلبه بر این وضعیت وجود دارند که می‌توانند مورد استفاده قرار بگیرند. در این مقاله موضوعات زیر را مورد بررسی قرار خواهیم داد:

- چگونگی ارسال مقادیر با استفاده از **Querystring**

• چگونگی استفاده از متغیرهای **Session** برای ارسال مقادیر

• چگونگی استفاده از متد **Server.Transfer** برای ارسال مقادیر

۹-۱- استفاده از Querystring

Querystring یک مکانیسم قدیمی برای ارسال مقادیر در بین صفحات است. مزیت اصلی این متد سادگی آن است. اما عیب آن این است که پس از ارسال، مقادیر در نوار آدرس مرورگر قابل مشاهده می‌باشند و نمی‌توان آبجکت‌ها را از این طریق ارسال کرد. این متد مناسب‌ترین راه برای ارسال تعداد کمی از مقادیری است که نیازی به محافظت از دید دیگران ندارند. برای اعمال کردن این متد مراحل زیر را انجام دهید:

• یک فرم وب با کنترل‌هایش را درست کنید

• یک کنترل دکمه‌ای **Button** یا **LinkButton** برای ارسال فرم به سرور بر روی فرم قرار دهید

• در رویداد کلیک دکمه یک متغیر از نوع **String** تعریف کنید که **URL** را برای فرم دیگر که مقادیر قرار است به آنجا ارسال شوند، نگه می‌دارد.

• مقادیر کنترل‌ها را در قالب پارامترهای **QueryString** در متغیر از نوع **String** قرار دهید

• از متد **Response.Redirect** که از متغیر **String** تعریف شده استفاده می‌کند برای هدایت کاربر به صفحه دیگر استفاده نمایید

قطعه کد زیر چگونگی انجام این مراحل را نشان می‌دهد:

فرم وب منبع

```
private void Button1_Click(object sender, System.EventArgs e)
{
      string url;
      url = "anotherwebform.aspx?name=" +
      TextBox1.Text + "&email=" +
      TextBox2.Text;
      Response.Redirect(url);
}
```

فرم وب مقصد

```
private void Page_Load(object sender, System.EventArgs e)
{
```

```
        Label1.Text = Request.QueryString["name"];
        Label2.Text = Request.QueryString["email"];
}
```

۹-۲- استفاده از متغیرهای Session

در این روش باید مقادیر کنترل‌ها را در متغیرهای **Session** ذخیره کنیم و در فرم وب دیگری به آنها دسترسی داشته باشیم. همانطور که می‌دانید ذخیره داده‌های زیاد در **Session** ممکن است اختلالاتی را در سرور بوجود آورد، بنابراین باید از این متد بدرستی استفاده شود. البته هر وقت که خواستید می‌توانید متغیرهای **Session** را از بین ببرید. مراحل اصلی برای استفاده از این متد به ترتیب زیر می‌باشد:

- یک فرم وب با کنترل‌هایش را درست کنید
- یک کنترل دکمه‌ای **Button** یا **LinkButton** برای ارسال فرم به سرور بر روی فرم قرار دهید
- در رویداد کلیک دکمه، متغیرهای **Session** را تعریف کرده و مقادیر کنترل‌ها را در آنها قرار دهید
- کاربر را با استفاده از **Server.Transfer** به صفحه‌ای دیگر هدایت کنید
- در فرم وب دیگر متغیرهای **Session** را دریافت کرده و پس از دریافت اگر لازم باشد آنها را پاک کنید.

کد زیر این مراحل را در عمل نشان می‌دهد:

فرم وب منبع

```
    private void Button1_Click(object sender, System.EventArgs
e)
    {
        //textbox1 and textbox2 are webform
        //controls
        Session["name"] = TextBox1.Text;
        Session["email"] = TextBox2.Text;
        Server.Transfer("anotherwebform.aspx");
    }
```

فرم وب مقصد

```
private void Page_Load(object sender, System.EventArgs e)
{
    Label1.Text = Session["name"].ToString();
    Label2.Text = Session["email"].ToString();
    Session.Remove("name");
    Session.Remove("email");
}
```

۹-۳- استفاده از Server.Transfer

این روش متدی پیچیده ولی روش ماهرانه‌ای برای ارسال مقادیر بین صفحات است. در اینجا مقادیری را که می‌خواهید در صفحات دیگر به آنها دسترسی داشته باشید به عنوان خصوصیات کلاس صفحه بیان می‌کنید. در کل این متد واضحتر و شئ‌گراتر از متدهای قبلی است. مراحل زیر را برای استفاده از این متد بترتیب دنبال کنید.

- یک فرم وب با کنترل‌هایش را درست کنید
- رویدادهای خصوصیت **Get** که مقادیر کنترل‌ها را برخواهند گرداند را تعریف کنید
- یک کنترل دکمه‌ای **Button** یا **LinkButton** برای ارسال فرم به سرور بر روی فرم قرار دهید
- در رویداد کلیک دکمه متد **Server.Transfer** که اجرای برنامه را به فرم تعیین شده انتقال می‌دهد فراخوانی کنید
- در فرم دوم شما می‌توانید با استفاده از خصوصیت **Context.Handler** به یک نمونه از فرم اول دسترسی داشته باشید. سپس می‌توانید از خصوصیات **Get** که برای دسترسی به مقادیر کنترل‌ها ایجاد کرده‌ایم استفاده کنید.

کد زیر برای اجرای یک نمونه از مراحل بالا تدارک دیده شده است:

فرم وب منبع

خصوصیات زیر را به فرم وب اضافه کنید:

```
public string Name
{
    get
    {
        return TextBox1.Text;
    }
}
```

```
    public string EMail
    {
        get
        {
            return TextBox2.Text;
        }
    }
```

حال **Server.Transfer** را فراخوانی کنید.

```
    private void Button1_Click(object sender, System.EventArgs
e)
    {
        Server.Transfer("anotherwebform.aspx");
    }
```

فرم وب مقصد

```
     private void Page_Load
    (object sender, System.EventArgs e)
    {
        //create instance of source web form
        WebForm1 wf1;
        //get reference to current handler instance
        wf1 = (WebForm1)Context.Handler;
        Label1.Text = wf1.Name;
        Label2.Text = wf1.EMail;
    }
```

یک مقاله مرتبط با این مقاله را می‌توانید در آدرس زیر مشاهده فرمایید:

www.iranasp.net/Articles/ShowArticle.aspx?articleid=102

۹-۴- حفظ حالت در ASP.NET

صفحات وب **Stateless** هستند ، یعنی به محض اینکه تولید آن ها در سرور به پایان رسید ، تمام منابع مربوط به آن ها تخریب شده و دیگر سرور هیچ نام و نشانی از آن ها را نمی داند و نگهداری نمی کند . بنابراین نگهداری و حفظ اطلاعات بین رفت و برگشت درخواست ها بین کلاینت ها و سرور اهمیت پیدا می کند . در **ASP** این مورد را با استفاده از **Session , application , query string , cookie** و موارد دیگر می توان حل کرد . در **ASP.NET** نیز هنوز این موارد به قوت خود باقی هستند اما با دیدی قوی تر و کاملتر . در حالت کلی دو روش اصلی برای حفظ حالت صفحات وب در **ASP.NET** وجود دارند (**client side and server side**) که در ادامه به بررسی آن ها خواهیم پرداخت

۹-۵- حفظ حالت سمت کلاینت :

در این حالت هیچ اطلاعاتی بر روی سرور ذخیره نشده و اطلاعات یا بر روی صفحات و یا بر روی کامپیوترهای کلاینت ذخیره می شوند :

- **۹-۵-۱- کوکی ها :**

کوکی ها یا به صورت داده هایی به شکل فایل متنی کم حجم و یا به صورت متغیرهایی ذخیره شده در سشن مرورگر وب کلاینت می باشند . دلیل استفاده ی اصلی از کوکی ها ردیابی اطلاعات است . برای مثال به خاطر سپردن اطلاعات مربوط به لاگین کردن کلاینت :

```
If(Rquest.Cookies[“username”] != null)
lblMessage.text = “Dear” + Rquest.Cookies[“Username”].Value +
“,Welcome shopping here!”;
else
lblMessage.text = “Welcome shopping here ! “;
```

اگر هم قصد ذخیره سازی اطلاعات کلاینت را در یک کوکی دارید می توانید از کد زیر زیر استفاده نمایید :

```
Response.Cookies[“username”].Value = username;
```

- **۹-۵-۲- فیلدهای مخفی :**

می توان کنترل های استاندارد را به کار گرفت و در خاصیت **value** آن ها مقداری را ذخیره نمود . برای این منظور می توان از کنترل **HtmlInputHidden** در **ASP.NET** استفاده نمود .

مثال :

```
Protected System.Web.UI.HtmlControls.HtmlInputHidden Hidden1;
//to assign a value to hidden field
Hidden1.Value = “this is a test”;
//to retrieve a value
String str = Hidden1.Value;
```

دو مورد را باید در این حالت به خاطر داشت :

۱. مقادیر ذخیره شده در این کنترل ها اگر به سورس صفحه ی **HTML** تولیدی مراجعه شود قابل مشاهده هستند . بنابراین اطلاعات حساس را در آن ها ذخیره نکنید .
۲. هنگام استفاده از این کنترل ها فقط متد **post** را برای فرستادن صفحه به سرور می توان به کار گرفت .

- **View State -۳-۵-۹:**

هر کنترل وب **ASP.NET** دارای خاصیت **ViewState** است . بنابراین این کنترل ها به صورت خودکار وضعیت خود را بین رفت و برگشت صفحه بین کلاینت و سرور حفظ می نمایند . همچنین از این خاصیت برای ذخیره سازی اطلاعات بین رفت و برگشت نیز می توان استفاده نمود .

```
//to save information
ViewState.Add("shape" , "circle");
//to retrieve information
String shapes = ViewState['shape"];
```

نکته :

مقادیر ذخیره شده در **ViewState** برخلاف فیلدهای مخفی ، فشرده سازی و رمزگذاری شده هستند بنابراین نگرانی در مورد ذخیره سازی اطلاعات حساس در آن ها وجود ندارد .

برای مثال یک نمونه خروجی **ViewState** در سورس صفحه ی نهایی به صورت زیر می تواند باشد :

```
<input type = "hidden" name="__VIEWSTATE"
value="dDwxNDg5OTk5MzM…"/>
```

- **Query Strings -۴-۵-۹**

مثالی از یک آدرس به همراه یک **query string** می تواند به شکل زیر باشد :

```
http://www.behbazar.com/Articles.aspx?id=1
```

اطلاعات مربوط به **id** را از آدرس فوق به شکل زیر می توان استخراج کرد :

```
String id;
Id = Request.Params["id"];
```

چند نکته راه هنگام کار با **query string** باید در نظر گرفت :

۱. همانطور که ملاحظه کردید مقادیر ذخیره شده در آن ها کاملا در نوار آدرس مرورگر ظاهر می شوند . بنابراین این روش را برای فرستادن اطلاعات حساس نمی توان به کار گرفت .
۲. بعضی از مرورگرها حد ۲۵۵ کاراکتر را برای فرستادن مقادیر فرستاده شده توسط **query string** ها ، در نظر می گیرند .
۳. تنها روش پست کردن **get** برای فرستادن این اطلاعات کارساز است .

۹-۶- حفظ حالت سمت سرور :

در این حالت ها اطلاعات صرفا در سرور ذخیره می شوند و امنیت بسیار بالاتری دارند ، اما منابع بیشتری از سرور را مصرف می کنند .

- **۹-۶-۱- استفاده از شیء Application :**

اطلاعات ذخیره شده در شیء **Application** برای تمام کاربران سایت در سشن های مختلف خودشان قابل استفاده است . بهترین استفاده از آن ها ذخیره سازی متغیرهایی هستند که به ندرت دچار تغییر می شوند و چون بین تمام سشن ها قابل مشاهده و تغییر است باید برای حفظ یکپارچگی آن از متد Lock و UnLock قبل و بعد از قراخوانی آن استفاده نمود :

```
Application.Lock();
Application["mydata"] = "mydata";
Application.UnLock();
```

- **۹-۶-۲- استفاده از شیء Session :**

متغیرهای ساخته شده از شیء سشن تنها برای همان سشن و کاربر قابل مشاهده هستند . از این متغیرها عموما برای انتقال داده ها و متغیرها بین صفحات بدون حفظ حالت وب استفاده می شود .

به صورت پیش فرض اگر کاربر ۲۰ دقیقه با صفحه ی **ASP.NET** کار نکند این متغیرها از طرف سرور تخریب می شوند (برای تغییر این زمان می توانید به **web.config** مراجعه نمایید) .

طرز استفاده از آن ها :

```
//to store information
Session["myname"] = "mohammad";
//to retrieve information
Myname = Session["myname"];
OR
//assign a value to the my variable session variable.
Session["myvariable"] = "simevalue";
//retrieve the value of the myvariable session variable.
String myString;
If(Session["myvariable"] != null)
     myString = (string)Session["myvariable"];
```

- **۹-۶-۳- استفاده از دیتابیس :**

اگر تعداد و میزان اینگونه متغیرها زیاد است بهتر است برای حفظ کارآیی وب سرور آن ها را در دیتابیس ذخیره نموده و سپس بازیابی نمایید .

چگونگی تصمیم گیری بین انتخاب یکی از موارد فوق :

برای انتخاب یکی از موارد فوق می توانید به سوالات زیر پاسخ دهید :

- میزان اطلاعاتی که می خواهید ذخیره کنید چقدر است ؟
- آیا کلاینت کوکی ها را می پذیرد ؟
- می خواهید اطلاعات را روی سرور ذخیره کنید یا کلاینت ؟
- آیا این اطلاعات حساس هستند ؟
- چه کارآیی را از صفحات انتظار دارید ؟

فصل دهم

ارسال ایمیل در ASP.NET

۱۰-۱- ارسال ایمیل در ASP.NET

یکی از کارهایی که انجام آن در برنامه های کاربردی مبتنی بر وب رایج است ، فرستادن ایمیل به مشتریان یا بینندگان سایت می باشد . اغلب ، این نامه ی الکترونیکی به طور اتوماتیک و بدون استفاده از داده های خاص تولید می شود . ولی گاهی اوقات ممکن است اپراتورهای سایت بخواهند یک نامه ی سریع از طریق سایت خود پست کنند . کنترل های وب ، به همراه کلاس های **SmtpMail** و **MailMessage** ، به شما کمک می کنند که این امکانات را خیلی سریع و آسان برای سایت خود فراهم کنید .

توجه :

در سایت بهبازار ما از امکان ارسال ایمیل استفاده می کنیم . و آن هنگامی است که کاربر در سایت ما ثبت نام می کند . همزمان با ثبت نام کاربر در سایت ایمیلی به به آدرس ایمیلی که کاربر در هنگام ثبت نام در اختیار ما قرار داده است ارسال می گردد و محتوای این ایمیل اطلاعات ثبت نام به همراه نام کاربری و کلمه

عبور کاربر خواهد بود . و همچنین از این طریق تاییدی بر ثبت کاربر خواهیم داشت و آن را به اطلاع وی می رسانیم . برای انجام مثال زیر که کدهای سایت بهبازار است شما به یک سرویس دهنده ی **SMTP** دسترسی دارید . **Windows2000 , Windows XP , Windows.NET Server** ، سرویس **SMTP** را به عنوان یکی از سرویس های ارائه شده توسط IIS به همراه دارند . نصب و پیکربندی سرویس **SMTP** نسبتا ساده است . برای کسب اطلاعات بیشتر در مورد چگونگی انجام این کار ، شما می توانید به آدرس http://msdn.microsoft.com/library/en-us/dnduwon/html/d5smtp.asp مراجعه کنید .

برای اینکه امکان ارسال ایمیل را برای سایت خود فراهم کنید ، مراحل زیر را دنبال نمایید (شرح بسیار کاملی در مورد ارسال ایمیل در پیوست آمده است) :

- در اولین قدم ابتدا یک ایمیل در کنترل پنل سایت خود درست کنید . که این کار به سادگی با ورود به کنترل پنل سایت امکان پذیر است و نیازی به توضیح بیشتر در اینجا ندارد . بعد از ایجاد ایمیل نام کاربری و کلمه عبور خود را برای استفاده در مراحل بعدی یادداشت کنید . به طور مثال نام کاربری ما در سایت بهبازار **info@behbazar.com** می باشد . و کلمه عبوری که محفوظ می باشد !
- در دومین مرحله شما نیاز دارید که **MailServer** سایت خود را شناسایی کنید و نام آن را در نظر داشته باشید تا در قدم های بعدی از آن استفاده کنید . این نام را می توانید از خدمات دهنده خود تقاضا کنید و یا اینکه از طریق روش هایی که در پیوست آمده است آن را استخراج کنید . معمولا برای سرورهایی که از **SMTP** برای ارسال ایمیل استفاده می کنند نام سرویس دهنده ایمیل به صورت **mail.domain-name.com** می باشد . در سایت بهبازار این سرور به صورت **mail.behbazar.com** است .
- بعد از انجام مراحل بالا نوبت به آخرین مرحله و مهمترین مرحله می رسد ، یعنی کدنویسی در صفحه ای که قصد ارسال ایمیل از طریق آن را دارید . کدهای زیر طریقه ارسال ایمیل در صفحه ثبت نام کاربران در سایت بهبازار را نشان می دهند :

```
private void SendMail()
{
  try
  {
      MailMessage Mailmsg = new MailMessage();
      Mailmsg.From = new MailAddress("email-address");
      Mailmsg.To.Add(Email);
      Mailmsg.Subject = "Welcome To BehBazar !!!";

      if (Password != null)
```

```
        {
            Mailmsg.Body = "Your Informations : \n";
            Mailmsg.Body += "\n Username : " + Email;
            Mailmsg.Body += "\n Password : " + Password;

            SmtpClient smtp = new SmtpClient("server-name");
            smtp.Credentials = new NetworkCredential("email-
address", "password");
            smtp.Port = 25;
            smtp.Send(Mailmsg);
        }
    }
    catch
    {
    }
}
```

در کد بالا که برای راحتی کار آن را ساده تر کرده ایم ، تابعی به نمایش درآمده است که بوسیله آن برای کاربران سایت پس از ثبت نام ایمیلی ارسال می گردد . همانطور که ملاحظه می کنید **MailMessage** یک **Message** را **New** می کنیم و با این متغیر کار کرده ، اطلاعاتی و متن پیغامی را که می خواهیم از این طریق برای کاربر ارسال نماییم در آن می ریزیم .

پس از این کار با تعریف یک شیء از نوع **SmtpClient** به نام **smtp** یک سرور **SMTP** را برای ارسال ایمیل تعریف می کنیم که همانطور که قبلا به آن اشاره شد نام سرور **SMTP** سایت شما می باشد . بعد از آن نام ایمیل سایت و همچنین کلمه عبور ایمیل را برای احراز هویت توسط سایت مشخص می کنیم .

در مرحله بعدی پورت ارسال ایمیل را که در قراردادهای اینترنت معمولا پورت شماره ۲۵ می باشد تعیین می کنیم . در انتها نیز با دستور **smtp.send(Mailmsg)** ایمیل مورد نظر را ارسال می نماییم . به همین سادگی !

شما می توانید تنظیمات ارسال ایمیل را در فایل **Web.config** نیز انجام دهید تا بدین وسیله دیگر احتیاج به تعریف در تمامی صفحاتی که قصد ارسال ایمیل دارید را نداشته باشید و تنها با یک بار تعریف در **Web.config** می توانید در تمامی صفحات سایت از آن تنظیمات استفاده کنید و تنها کافی است که متن ایمیل ارسالی و پیغام خود را مشخص کنید .

فصل یازدهم

آمار کاربران سایت در ASP.NET

1-11- آمار کاربران سایت در ASP.NET

روزانه کاربران زیادی از سایت شما بازدید می کنند و برای هر مدیر سایت آمار بازدیدها، صفحات بازدید شده، ساعت و تاریخ بازدید، لینک و سایتی که کاربر به واسطه آن از سایت ما بازدید نموده است و . . . از اهمیت ویژه‌ای برخوردار است.

برای آمارگیری از سایت‌ها از روشهای مختلفی می توان استفاده نمود:

۱- استفاده از سایتهای آمارگیری رایگان همانند: **Nedstat** یا **Sitemitter** و . . .
۲- به روش برنامه‌نویسی و بررسی از داخل برنامه

در روش اول به نوعی برای نمایش به کاربران از اعتبار بیشتری برخوردار است اما نقص آن علاوه بر تبلیغ مجانی برای دیگران این است که در بعضی از **Firewall** ها به اسکریپتی که اطلاعات کاربر را به این سایتها ارسال می نماید همانند یک تروجان (اسب تراوا یا سیستم جاسوسی) نگاه می کنند و اجازه عبور به آن نمی دهند.

به همین خاطر روش دوم می تواند دقیق تر عمل نماید. از سوی دیگر ذخیره اطلاعات کاربران در یک بانک اطلاعاتی می تواند منشأ گزارشات بسیار جالبی برای مدیران سیستم شود.

در نمونه برنامه زیر سعی کرده‌ایم اطلاعات نسبتاً جالبی از رفتار کاربران به کمک **ASP.NET** را به نمایش گذاریم. البته مطالب ذیل نمونه‌ای از اطلاعات کاربران سایت می باشد که می شود به سلیقه و نیاز خودتان آنرا تغییر دهید (کلیه متغیرهایی که نوع آنها ذکر نشده است از نوع **String** می باشد).

الف) نام دستگاه کاربر و **IP** آن

```
String ClientName = Request.UserHostName;
String ClientIP = Request.UserHostAddress;
```

ب) لینکی که کاربر با کلیک بر روی آن سایت را یافته است.

```
String REFERER = Request.ServerVariables.Item("HTTP_REFERER");
```

ج) اطلاعاتی در رابطه با سیستم کاربر (Client)

```
String LANGUAGE =
Request.ServerVariables.Item("HTTP_ACCEPT_LANGUAGE");

String AGENT =
Request.ServerVariables.Item("HTTP_USER_AGENT");

String Platform = Request.Browser.Platform();
```

د) QUERY STRING صفحه حاضر

```
QUERY STRING = Request.ServerVariables.Item("QUERY_STRING");
```

ح) اطلاعاتی در رابطه با Browser کاربر

```
String Browser = Request.Browser.Browser();

String Browser_Type = Request.Browser.Type();

String Browser_Version = Request.Browser.Version();
```

ز) زمان و تاریخ بازدید

```
DateTime MyDateTime = new DateTime();

MyDateTime = DateTime.Now;

string MyDate = MyDateTime.ToString("MM/dd/yyyy");
```

```
string MyTime = MyDateTime.ToString("hh:mm:ss");
```

بدیهی است در صورتی که بخواهیم به محض ورود کاربر به سایت این مجموعه اطلاعات را جمع‌آوری نماییم، می بایست آن را در زیربرنامه **Session_Start** انجام دهیم (این روتین در **Global.asax** قرار دارد).
با ارسال محتویات این متغیرها به بانک اطلاعاتی می توان گزارشاتی از جمله تعداد بازدیدها در مقاطع زمانی مختلف، صفحات پر بیننده، پر بیننده‌ترین صفحه امروز و . . . را استخراج نمود

فصل دوازدهم

کار با تصاویر در ASP.NET

- ذخیره تصاویر در **SQL Server**
- کوچک سازی تصاویر در **ASP.NET**

۱۲-۱- ذخیره تصاویر در SQL Server

به طور معمول، تصاویر در پوشه های روی وب سرور ذخیره می شوند نه در دیتابیس، این اما برای فایل های با حجم بالاست. در بعضی موارد ، مثلا یک بانک ، آن ها از تصویر امضای مشتری اسکن می گیرند و آن را در بانک اطلاعاتی خود ذخیره می کنند.

- الگوی بانک اطلاعاتی مورد استفاده : مایکروسافت **SQL Server 2000** را به عنوان بانک اطلاعاتی استفاده می کنیم، از نوع داده ای **image** استفاده خواهیم کرد. نوع داده ای **image** برای ذخیره کردن تصاویر در بانک اطلاعاتی استفاده می شود.

- کنترل هایی که در این برنامه از آن ها استفاده می کنیم عبارتند از :

System.Web.UI.HtmlControls.HtmlInputFile

System.Web.UI.WebControls.TextBox

System.Web.UI.WebControls.Button

- و فضا نام هایی که در این برنامه از آن ها استفاده می کنیم عبارتند از:

```
System.Data.SqlClient

System.Drawing

System.Data

System.IO

System.Drawing.Imaging
```

راه حل

از کلاس **HtmlInputFile** برای ساختن کنترل آپلود فایل استفاده می کنیم. مثال زیر یک فایل **ASPX** کامل است که به کاربر این امکان را می دهد که یک تصویر و توضیح آن را آپلود و به بانک اطلاعاتی اضافه کند . متد **OnUpload** تصویر و توضیح آن را در یک جدول بانک اطلاعاتی **SQL Server** به نام **Pictures** در دیتابیس **MyData** اضافه می کند.

```
// اطلاعاتي سورس كد ذخيره تصوير در بانك

public void OnUpload(Object sender, EventArgs e)
{
    // فايل ورودي از byte[] ساختن يك
    int len = Upload.PostedFile.ContentLength;
    byte[] pic = new byte[len];
    Upload.PostedFile.InputStream.Read (pic, 0, len);

    // اطلاعاتي افزودن تصوير و توضيح تصوير به بانك
    SqlConnection connection = new
      SqlConnection
(@"server=127.0.0.1;database=MyData;uid=sa;pwd=yourpass");
    try
    {
        connection.Open ();
        SqlCommand cmd = new SqlCommand ("insert into Image "
          + "(Picture, Comment) values (@pic, @text)",
connection);
        cmd.Parameters.Add ("@pic", pic);
        cmd.Parameters.Add ("@text", Comment.Text);
        cmd.ExecuteNonQuery ();
    }
    finally
    {
        connection.Close ();
    }
}
```

تابعی که در بالا معرفی کردیم توسط ویژگی **OnClick** یک دکمه فراخوانی می شود.

۱۲-۲- چگونه می توانم یک تصویر را از دیتابیس خوانده و در صفحه وب نمایش دهم؟

از یک صفحه وب برای نمایش تصویر استفاده کرده ام، کد زیر برای نمایش تصویر در صفحه وب استفاده می شود

```
private void Page_Load(object sender, System.EventArgs e)
{
    MemoryStream stream = new MemoryStream ();
    SqlConnection connection = new
      SqlConnection
(@"server=127.0.0.1;database=MyData;uid=sa;pwd=yourpass");
    try
    {
        connection.Open ();
        SqlCommand command = new
          SqlCommand ("select Picture from Image",
connection);
        byte[] image = (byte[]) command.ExecuteScalar ();
        stream.Write (image, 0, image.Length);
        Bitmap bitmap = new Bitmap (stream);
        Response.ContentType = "image/gif";
        bitmap.Save (Response.OutputStream, ImageFormat.Gif);
    }
    finally
    {
        connection.Close ();
        stream.Close ();
    }
}
```

توابع **GDI+** ویژگی های پیشرفته زیادی برای مدیریت و دستکاری داده های تصویری پیشنهاد می کنند . این مثال نگاه سریعی به کارهایی که از طریق فضا نام های **System.Drawing.Imaging** و **System.Drawing** می توانید انجام دهید دارد. به عنوان مثال شما می توانید برنامه را برای ذخیره و مدیریت تصاویر بر روی وب گسترش دهید یا می توانید یک برنامه ساده بنویسید که به کاربر امکان ویرایش و دستکاری تصاویر را می دهد.

چگونه از این کدها استفاده کنیم؟

ابتدا یک دایرکتوری مجازی بسازید و فایل های پروژه را در آن قرار بدهید ، سپس رشته اتصال به بانک اطلاعاتی را تغییر دهید (بر اساس سرور و نام کاربری و کلمه عبور و نام دیتابیس خودتان آن را ست کنید) حالا می توانید از این پروژه استفاده کنید.

۱۲-۳- کوچک سازی تصاویر در ASP.NET

بعضی از مواقع در هنگام طراحی سایت ما نیاز داریم که عکس هایی را از کاربران دریافت کنیم . به عنوان مثال در سایت بهبازار ما برای آگهی های خود از کاربر عکس آگهی را دریافت می کنیم (البته در صورت تمایل کاربر) و در هنگام نمایش آگهی عکس آگهی نیز به نمایش در می آید . اما مشکل زمانی بوجود می آید که عکس های کاربران دارای اندازه های مختلف باشند و همچنین از حجم های زیادی برخوردار باشند . اگر ما بخواهیم عکس ها را به همان نحوی که از کاربران دریافت می کنیم به نمایش درآوریم سایت از قیافه افتاده و شکل ظاهری زننده ای پیدا خواهد کرد و همچنین عکس های با حجم بالا کارآیی سایت را کاهش می دهند .

اما راه حل مناسب برای جلوگیری از مشکلات نامبرده در بالا ، کوچک سازی تصاویر دریافتی می باشد . برای کوچک سازی تصاویر از تابعی استفاده می کنیم که عکس را به تناسب طول و عرضش کوچک کند تا عکس از قواره نیافتد . کدهای زیر این کار را در سایت بهبازار انجام می دهند :

```
 private System.Drawing.Image
CreateThumbnail(System.Drawing.Image src)
{
    int maxSize = 130;

    int w = src.Width;
    int h = src.Height;

    if (w > maxSize)
    {
        h = (h * maxSize) / w;
        w = maxSize;
    }

    if (h > maxSize)
    {
        w = (w * maxSize) / h;
        h = maxSize;
    }

    // The third parameter is required and is
    // of type delegate.  Rather then create a method that
    // does nothing, .NET 2.0 allows for anonymous
```

```
        // delegate (similar to anonymous functions in other
languages).
        return src.GetThumbnailImage(w, h,
               delegate() { return false; }, IntPtr.Zero);
    }
```

تابع **Createthumbnail** این کار را انجام می دهد . در واقع عکسی که کاربر ارسال کرده است به عنوان مقدار ورودی این تابع می باشد و این تابع روی این عکس کار کرده و مقدار کوچک شده عکس را بر می گرداند . متغیر **maxsize** مقدار حداکثر اندازه پهنا و ارتفاع عکس را مشخص می کند تا هنگام کوچک سازی ارتفاع و پهنای عکس از این مقدار فراتر نرود . در انتهای تابع با **return** کردن عکس کوچک شده تابع **CreateThumbnail** به کار خود پایان می دهد .

برای دریافت عکس ارسال شده از کاربر و اعتبارسنجی عکس از کدهای زیر استفاده می کنیم و پس از تایید عکس آن را به تابعی که در بالا ذکر شد ارجاع می دهیم تا عملیات کوچک سازی روی آن انجام گیرد :

```
private bool Pic ()
{
    Boolean fileOK = false;

    PicExtension =
System.IO.Path.GetExtension(FileUploadPic.FileName).ToLower();
     String[] allowedExtensions =
               {".gif", ".png", ".jpeg", ".jpg"};
      for (int i = 0; i < allowedExtensions.Length; i++)
      {
           if (_PicExtension == allowedExtensions[i])
           {
                fileOK = true;
           }
      }

      if (fileOK)
      {

          try
          {
              PicLink = piclink;

              int FileSize =
FileUploadPic.PostedFile.ContentLength;
              FileSize = FileSize / 1024;
              if (FileSize > 80)
              {
          lblSuccess.Text = "حجم عکس باید کمتر از 80 کیلو بایت باشد.";
                  lblSuccess.Visible = true;
                  return false;
```

```
                }

                FileUploadPic.PostedFile.SaveAs(PicLink);


/************************************************/

               using (System.Drawing.Image im =
                System.Drawing.Image.FromFile(_PicLinkTemp))
                using (System.Drawing.Image tn =
                          this.CreateThumbnail(im))
                {
                    if (_PicExtension == ".gif")
                    {
                        tn.Save(_PicLink, ImageFormat.Gif);
                    }
                    else if (_PicExtension == ".png")
                    {
                        tn.Save(_PicLink, ImageFormat.Png);
                    }
              else if (_PicExtension == ".jpeg" || _PicExtension
== ".jpg")
                    {
                        tn.Save(_PicLink, ImageFormat.Jpeg);
                    }
                }

/************************************************/
                File.Delete(_PicLinkTemp);

                PicLink = piclink;

                return true;
            }
            catch
            {
                lblSuccess.Text = "باشد نمی مقدور عکس آپلود.";
                lblSuccess.Visible = true;
                return false;
            }
        }
        else
        {
            lblSuccess.Text = "باشد نمی مجاز عکس نوع.";
            lblSuccess.Visible = true;
            return false;
        }
}
```

در این کدها ابتدا پسوند عکس مورد آزمایش قرار می گیرد و در صورت مجاز بودن پسوند کار ادامه می یابد و در صورتی که پسوند ارسال شده توسط کاربر غیر مجاز باشد پیغام مبنی بر آن به کاربر داده می شود و از وی تقاضای ارسال مجدد عکس با پسوند مجاز می شود . پس از آن حجم عکس مورد تست واقع می شود و در صورتی که حجم عکس بیش از ۸۰ کیلوبایت باشد از کاربر تقاضای کاهش حجم عکس و ارسال مجدد را می کند و اما در صورت صحیح بودن این مورد آنگاه عکس به تابع بالایی یعنی **CreateThumbnail** فرستاده می شود تا عملیات کوچک سازی در آن انجام شود .

برای دریافت عکس از کاربر از کنترل **FileUpload** استفاده می شود . این کنترل که شکل آن به صورت زیر می باشد دارای یک دکمه به نام **Browse** است که از طریق آن آدرس عکس در سیستم کاربر را دریافت می کند و پس از زدن دکمه تایید آدرس در **TextBox** روبروی دکمه ی **Browse** به نمایش درمی آید .

شکل ۱۲-۱- کنترل **FileUpload**

این کنترل دارای خواص زیادی است که از طریق آن می توان به اطلاعات زیادی از عکس دست یافت و از آن ها در کدهای خود استفاده کرد . مانند یافتن پسوند فایل ارسال شده ، حجم فایل ، اندازه پهنا و ارتفاع عکس و ...

فصل سیزدهم

امنیت در ASP.NET

- استفاده از فایل **Web.config**
- تعیین اعتبار ورودی کاربر
- جلوگیری از حملات اسکریپتی
- اطمینان نکردن به کنترل های تعیین اعتبار
- محافظت در برابر **SQL Injection**
- ساخت تصاویر امنیتی
- کد کردن **Connection String**

۱۳-۱-امنیت در ASP.NET

ASP.NET با فراهم آوردن زیر ساخت محافظت از دسترسی غیر مجاز به صفحات مختلف وب ، امکان برنامه نویسی امنی را به سادگی فراهم آورده است .

سه لایه امنیتی جهت برنامه های **ASP.NET** قابل تعریف هستند : لایه **IIS** ، لایه **ASP.NET Worker Process** و لایه **Application** . به عنوان برنامه نویس ، می توان بر روی دو لایه ی اول نیز تاثیر گذار بود و آن ها را تنظیم نمود ، اما بیشترین مسئولیت یک برنامه نویس وب ، متوجه لایه ی سوم اشاره شده می باشد و این لایه ی امنیتی محور اصلی بحث ماست .

مقابله با خطاها و اهمیت آن ها در تأمین امنیت سایت

محیط **.NET** ، در هنگام رخ دادن خطایی در برنامه ، اطلاعات قابل توجهی را جهت **debug** کردن بهتر کد ارائه می دهد ، اما یک مهاجم با سعی و خطا و بررسی نتایج حاصل ، می تواند اطلاعات قابل توجهی را از برنامه ، **server** مورد استفاده و سایر موارد مربوطه به دست آورد .

روش مطمئن برخورد با خطاها ، همواره باید با ارائه ی جمله ی " خطایی در برنامه رخ داده است " خاتمه یابد . به عنوان مثال در یکی از صفحات سایت بهبازار از کد زیر برای این منظور استفاده شده است :

```
try
        {
```

```
            //do something
        }
        catch
        {
            Response.Redirect("Cost.aspx");
            lblSuccess.Text = " مشکلي در ذخيره اطلاعات بوجود آمده
است";
            lblSuccess.Visible = true;
        }
```

در مثال فوق اگر برنامه با خطا مواجه شود تنها یک جمله ساده به کاربر نمایش داده می شود .

۱۳-۲- استفاده از فایل Web.config

جهت نمایش صفحه ای عمومی به جای جزییات خطای رخ داده ، می توان از تنظیمات فایل **web.config** بدون هیچ گونه برنامه نویسی خاصی نیز استفاده نمود . شکل عمومی تگ های آن به صورت زیر می باشد .

```
<customErrors mode="on|off|RemoteOnly" defaultRedirect="url">
    <error statusCode="403" redirect="NoAccess.htm" />
    <error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```

در حالتی که مقدار **mode** به **off** تنظیم شود ، جزییات خطا به صورت کامل نمایش داده خواهد شد . اگر مقدار **mode** به **on** تنظیم شود ، نمایش خطاهای سفاری فعال شده و به صفحه ای از پیش تعیین شده هدایت خواهند شد . و اگر مقدار **mode** به **RemoteOnly** تنظیم گردد ، یک صفحه خطای عمومی به کاربر راه دور نمایش داده شده افرادی که دسترسی به server داشته باشند ، با مرور صفحه ی مربوطه به صورت **local** ، می توانند خطا را به صورت کامل مشاهده نمایند (حالت پیش فرض و بهترین حالت از لحاظ امنیتی) .

همچنین حالت **compile** برنامه را می توان در فایل **web.config** به صورت زیر نیز تنظیم نمود (حتما هنگام ارائه برنامه ی نهایی ، باید مقدار آن به **false** تنظیم شود) .

```
<compilation   debug="true"/>
```

بنابراین بهترین حالت جهت ارائه برنامه ، افزودن و تنظیم دو خط زیر در فایل web.config برنامه است :

```
<trace  enabled = "false" />
<compileation  debug = "false" />
```

۱۳-۳- تعیین اعتبار ورودی کاربر

در صورتی که قسمت های دریافت اطلاعات در سایت و محتوای ورودی آن ها به درستی مورد بررسی قرار نگیرند ، این مکان ها تبدیل به نقاط اصلی ضعف امنیتی برنامه شما خواهند شد. این خطرات شامل امکان اجرای دستورات بر روی **server** ، دسترسی به اطلاعات حساس و امثال آن می باشند . همان طور که ذکر گردید ، ریسک عدم بررسی ورودی کاربر به شدت بالا بوده و دلایل زیر را جهت آن باید در نظر گرفت :

- روش های بسیاری جهت تخریب اطلاعات یک برنامه در اثر عدم بررسی دقیق ورودی کاربر وجود دارد .
- نیازی به داشتن کلمه ی عبور جهت دسترسی به اطلاعات حساس نخواهد بود .
- مهاجم می تواند به هر آن چه برنامه به آن ها دسترسی دارد ، دست پیدا کند .
- اغلب برنامه های تحت وب جهت برخورد با اینگونه خطرات به درستی تنظیم نشده اند .

۱۳-۴- مدیریت ورودی های خطرناک مهاجمان

جلوگیری از حملات اسکریپتی (**Cross Site Scripting Attacks**)

کد **ASP.NET** زیر را در نظر بگیرید :

```
<script language=c# >
     Response.write("Hello, " + Request.QueryString("name"));
</script>
```

در ظاهر ، کد فوق هیچ گونه مشکلی ندارد و طریقه ی کار و دستیابی به آن نیز همانند آدرس زیر می باشد :

```
http://www.behbazar.com/welcome.aspx?name=Mohammad
```

از طریق QueryString مقدار name دریافت شده و سپس نمایش داده می شود ، اما به کد زیر توجه نمایید :

```
http://www.behbazar.com/welcome.aspx?name=<script>alert('hi
!') ; </script>
```

کد فوق مخرب نیست و تنها پس از درخواست آدرس فوق به شکل داده شده ، یک **Alert** که **Hi** را نمایش می دهد ظاهر می گردد ، اما تخریب از جایی شروع می شود که کاربری با دادن یک چنین لینکی در سایت محتویات کوکی مدیر سیستم را دزدیده و به آدرسی در سایتی دیگر منتقل و ثبت کند (از کوکی ها عموما برای ثبت اطلاعات **Login** افراد استفاده می شود و به این صورت جعل خواهد شد) .

دو روش برای جلوگیری از این امر وجود دارد :

- استفاده از **Regular Expression** ، جهت بررسی ورودی کاربر (آیا واقعا کاربر یک اسم که حاوی حروفی مشخص است را وارد کرده یا خیر) . برای مثال :

```
Regex r = new Regex(@"^[\w]{1,40}$");
If (r.Match(strName).Success)
{
     //Cool ! the string is OK
}
Else
{
     // Not Cool @ Invalid string
}
```

کد فوق بررسی می کند ، که آیا ورودی کاربر حداکثر به طول ۴۰ است یا خیر و آیا تنها حاوی حروف و اعداد مجاز می باشد یا خیر ؟ (برای مثال < در ابتدای اسکریپت قابل قبول نخواهد بود) .

- استفاده از **HttpServerUtility.HtmlEncode** و **UrlPathEncode** در **ASP.NET** و یا **Server.HTMLEncode** در **ASP** ، برای تبدیل ورودی خطرناک به خروجی مطمئن .

در **ASP.NET** عموما بررسی ورودی خطرناک (هر نوع کد **HTML**) وارد شده توسط کاربر نیز به صورت توکار صورت می گیرد .

علاوه بر موارد فوتق **Microsoft Anti-Cross Site Scripting Library** نیز از سایت **Microsoft** در این زمینه قابل دریافت است .

۱۳-۵- اطمینان نکردن به کنترل های تعیین اعتبار در ASP.NET

در **ASP.NET** یک سری کنترل تعیین اعتبار مانند **Required Field Validator** , **Regular Expression** و امثال آن وجود دارند . این نوع کنترل ها جهت بازدارندگی کاربرهای معمولی بسیار مفید بوده و در جهت بالابردن سرعت تولید یک برنامه بسیار مفید هستند ، اما حداقل دو راه برای از کار انداختن آن ها وجود دارد .

۱. خاموش کردن پردازش اسکریپت ها در مرورگر وب (خصوصا در فایرفاکس) .
۲. جعل کردن صفحه ی پویای وب (ذخیره کردن خروجی صفحه بر روی کامپیوتر و سپس حذف اسکریپت های آن و اصلاح مسیر صفحه ای که باید اطلاعات به آن **post** شود) .

بنابراین اطمینان مطلق به کنترل های تعین اعتبار **ASP.NET** در اینترنت ، بسیار کار نابجا و خطرناکی است ! بنابراین هنگام دریافت اطلاعات از کاربر ، علاوه بر استفاده از کنترل **validator** سمت **client** باید در سمت **server** نیز ورودی کاربر به صورت دقیق بررسی گردد .

۱۳-۶- محافظت از برنامه های خود در مقابل SQL Injection

تزریق **SQL** تکنیکی است ، که مهاجم را قادر می سازد ، تا دستورات **SQL** غیر مجازی را با بهره گیری از ضعف چک نکردن داده های ورودی توسط برنامه نویس ، که از این ورودی ها در عبارات **SQL** پویای خودش استفاده می نماید ، روی بانک اطلاعاتی اجرا کند .

اولین بار این موضوع توسط **David Litchfield** ، در مقاله ای با نام **Web Application with ODBC Eroor Message Disassembly** ، گوشزد گردید .

مثال اول ، چگونگی انجام تزریق **SQL** در یک صفحه ی **Login**

عموما صفحات ورود کاربر ، از عبارات پویای **SQL** برای چک کردن این امر که آیا مشخصات کاربر در بانک اطلاعاتی موجود است یا خیر ، استفاده می نمایند . در ادامه ، مثالی متداول در این زمینه را ملاحظه می فرمایید .

در اینجا **UserID** و **Password** کاربر دریافت شده و سپس توسط عبارت **SQL** ساخته شده بررسی می شود ، که آیا رکوردی مربوط به این کاربر درون جدول **Users** وجود دارد یا خیر ؟ اگر پاسخ مثبت باشد ،

به صفحه ی **loginsucceeded.aspx** هدایت شده و در غیر این صورت صفحه ی **loginfaild.aspx** نمایش داده می شود .

```
SELECT * FROM Users
WHERE User name = '<field from web form>'
AND Password = '<field from web form>'
IF [Stuff isReturned] { Login looks good}
ELSE {Login looks bad}
```

در نگاه اول کد فوق ، کدی است عدی و هیچ گونه مشکلی خاصی ندارد .

اما اگر کاربر به جای **id** ، **hi' or 1=1–** و به جای پسورد نیز همان را وارد نماید (البته الزامی نیست) به راحتی از صفحه ی **Login** با موفقیت رد خواهد شد ، چرا ؟

در حالت عادی ما انتظار داریم ، که با ورودی کاربر ، عبارت **SQL** هایی شبیه به عبارت زیر ایجاد گردد :

```
SELECT * FROM Users WHERE username = 'mohammad' AND password
= 'jafari'
```

اما در انیجا کاربر ، عبارت زیر را برای ما ایجاد کرده است :

```
SELECT * FROM Users WHERE username = 'hi' or 1=1–' AND
password = 'hi' or 1=1–'
```

همان طور که می دانید در عبارت **SQL** ، رشته ها با **single quote** خاتمه می یابند و استفاده از – در انتهای عبارت وارد شده نیز ایجاد خطا به دلیل استفاده از **quote marks** بسته نشده را در **SQL** منتفی می کند . در زبان **C#** برای نوشتن توضیحات مربوط به کدها از **/**** یا **//** استفاده می شود و در زبان **SQL** از – (در حقیقت قسمت بعدی عبارت **SQL** نادیده گرفته خواهد شد) .

و در عبارت فوق به دلیل استفاده از **or** و با توجه با اینکه همواره **1=1** می باشد ، رکوردهای مورد نظر مهاجم ایجاد گردیده عبارت **SQL** حداقل یک رکورد را برمی گرداند (در اینجا تمام رکوردها) و شرایط آن تعبیر به **True** خواهد شد و در نهایت صفحه ی **login** پشت سر گذاشته می شود .

۱۳-۷- روش محافظت در برابر تزریق عباراتی که از **or** و **single quote marks** استفاده می کنند

الف) استفاده از رویه ی ذخیره شده استاندارد **sp_audit** در **SQL Server** می باشد . در اینجا اگر مهاجم روش قبل را در پیش گیرد ، با صفحه ی خطای زیر مواجه خواهد گردید و دیگر عبور از صفحه ی **LOGIN** با اینگونه تزریق ها میسر نمی گردد .

ب) استفاده از تابع **REPLACE()** در **ASP**

مثال دوم ، تزریق **SQL** در **Query String**

لینک زیر را در نظر بگیرید :

```
http://www.behbazar.com/article.aspx?ID=100
```

استفاده از **query string** ، روشی است بسیار متداول در تمام زبان های برنامه نویسی سمت **Web Server** ، فرض کنید می خواهید مقالات سایت خود را به صورت پویا نمایش دهید . در یک صفحه عنوان تمام مقالات را از بانک اطلاعاتی خوانده و به صورت لینک هایی که انتهای آن ها حاوی **query string** ای که بیانگر شماره ی منحصر به فرد این مقاله در بانک اطلاعاتی است ، در می آورید . هنگامی که کاربر روی عنوان یک مقاله کلیک می کند به صفحه ی **article.aspx** هدایت شده و در این صفحه قبل از هر چیزی مقدار **id** دریافت می شود ، سپس به صورت پویا محتویات این مقاله از بانک اطلاعاتی خوانده شده و نمایش داده می شود . بنابراین عبارت **SQL** ما شبیه به کد زیر می باشد :

```
strSQL = "SELECT * FROM  tblArticle  WHERE  ID = " + ID ;
```

فرض کنید مهاجم به جای لینکی عادی ، لینک زیر را در مرورگر وارد نماید :

```
 or 1=1http://www.behbazar.com/article.aspx?ID=0
```

در این حالت عبارت پویای SQL ما به شکل زیر در خواهد آمد :

```
SELECT * FROM  tblArticle  WHERE  ID = 0 or 1=1
```

این عبارت ، تمام مقالات موجود را برمی گرداند . البته این مورد شاید مطلب خطرناکی به نظر نرسد ، اما لینک زیر را در نظر بگیرید .

```
http://www.behbazar.com/article.aspx?ID=100 ; DELETE FROM
tblArticles
```

خودتان عبارت پویای **SQL** نهایی را حدس بزنید ! لینک فوق سبب می شود کل محتویات جدول مقالات بدون هیچ گونه خطایی و کاملا مطابق با دستورات **SQL** حذف شوند .

۱۳-۸- روش محافظت در برابر تزریق SQL در Query String

در کد قبل ما انتظار داریم که **ID** دریافت شده از طریق **Query String** یک عدد باشد . پس بهتر است که این مورد را قبل از استفاده از آن بررسی نماییم . در زبان هایی مانند **C#** که به تعریف داده ها بسیار مقید هستند ، خود به خود مسئله حل شده است. چون برنامه نویسی نمی تواند بدون تعریف نوع داده از آن استفاده نماید و برای مثال اگر داده ی ورودی به متغیر تعریف شده از نوع عددی ، یک رشته باشد ، خود به خود برنامه متوقف می گردد .

۱۳-۹- انتخاب پسوردهای قوی

برای امنیت پسوردهای کاربران باید آن ها را وادار کرد که از پسوردهای قوی استفاده نمایند تا از شر برنامه ی های **hack** توسط **dictionary** خلاص شویم . در **ASP.NET 1.x** روشی توکار جهت اعمال سیاست های پسورد وجود ندارد . اما به سادگی با استفاده از **regular expression** می توان این بررسی را انجام داد . در **ASP.NET 2.x** و اضافات جدید آن مانند مبحث **membership** ، امکان اعمال این گونه سیاست ها به صورت استاندارد مهیا گردیده است ، که در ادامه به آن خواهیم پرداخت .

وادار کردن کاربران به انتخاب پسوردهای قوی در ASP.NET 2.x

بدیهی است روشی که در **ASP.NET 1.x** به کار گرفته می شود در این نگارش نیز قابل استفاده است ، اما با استفاده از مبحث جدید **membership** می توان بدون نوشتن حتی یک خط کد تمامی موارد فوق را اعمال نمود .

یکی از موارد امنیتی جدید اضافه شده به **ASP.NET 2.0** ، بحث **membership** و ارائه یک راه حل استاندارد و بسیار پخته جهت بحث اعتبارسنجی و ثبت نام کاربران در سایت است . پس از اضافه کردن تنظیم های استاندارد **membership** به فایل **web.config** ، خط زیرا را در آن به صورت ارائه شده تنظیم نمایید :

```
PasswordStrengthRegularExpression =
"(?=.{8,}) [a-z] + [^a-z] + | [^a-z] + [a-z] + "
```

همان طور که ملاحظه می کنید ، با کمک **regular expression** ، پسورد انتخابی کاربر بررسی شده و این پسورد باید حداقل ۸ کاراکتر حداقل دارای یک حرف کوچک و حداقل دارای یک کاراکتر که جزء حروف کوچک نیست ، تشکیل شده باشد . تمام این موارد ، بدون اینکه حتی یک خط کد اضافی بنویسید برای شما انجام خواهد شد .

سایر مواردی که می توان فقط با یک تنظیم ساده در **web.config** در قسمت **membership** انجام داد :

passwordFormat : آیا پسورد **hash** شده در بانک اطلاعاتی ذخیره شود یا خیر .

minRequirePasswordLength : حداقل طول پسورد مجاز .

و امثال آن .

۱۳-۱۰- آموزش ساخت تصاویر امنیتی

تصویر امنیتی چیست ؟ احتمالا شما در هنگام دانلود فایل یا هنگام ثبت نام در برخی از سایت ها درگیر پر کردن فرم های کوچک و بزرگی شده اید. در این فرم ها احتمالا با یک تصویری که در آن چند کاراکتر و یا عدد به شکلی سوال انگیز در کنار هم قرار گرفته است برخورد کرده اید.در کنار این تصاویر از شما خواسته شده است تا کاراکترهائی را که در تصویر مشاهده می کنید را در یک **TextBox** وارد کنید. اما این تصاویر به چه دردی می خورند ؟

کاربرد تصاویر امنیتی ؟ تصور کنید در سایت شما بخش نظرسنجی وجود دارد و در آن هر فرد مراجعه کننده به سایت (**Anonymous**) می تواند نظر خود را در آن وارد کند. شما چند راه برای این کار دارید یا تعداد نظراتی را که هر فرد می تواند ثبت کند را محدود سازید و یا اینکه هیچ محدودیتی وجود نداشته باشد. در صورتی که قصد محدود سازی کاربر را داشته باشید احتمالا به سراغ متغیر **Session** می روید و با الگوریتم خاص خود این مهم را انجام می دهید. مثلا کاربر را مجبور می کنید تا فقط در بازه های زمانی ۱۰ دقیقه ای امکان ثبت را داشته باشد ! سناریو ای را در نظر بگیرید که چند کاربر که از یک **PC** استفاده می کنند و یا کاربری قصد ارسال دو نظر را دارد، احتمالا با خواندن سناریوهائی از این دست به شما عذاب

وجدان دست خواهد داد. حال فرض کنیم شما راه حل دوم را انتخاب کنید و هیچ محدودیتی برای ورود اطلاعات در نظر نگیرید. سناریوای را در نظر بگیرید که در آن هکری با نوشتن یک برنامه ساده سعی می کند به سرعت در وب سایت شما مدام نظر ثبت کند. اگر هر رکورد جدول نظر شما ۲۰۹۲ بایت اشغال کند (۲۰۴۸ بایت برای نظر) و در هر ۳ ثانیه (بهترین حالت برای شما و بدترین حالت برای هکر) بتواند یک رکورد ثبت کند. اگر بانک اطلاعاتی سایت شما **mb**۱۰ فضای آزاد داشته باشد در عرض ۴ ساعت بانک اطلاعاتی شما پر از اطلاعات هرزه می شود. احتمالا در بهترین حالت بین ۳ تا ۴ روز بانک شما پر است و تا آن زمان هیچ چیزی نمی تواند در بانک اطلاعاتی شما درج شود و چندین ساعت از وقت شما صرف پاک کردن اطلاعات هرزه می شود.

نحوه ساخت تصاویر امنیتی

روشی که در این قسمت برای ساخت تصاویر امنیتی استفاده شده است بدین شرح است که از طریق ایجاد یک لینک به **HttpHandler** تصویر امنیتی را نمایش می دهیم.

```
SecurityLevel.Low</td>
<td style="width: 100px">
<img id="iLow" src="si.ashx?YlQyc0MwSHJ1VFklM2QmMzAwJjUwJkxvdw%3d%3d"
style="border-color:Black;border-width:1px;border-style:Solid;" /></td>
</tr>
```

همان طور که در تصویر مشاهده می شود از طریق پسوند **ashx** به **HttpHandler** تولید کننده تصویر امنیتی متصل شده و تصویر امنیتی را دریافت می کنیم. این تصویر امنیتی بر اساس مشخصات مورد نظر ما ایجاد می شوند. این مشخصات در ادامه ی پسوند **ashx** به عنوان **Query** و به صورت کاملا رمزنگاری شده تولید شده اند.

شکل ۱۳-۱- ساخت تصویر امنیتی

کلاس SecurityImage

این کلاس وظیفه تولید آدرس یک تصویر امنیتی را دارا می باشد. در حقیقت ما فقط از این کلاس استفاده می کنیم و خود را از درگیری با سایر مسائل رها می سازیم. به مثال زیر توجه فرمائید :

```
if (!this.IsPostBack)
        {
            SecurityImage simg = new
SecurityImage(SecurityLevel.Low, 300, 50);
```

```
            Session["simg_code"] = simg.Code;
            simg.Generate();
            iLow.ImageUrl = simg.SecurityImageUri;
        }
```

پارامترهای ورودی سازنده کلاس **Security Image** سطح امنیتی کد و اندازه تصویر خروجی را تعیین می کنند. متد **Generate** با هر بار فراخوانی یک کد جدید تولید می کند و به طبع آن **SecurityImageUri** و **Code** نیز تغییر می کند.

این کد را در هنگام **Load** شدن صفحه به کار گرفته و کد امنیتی ایجاد شده را در **Session** نگه داری می کنیم. بعد از اینکه کاربر فرم مورد نظر را پر کرده و پست می کند. می توان مقدار تایپ شده توسط کاربر را با اصل کد امنیتی موجود در **Session** چک کرده و در صورت صحت آن اطلاعات را ثبت کنیم.

مهمترین متد کلاس **SecurityImage** به شرح زیر است :

```
    private string GenerateCode()
    {
        Random r = new Random((int)DateTime.Now.Millisecond);
        int maxNumber = 0;
        int numberCodeQuantity = 0, alphabeticCodeQuantity =
0;
        string code = String.Empty;

        maxNumber = r.Next(6, 7);
        numberCodeQuantity = 1;
        if (SecurityLevel == SecurityLevel.Low)
            numberCodeQuantity = 5;
        else
            alphabeticCodeQuantity = maxNumber -
numberCodeQuantity;

        for (int i = 1; i <= numberCodeQuantity; ++i)
        {
            code += ((char)r.Next(48, 57)).ToString();
        }
        for (int i = 1; i <= alphabeticCodeQuantity; ++i)
        {
            code += ((char)r.Next(65, 90)).ToString();
        }
        return code;
    }
```

این متد عمل تولید کد امنیتی متناسب با سطح امنیتی تعیین شده انجام می دهد. پس از تولید کد باید اطلاعات شامل اندازه تصویر، سطح امنیتی تصویر و کد مورد نظر کد شده و به صورت یک **Query** درآیند.

```
     Public string Generate()
    {
        byte[] IV = new byte[8] { 120, 34, 63, 127, 93, 240,
23, 232 };
        string cryptoKey = "GalaxyRoad2004@yahoo.com";

        _code = GenerateCode();
        byte[] codebytes =
System.Text.Encoding.ASCII.GetBytes(_code);
        TripleDESCryptoServiceProvider tripleDES = new
TripleDESCryptoServiceProvider();
        MD5CryptoServiceProvider md5 = new
MD5CryptoServiceProvider();
        tripleDES.Key =
md5.ComputeHash(System.Text.Encoding.ASCII.GetBytes(cryptoKey)
);
        tripleDES.IV = IV;
        byte[] codeBuffer =
tripleDES.CreateEncryptor().TransformFinalBlock(codebytes, 0,
codebytes.Length);
        string secretCode = Convert.ToBase64String(codeBuffer,
0, codeBuffer.Length);
        secretCode =
HttpContext.Current.Server.UrlEncode(secretCode);
        string query = secretCode + "&" + Width.ToString() +
"&" + Height.ToString() + "&" + SecurityLevel.ToString();
        byte[] queryBytes =
System.Text.Encoding.ASCII.GetBytes(query);
        Random r = new Random((int)DateTime.Now.Millisecond);
        _simgUri = r.Next().ToString() + ".ashx?" +
HttpContext.Current.Server.UrlEncode(Convert.ToBase64String(qu
eryBytes, 0, queryBytes.Length));
        return _simgUri;
    }
```

متد فوق این کار را برای ما انجام می دهد و اطلاعات مورد نظر را به صورت یک **Query**، رمز می کند و آن را در قالب یک **Uri** فایل تصویری **jpg** باز می گرداند.

در سمت سرور ما کلاس **SecurityImageHandler** را داریم که وظیفه ی **Handler** کردن درخواست های تصاویر امنیتی را بر عهده دارد. مهمترین متد این کلاس به شرح زیر است :

```
     public void ProcessRequest(HttpContext context)
    {
        string secretString;
        string code;
        try
        {
```

```
            secretString =
context.Request.Url.Query.Substring(1);
            code = Decode(secretString);
        }
        catch (Exception)
        {
            return;
        }
        Size size = new Size(300, 50);
        Bitmap bmp = new Bitmap(size.Width, size.Height);
        Bitmap result = new Bitmap(imageWidth, imageHeight);
        Graphics g = Graphics.FromImage(bmp);
        …
        result.Save(context.Response.OutputStream,
System.Drawing.Imaging.ImageFormat.Jpeg);
    }
```

برای اختصار از آوردن کدهای تولید تصویر خودداری کردم. قبل از تولید تصویر کد امنیتی باید **Query** ارسال شده **Decode** گردد که این عمل توسط متد زیر صورت می گیرد :

```
    private string Decode(string secretCode)
    {
        byte[] queryBytes =
Convert.FromBase64String(HttpContext.Current.Server.UrlDecode(
secretCode));
        string query =
System.Text.Encoding.ASCII.GetString(queryBytes, 0,
queryBytes.Length);
        string[] parameters = query.Split('&');
        imageWidth = int.Parse(parameters[1]);
        imageHeight = int.Parse(parameters[2]);
        ParseSecurityLevel(parameters[3]);
        byte[] codeBuffer =
Convert.FromBase64String(HttpContext.Current.Server.UrlDecode(
parameters[0]));
        byte[] IV = new byte[8] { 120, 34, 63, 127, 93, 240,
23, 232 };
        string cryptoKey = "GalaxyRoad2004@yahoo.com";
        TripleDESCryptoServiceProvider tripleDES = new
TripleDESCryptoServiceProvider();
        MD5CryptoServiceProvider md5 = new
MD5CryptoServiceProvider();

        tripleDES.Key =
md5.ComputeHash(System.Text.Encoding.ASCII.GetBytes(cryptoKey)
);
        tripleDES.IV = IV;
        byte[] codebytes =
tripleDES.CreateDecryptor().TransformFinalBlock(codeBuffer, 0,
codeBuffer.Length);
```

```
        return System.Text.Encoding.ASCII.GetString(codebytes,
0, codebytes.Length);
    }
```

نحوه استفاده از SecurityImage تهیه شده در این بخش :

کد زیر را در **PageLoad** قرار دهید :

```
if (!this.IsPostBack)
        {
            SecurityImage simg = new
SecurityImage(SecurityLevel.Low, 300, 50);
            Session["simg_code"] = simg.Code;
            simg.Generate();
            iLow.ImageUrl = simg.SecurityImageUri;
        }
```

کد زیر را در هم برای چک کردن اینکه آیا کاربر کد را صحیح وارد کرده است یا خیر استفاده نمائید :

```
    protected void Check_Click(object sender, EventArgs e)
    {
    bool correct = false;
        if (Session["simg_code
"].ToString().ToLower().Equals(tbCode.Text.ToLower()))
            correct = true;

    if (correct) // Success
        Do some thing
    Else // Failed
        Response.Write("Failed.");
    }
```

در فایل **web.config** مابین تگ های <**system.web**> کد زیر را درج نمائید :

```
<httpHandlers>
    <add path="*.ashx" verb="*"
type="Farahy.Security.SecurityImageHttpHandler,
SecurityImage"/>
</httpHandlers>
```

و در نهایت مطمئن شوید که فایل **securityimage.dll** را در شاخه **bin** وب اپلیکیشن کپی کرده اید.

۱۳-۱۱- کد کردن Connection String

اطلاعات **Connection String** در پروژه های **Web Based** از مهمترین مواردی است که برای مخفی ماندن آن از دید دیگران باید اهتمام ویژه ای به خرج داد. برای حفظ امنیت این اطلاع در **ASP.NET** راه کارهای مختلفی وجود دارد که در اینجا به یکی از آنها اشاره می کنیم که کد کردن **Connection String** و سپس قرار دادن آن در **web.config** و یا متن برنامه است.

شاید این روش در نگاه اول روش بسیار امنی به نظر بیاید اما در واقع این طور نیست زیرا به راحتی **Decode** می شود ولی به هر حال مسکنی موقت است.

برای کد کردن یک رشته کاراکتری ابتدا لازم است **System.Text** را **using** کنید.

```
Using System.Text;
```

سپس میتوانید به وسیله چند خط زیر یک رشته کاراکتری را که می تواند همان **Connection String** باشد کد کنید

```
string strConnectionString =
    "server=127.0.0.1;database=db;uid=user;pwd=pass";
    string strToEncode =
        Convert.ToBase64String
        (ASCIIEncoding.ASCII.GetBytes(strConnectionString));
```

مقدار کد شده در متغییر **strToEncode** قرار گرفته است . این مقدار کد شده را در جائی مثل **web.config** قرار دهید. حال زمانی که در برنامه می خواهید **Connection String** را به یک **SqlConnection** بدهید تا آن را **Open** کند، باید آن را **Decode** کنید. برای این کار می توان تابع زیر را نوشت که مقدار کد شده را به عنوان پارامتر می گیرد و مقدار **Decode** شده را بر می گرداند

```
string Decoder(string strToEncode)
{
    return
        ASCIIEncoding.ASCII.GetString
            (Convert.FromBase64String(strToEncode));
}
```

فصل چهاردهم

مقابله با خطاها

- اشکالزدایی خطاهای سایت
- کدنویسی تدافعی
- مقابله با استثناء ها

۱۴-۱- مقابله با خطاها

در این فصل به بررسی موضوعی می پردازیم که در طول ایجاد سایت باید آن را نیز مد نظر قرار دهید و آن نحوه ی مقابله با خطاهاست (**Dealing With Errors**) .

این یک واقعیت است که در هنگام ایجاد برنامه ، به خطاهایی برخورد می کنید . از آن جا که هر کسی ممکن است دچار اشتباه شود ، بنابراین این موضوع چیزی نیست که از آن شرمنده بوده و نگرانش باشید . برخی از این اشتباهات ساده تایپی هستند و برخی دیگر اشتباهات پیچیده تری می باشند که ممکن است بدلیل نداشتن تمرین کافی باشد . بنابراین آن چه که در این فصل مورد بحث قرار می گیرد ، کلیه جنبه های اداره خطا را می پوشاند .

۱۴-۲- کدنویسی تدافعی

کد نویسی تدافعی (**Defensive coding**) کلا به پیش بینی باز می گردد . کار بر روی آن چه که ممکن است اتفاق بیافتد و نوشتن کدهایی که از بروز آن ممانعت کنند . یکی از قوانین کدنویسی تدافعی این است که هرگز نباید چیزی را تصور کنید خصوصا اگر کاربر باید اطلاعاتی را وارد نماید . اکثر کاربران بسیار خوشحال می شوند ، تا بتوانند به همان صورتی که از سایت انتظار دارند استفاده کنند ، اما هکرها (**Hackers**) نیز همواره به دنبال راه نفوذی برای ورود به سایت ها جهت انجام اعمال خرابکارانه هستند . بنابراین باید هر کاری انجام دهید ، تا این خطر را به حداقل ممکن کاهش دهید .

مورد هک واقع شدن ، تنها دلیل برای کدنویسی تدافعی نمی باشد . اشتباهات کدنویسی یا آسیب پذیری آن ممکن است توسط کاربران پیدا و مشخص نگردد ، اما توسط خودتان یا ابزارهای تست می توان آن ها را یافت . برای اصلاح کردن این اشتباهات با منابعی درگیر خواهید بود – شاید مدیر پروژه ، یک برنامه نویس اشتباه را برطرف سازد ، یا یک ابزار تست کننده برنامه را ریست کند ، همه موارد نیازمند صرف وقت و هزینه است . همچنین ، هر تغییری در کد می تواند به بروز اشتباهات بالقوه منتهی شود – ممکن است این طور هم نباشد ولی احتمالش همواره وجود دارد . همانطور که برای اصلاح اشتباهات کدهایی را به برنامه اضافه می کنید ، کد اولیه پیچیده تر شده و به ناگاه ممکن است با راه حلی ناقص کار را به پایان برسانید .

بنابراین برای محافظت از کدهایتان چه باید بکنید ؟ برای کدنویسی تدافعی چند روش وجود دارد که در ذیل به بررسی آن ها می پردازیم .

۱۴-۳- چک کردن پارامترها

یکی از روش های کدنویسی تدافعی ، چک نمودن پارامترهای متدهاست . در هنگام نوشتن زیر برنامه یا توابع هرگز نباید چنین تصور کنید که مقدار ارسالی به پارامترها معتبر می باشد . باید خودتان آن ها را چک کنید ، خصوصا اگر محتویات آن ها از خارج برنامه تان سرچشمه می گیرد .

به طور مثال کد زیر را که در صفحه ی مقالات بهبازار نوشته ایم در نظر بگیرید :

```
_ID = Request.QueryString["id"];

if (_ID != null)
{
    func_FillDataGrid();

    this.Title = lblTitle.Text;
}
else
{
    Response.Redirect("~/Default.aspx");
}
```

در این قطعه کد ما پارامتر **ID** را چک می کنیم تا در صورتی که **Null** برمی گرداند کاربر را به صفحه اصلی وب سایت بازگرداند . با این طریق اگر اشتباهی در صفحه رخ دهد از دید کاربر مخفی می ماند . یا حتی می توانید مقدار **ID** را با تعداد ستون های موجود در جدول بانک اطلاعاتی خود مقایسه کنید و در صورتی که این مقدار از تعداد سطرهای جدول بیشتر بود و در جدول موجود نبود پیغام خطایی را به کاربر نشان دهد تا کاربر یا آدرس را درست کند و یا به صفحه اصلی بازگردانده شود . این کارها از نفوذ هکرها از این طریق نیز جلوگیری خواهد کرد .

۱۴-۴- اجتناب از فرضیات

همانند چک کردن پارامترها ، باید در کدنویسی از در نظر گرفتن هر نوع فرضیه ای اجتناب کنید . به طور مثال شما صفحه ای دارید که در آن باید حتما ایمیل کاربر به طور صحیح و کامل وارد شود ، بنابراین شما باید کاربر را وادار به وارد کردن صحیح آدرس ایمیل نمایید و نباید فرض را بر این بگذارید که کاربر همیشه این کار را به درستی انجام می دهد . حتی ممکن است که بعضی از هکرها عمدا آدرس را اشتباه وارد کنند تا شاید بتوانند از این طریق راه نفوذی به سیستم شما پیدا کنند ! بنابراین در برنامه نویسی هیچ گاه فرض

را بر این نگذارید که با یک کاربر کاملا درستکار و بدون خطا روبرو هستید ، یادتان باشد که ، انسان جایز الخطاست !

۱۴-۵- پارمترهای پرس و جو

اگر در برنام هایتان از **Wizard** و کنترل های منبع داده ایجاد می کنید ، آن گاه با چنین مشکلی برخورد نخواهید کرد ، زیرا دستورات **SQL** تولید شده شامل پارامترها هستند . اما اگر دستورات **SQL** را بصورت دستی ایجاد کنید و یا از برنامه دیگری به ارث ببرید ، ممکن است با کدی مثل کد زیر برخورد کنید :

```
string _strSql = "SELECT * FROM  Art WHERE ArtNo = " + ID;
```

کد فوق مقدار یک متغیر به نام **ID** را با فیلد **ArtNo** مقایشه می کند و با اجتماع کل رشته یک دستور **SQL** کامل را تشکیل می دهد . مشکل اینجاست که اگر کاربر به جای **ID** از تزریق **SQL** (**SQL Injection**) استفاده کند که در بخش امنیت در **ASP.NET** توضیح داده شده است آن گاه با مشکل مواجه خواهید شد . بنابراین برای حل این نوع مشکل می توانید از رویه های ذخیره شده استفاده کنید تا کاملا این نوع حملات خنثی گردند .

۱۴-۶- تعیین اعتبار کاربر

قبلا بیان کردیم که هر چیزی که کاربر وارد می کند ، بطور خودکار مشکوک می باشد و یکی از کارهایی که می توان برای کاستن خطر مورد استفاده قرار داد ، تعیین اعتبار (**Validation**) ورودی می باشد . بهتر است که همان ابتدا از وارد کردن مقادیر غلط توسط کاربر ممانعت کنیم . این عمل به دو دلیل خوب است : اولا بدین معناست که کدتان ایمن تر می باشد ، ثانیا کاربران با پیام های خطای ابهام انگیز مواجه نخواهند شد . آن ها ممکن است هنوز هم پیامی مبنی بر اینه ورودی تان اشتباه است را دریافت کنند .

برای جلوگیری از چنین مشکلاتی پینج کنترل تعیین اعتبار در **ASP.NET** وجود دارد که می توانند ورودی کاربر را قبل از رسیدن آن به کد چک کنند :

- **RequireFieldValidator** : این کنترل تضمین می کند که یک فیلد خالی باقی مانده نباشد.
- **CompareValidator** : متن وارد شده را با یک مقدار و یا کنترلی دیگر مقایسه می کند .
- **RangValidator** : تضمین می کند که متن وارد شده در یک محدوده خاص قرار دارد .
- **RegularExpressionValidator** : متن وارد شده را با یک عبارت منظم (**Regular Expression**) تطبیق می دهد .

- **CustomValidator** : که با اجرای کدی دلخواه (**Custom Code**) متن وارد شده را تعیین اعتبار می کند .

کنترل های فوق از کنترل های استاندارد **ASP.NET** هستند که معمولا در کنار کنترل هایی که می خواهند آن ها را تعیین اعتبار کنند ، بر روی صفحه قرار می گیرند . یک کنترل **ValidationSummary** نیز وجود دارد که امکان نمایش پیام های خطا را بطور یکجا مهیا می کند .

۱۴-۷- مقابله با استثناء

مقابله با استثناء (**Exception Handling**) اصطلاحی است که به هر چیز غیر عادی که در برنامه رخ می دهد اطلاق می شود . هر چند مقابله با استثناء بخشی از کدنویسی تدافعی می باشد ، تفاوتی بین نوشتن برنامه هایی که براحتی می توانند مورد اشتباه واقع شوند و یا حتی انتظار دارید که اشتباه شوند (مثل کلمه عبور) و چیزهایی که هیچ کنترلی بر روی آن ها ندارید ، موجود می باشد . مثلا در نظر بگیرید ، اگر نتوانید به بانک اطلاعاتی متصل شوید ، چه اتفاقی می افتد ؟ چه باید بکنید ؟ این شرایط را چگونه می توانید اداره کنید ؟ اینک این حالت شبیه حالتی است که واقعا نمی توانید کاری را در برنامه انجام دهید ، زیرا شاید برنامه کاملا به بانک اطلاعاتی وابسته باشد ، اما هنوز هم نمی خواهید که کاربرانتان با یک پیام خطای زشت مواجه شوند – می خواهید پیام خطای خودتان را نشان دهید . شاید به اطلاع آن ها برسانید ، خطایی رخ داده است و می توانید این کار را بعدا بطور مجدد تست نمایید .

بنابراین پیشنهاد می شود استراتژی داشته باشیم که از قبل بر روی آن کار کرده باشیم و بدانیم که وقتی خطایی اتفاق می افتد ، چه باید بکنیم .

استثناء ها چیستند ؟

استثناء محدوده ای از کلاس ها هستند که همگی آن ها از کلاس پایه **Exception** مشتق شده اند . انواع مختلفی از استثناء ها مثل **FileNotFoundException** وجود دارد . این استثناء زمانی رخ می دهد که بخواهید به فایلی که موجود نیست دسترسی پیدا کنید . یا استثنای **SqlException** ، این استثناء زمانی اتفاق می افتد که در بانک اطلاعاتی مشکلی وجود داشته باشد .

شیء استثناء

شیء استثناء (**Exception Object**) ، کلاس پایه کلیه استثناء هاست و خواص اصلی کلیه استثناء ها را دارا می باشد . این خواص در جدول زیر توضیح داده شده اند :

خاصیت	توضیحات
Data	کلکسیونی از جفت های کلید/مقدار که اطلاعات اضافه ای را درباره استثناء فراهم می کند .
HelpLink	لینکی به فایل **Help** (کمک) است . این فایل شامل توضیحاتی درباره استثناء می باشد .
InnerException	استثنای اصلی که باعث بروز مشکل شده است .
Message	متنی که استثناء را توصیف می کند .
Source	نام برنامه یا شیئی که باعث بروز مشکل شده است .
StackTrace	ردگیری متد فراخوانی شده ای که به مشکل منتهی می شود .
TargetSite	متدی که استثنای فعلی را ایجاد کرده است.

جدول ۱۴-۱- خواص شیء استثناء

ممکن است متوجه خاصیت **InnerException** شده باشید . این خاصیت این امکان را برای استثناء ها فراهم می کند که بتوانند به صورت پشته بر روی هم انباشته شوند و این در زمانی که لایه هایی از کد دارید ، مفید می باشد . مثلا ، ممکن است یکی از بخش های زیرین چارچوب **.NET** استثنائی ایجاد کند ، اما این استثناء باید جهت فراهم کردن اطلاعات بیشتر در راستای دیگری پیچیده (**Wrap**) شود .

سایر استثناء ها ممکن است برای تعریف چیزهای خاصی بر روی استثناء ، خواص اضافه ای را تعریف کنند . مثلا **SqlException** زیر را هم تعریف می نماید :

خاصیت	توضیحات
Class	شدت (**Severity**) خطا ، بصورتی که **SQL Server** تعریف می کند .
Errors	کلکسیونی از اشیاء **SqlError** که جزئیات مشکل را بیان می کند .
LineNumber	شماره خطی در داخل **SQL** یا روال ذخیره شده که خطا در آن جا رخ داده است.
Number	شماره خطا .
Procedure	نام روال ذخیره شده ای که خطا در آن جا رخ داده است .
Server	نام ماشین **SQL Server** .
Source	نام فراهم کننده داده .
State	کد خطای سمت **SQL Server** .

جدول ۱۴-۲- خواص **SqlException**

می توانید مشاهده کنید ، وقتی استثنائی اتفاق می افتد ، اطلاعات خاصی درباره مشکل را در اختیار خواهید داشت ، اما اگر از نوع صحیح استثناء استفاده کنید ، اطلاعات بیشتری را بدست خواهید آورد .

نحوه به تله انداختن استثناء ها

بمنظور در دست گرفتن کنترل برنامه ها ، به تله انداختن استثناء ها (**Trapping Exception**) بسیار حیاتی می باشد و این کار توسط دستورات **Try ... Catch** انجام می شود . فهم این موضوع از طریق بیان یک مثال ساده تر می باشد .

به عنوان مثل کد زیر را که ما در یکی از صفحات سایت بهبازار استفاده کرده ایم در نظر بگیرید :

```
private void func_Fill ()
{
        try
        {
            strSql = "SELECT * FROM  Art WHERE ArtNo = " + ID;

            ds = new DataSet();
            da = new SqlDataAdapter(strSql, con);
            da.Fill(_ds, "Art");

          lblTitle.Text = _ds.Tables["Art"].Rows[0]["
          Title"].ToString();
          lblResume.Text = _ds.Tables["Art"].Rows[0]["
          Resume"].ToString();
          lblDate.Text = _ds.Tables["Art"].Rows[0]["
          Date"].ToString();
          lblTime.Text = _ds.Tables["Art"].Rows[0]["
          Time"].ToString();
          lblBody.Text = _ds.Tables["Art"].Rows[0]["
          Body"].ToString();
          lblAuthor.Text = _ds.Tables["Art"].Rows[0]["
          Author"].ToString();
        }
        catch
        {
            Response.Redirect("~/ErrorPage.aspx");
        }
}
```

در این کد از در بخش **Try** اعمالی که می خواهیم در حالت عادی انجام دهیم را نوشته ایم و در بخش **Catch** دستوری نوشته شده است که در صورت وقوع خطا کاربر را به صفحه ی پردازش خطاها ارجاع می دهد . هنگامی که این تابع اجرا می شود ، ابتدا بخش **Try** شروع به اجرا شدن می کند ، در صورتی که در هنگام پردازش کدها خطایی رخ دهد برنامه به بخش **Catch** پرش می کند و دستورات قسمت **Catch** اجرا می شود . به همین سادگی !

اداره کردن استثناء های سراسری

اداره کردن استثناء ها در محلی که رخ می دهند هم خوبست و هم بد . در صفحه **CheckOut** بخاطر وجود تراکنش ، با استثناء باید بصورت محلی برخورد می شد ، اما در بسیاری از موارد شکل خاصی از اداره کردن متمرکز استثناء ها مورد نیاز می باشد . همچنین به روش هایی برای اداره کدن استثناء هایی که در جای دیگری به دام نیفتاده اند نیاز دارید . در این مورد نیز **CheckOut** نمونه خوبی می باشد ، زیرا یک استثنای **SqlException** برای اداره کردن وجود دارد و چیز دیگری موجود نیست . اگر استثناء های دیگری رخ دهد ، چه اتفاقی می افتد ؟ این نکته بسیار مهمی است ، زیرا نمی توان در هر جایی ازکد که احساس می شود ممکن است استثنائی رخ دهد ، یک بلاک **Try/Catch** قرار داد . در واقع این کار اصلا خوب نیست و موجب کاهش خوانائی برنامه می گردد .

فایل **Clobal.asax** برای مدیریت نمودن استثناء های سراسری می باشد و حاوی کدهای برنامه کاربردی (**Application**) می باشد . در اینجا اصطلاح **Application** دارای معنی خاصی است ، زیرا کدهای موجود در **Global.asax** به رویدادهایی که در سطح برنامه کاربردی رخ می دهند ، پاسخ می دهد . این رویدادها ، رویدادهایی هستند که در حین اجرای برنامه کاربردی و در جاهای خاصی ایجاد می گردند . این رویدادها توسط **ASP.NET** ایجاد می شوند . **Global.asax** صفحه ایست که تنها شامل کد (**Code-Only**) می باشد ، و هیچ فعل و انفعالی با کاربر ندارد .

در صفحه **Global.asax** چندین رویداد وجود دارد :

- رویداد **Application_Start** : این رویداد ابتدای آغاز برنامه کاربردی ایجاد می گردد و این زمانی خواهد بود که اولین کاربر به سایت دسترسی یافته و باید برای تنظیم شرایط شروع اولیه مورد استفاده قرار گیرد .
- رویداد **Application_End** : زمانی که برنامه متوقف می شود ، این رویداد ایجاد خواهد شد.
- رویداد **Session_Start** : وقتی کاربری جلسه ای را آغاز می کند ، این رویداد ایجاد می گردد و این زمانی خواهد بود که کاربر دسترسی به سایت را برای اولین بار آغاز می نماید و شامل زمانی که کاربر پنجره مرورگر را بسته و مجددا باز می نماید نیز خواهد بود .
- رویداد **Session_End** : وقتی کاربر جلسه را به اتمام می رساند ، این رویداد ایجاد می گردد و زمانی که کاربر مرورگر را می بندد را شامل نمی شود ، زیرا جلسات دارای یک زمان اتمام (**TimeOut**) هستند . اگر در طول این زمان هیچگونه فعالیتی از سوی کاربر انجام نشود ، جلسه پایان می یابد .

- رویداد **Application_Error** : زمانی که رویدادی اداره نشده باشد ، ایجاد می گردد .
- رویداد **Profile_OnMigrateAnonymous** : وقتی کاربری بی نام به سایت وارد (**Login**) می شود . این رویداد ایجاد شده و اجازه جابجایی خواص **Profile** را فراهم می کند .

همانطور که می توانید ببینید ، رویدادی که به آن علاقمندید در رویداد **Application_Error** قرار دارد و جاییست که می توانید کدهای لازم را برای اداره کردن متمرکز استثناء های بدام نیفتاده اضافه کنید

۱۴-۸- صفحات خطای سلیقه ای

یکی از مشکلات مربوط به کدهای اداره خطایی که تا به حال ملاحظه کردید ، این است که کاربر هنوز هم پیامی گیج کننده را مشاهده می کند . در حالت ایده آل ، می خواهید پیامی را به کاربر نمایش دهید ، که واضح بوده هیچ گونه ابهامی در آن وجود نداشته باشد و از ردگیری پشته استفاده نشود . این کار دو علت دارد : اولا ردگیری پشته چیزی نیست که کاربران بخواهند آن را مشاهده کنند . اگر چیزی خراب شود ، آن گاه نیاز است که توصیف روشن را از بروز مشکل مشاهده کنند . ثانیا نایش یک ردگیری پشته ، اطلاعات زیادی را درباره سایت تان فراهم می کند که هکرها می توانند از این جزئیات بخوبی استفاده کنند . حتی اگر سایت تان ایمن باشد ، آن ها می توانند در حین اینکه سعی دارند سایت تان را هک کنند ، باعث کند شدن سیستم نیز شوند .

همانند استثناء ها انواع دیگری از خطاها نیز وجود دارد که رویت آن ها برای کاربر خوشایند نیست . مثلا اگر صفحه ای را تغییر نام دهید و لینک های مربوط به آن را به روز نکنید ، یا اینکه کاربر نام اشتباهی را برای صفحه تایپ کرده باشد ، با خطای شماره ۴۰۴ مواجه خواهید شد . این شماره خطا بیانگر آن است که صفحه یافت نشده است . برنامه های **ASP.NET** را می توان به گونه ای پیکربندی کرد که بسته به نوع خطا ، کاربر را به صفحات دیگری هدایت کند .

پیکربندی صفحات خطای سلیقه ای

پیکربندی صفحات خطای سلیقه ای در فایل **Web.config** و با استفاده از بخش **customErrors** انجام می شود . برای مثال :

```
<customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404"
redirect="FileNotFound.htm" />
</customErrors>
```

ویژگی **mode** می تواند یکی از حالات زیر باشد :

- حالت **off** : در این حالت جزییات خطای **ASP.NET** همواره نشان داده می شود ، حتی اگر یک صفحه خطای سلیقه ای نیز وجود داشته باشد .
- حالت **on** : در این حالت خطای سلیقه ای همواره نشان داده می شود و جزئیات خطای **ASP.NET** هرگز نشان داده نمی شود .
- حالت **RemoteOnly** : در این حالت جزئیات خطای **ASP.NET** تنها برای کاربران محلی نشان داده می شود . کاربران محلی کاربرانی هستند که بطور محلی به ماشین وارد شده اند (**Logged on**) . برای کاربران راه دور (هر شخص دیگر که از سایت استفاده می کند) یکی از دو چیز نشان داده می شود : صفحه پیش فرضی که وقوع خطایی را به کاربر اطلاع می دهد ولی هیچ جزئیاتی از خطا را نشان نمی دهد ، یا اگر صفحه خطای سلیقه ای موجود باشد آن را نشان می دهد .

مقادیر **on** و **RemoteOnly** باید برای یک سایت فعال استفاده شود ، در حالیکه مقدار **off** می تواند برای اشکالزدائی استفاده گردد .

ویژگی **defaultRedirect** صفحه ای را که در صورت بروز یک خطای اداره نشده نشان داده می شود را تعریف می کند .

عنصر **error** جزییات مربوط به خطاهای خاصی را توضیح داده و صفحه ای که در صورت بروز خطا باید به آن رفت را مشخص می نماید . در اینجا **statusCode** برابر ۴۰۴ است که نمایانگر صفحه ای گم شده می باشد ، بنابراین اگر کاربر نتواند صفحه ای را که به دنبالش می گردد ، پیدا کند به صفحه **FileNotFound.htm** هدایت خواهد شد . بدین ترتیب می توانید برای هر خطایی ، صفحه ای خاص داشته باشید و کاربر را در حل مشکل کمک و راهنمایی نمایید . در مثال صفحه گم شده ، چنین توضیح داده می شود که صفحه یافت نشد و می توانید کاربران را راهنمایی کنید که **URL** تایپ شده را چک کرده و در صورت اشتباه آن را تصحیح نمایند .

۱۴-۹- اشکالزدایی و ردگیری

کنترل نمودن نحوه نمایش خطاها برای کاربر تنها بخشی از فرآیند تولید سایت ها می باشد و ردگیری (**tracing**) خطاهای ایجاد شده نیز به همین اندازه اهمیت دارد . در حالت ایده آل ، خطاها باید در حین تولید و تست سایت مشخص شوند و در بسیاری از موارد ، تست کردن سایت به عنوان بخشی از فرآیند

تولید به حساب می آید . بسیاری از شرکت ها مثل مایکروسافت ، شامل تیم هایی برای تست کردن پروژه ها در حین فرآیند تولید هستند تا در این فرآیند تا آن جا که امکان دارد ، خطاها را یافته و برطرف سازند .

شما نیز به عنوان تولید کننده ممکن است دچار اشتباه شوید . این اشتباهات می تواند بسیار ساده (مثل خطاهای تایپی) و یا پیچیده (مثل خطاهای کدنویسی) باشد . ردیابی اشتباهات تایپی بسیار ساده است و این نوع اشتباهات معمولا خطاهای کامپایلری را تولید می نمایند . اما خطای زمان اجرا می تواند بسیار مشکل زا باشد . ردگیری و اشکالزدایی ، دو روش اصلی برای یافتن خطاها می باشند .

فصل پانزدهم

نصب و راه اندازی سایت

- ثبت دامنه و هاست برای سایت
- ارسال فایل های سایت از طریق **FTP**
- استفاده از کنترل پانل سایت
- جستجوی نام های دامنه

نصب و راه اندازی سایت

۱۵-۱- ثبت دامنه و هاست برای سایت

برای ثبت سایت خود و انتشار آن بر روی شبکه جهانی وب ، باید دو کار ابتدایی را انجام دهید :

۱. خرید دامنه به عنوان شناسه و آدرس وب سایت خود .

۲. خرید هاست برای ارسال صفحات سایت خود به روی آن . هاست در واقع فضایی است که بر روی سرور مورد نظر به شما اختصاص داده می شود تا بتوانید پروژه خود را به اصطلاح آپلود کنید .

برای خرید دامنه می توانید با یکی از شرکت هایی که خدمات اینترنت ارائه می دهند قرارداد ببندید و نام مورد نظر خود را به آن ها برای ثبت سفارش دهید . بعد از سفارش شما و ارسال نام انتخابی خود به شرکت آن شرکت ابتدا به جستجوی دامنه انتخابی شما خواهد پرداخت و در صورتی که پیش از این توسط شخص دیگری ثبت نشده باشد ، آن را برای شما ثبت خواهد کرد .

البته در صورت تمایل و برای جلوگیری از اتلاف وقت و ارسال چند باره ی نام دامنه به شرکت ، شما خود نیز می توانید این جستجو را انجام دهید . به طور مثال این خدمات در سایت **www.parsdev.ir** ارائه می شود . که شما می توانید نام و پسوند مورد نظر خود را جستجو کنید .

و اما برای خرید هاست ، باید به نکاتی قبل از سفارش هاست دقت کنید :

۱. اول از همه و مهمترین آن این است که هاست مورد نظر شما باید چه نوع سیستم عاملی داشته باشد ؟ (ویندوز یا لینوکس ؟) . این مورد را باید با توجه به پروژه خود معین کنید . به طور مثال سایت بهبازار که توسط با کدهای **ASP.NET 2.0** نوشته شده است حتما باید به روی سیستم عامل ویندوز آپلود گردد و همچنین بر روی سیستم عامل مورد نظر باید نرم افزارهای لازم جهت اجرای فایل های **ASP.NET 2.0** نصب شده باشد . همچنین سیستم عامل باید از **SQL Server 2000** هم پشتیبانی کند .

۲. عامل دوم که جهت انتخاب سرور مورد نظر باید در نظر بگیرید ، میزان فضای مورد نیاز برای فایل های خود است که برای این کار باید پیش بینی آینده را نیز بکنید که چقدر حجم سایت شما افزایش خواد یافت . ما برای هاست بهبازار فضای **200 MB** را در نظر گرفته ایم که با پیش بینی انجام شده این میزان فضا حداقل برای مدت یکسال آینده سایت ما کافی می باشد و در نیاز می توانیم در قرار سال بعد این میزان را افزایش دهیم .

۳. مورد سوم برای انتخاب سرور میزان پهنای باند ماهیانه می باشد که یکی از موارد مهم نیز می باشد ، چرا که این عامل تاثیر مستقیم و بسزایی در سرعت لود شدن صفحات سایت شما خواهد داشت . برای این مورد نیز باید پیش بینی لازم را انجام دهید . ما برای بهبازار پهنای باند ماهیانه **15 Gig** را در نظر گرفته ایم .

۴. و دیگر مواردی که می توانید در نظر بگیرید ، میزان فضایی که به دیتابیس شما اختصاص داده می شود ، تعداد **FTP** ها ، تعداد **SubDomain** ها ، تعداد **Email** ها و امکانات جانبی سرور و کنترل پنل آن می باشد که بسته به نیازتان می توانید انتخاب کنید .

۱۵-۲- ارسال فایل های سایت از طریق FTP

شما برای ارسال فایل های سایت خود بر روی هاست می توانید از برنامه های مخصوص اینکار مانند **Smart FTP** و ... استفاده کنید. اما یکی از روش های راحت و بدون نیاز به نرم افزاری خاص استفاده از خود **IE** می باشد . به اینصورت که شما آدرس **FTP** سایت خود را به صورت زیر در قسمت **Address bar**، اکسپلورر وارد می کنید :

```
ftp://www.behbazar.com
```

بعد از زدن دکمه **Enter** ، پنجره ای برای شما باز خواهد شد که از شما نام کاربری و کلمه عبور **FTP** را می خواهد . بعد از وارد کردن این دو (آن ها را باید از شرکتی که هاست را از آن تهیه کرده اید درخواست کنید) صفحه ی **FTP** برای شما به نمایش در خواهد آمد و شما می توانید به راحتی و تنها با کپی کردن فایل های مورد نظر خود در بخش مخصوص سایت خود را به سرور انتقال دهید .

۱۵-۳- استفاده از کنترل پنل سایت

برای استفاده از کنترل پنل سایت خود و امکانات آن باید ابتدا آدرس **CPANEL** خود را وارد کنید و سپس به صفحه ای ارجاع خواهید شد که در آن نام کاربری و کلمه عبور **CPANEL** را از شما می خواهد (این دو را نیز باید از شرکت مذکور درخواست کنید) ، بعد از وارد کردن آن ها به بخش کنترل پنل هدایت خواهید شد . که در آن جا می توانید از امکانات زیادی که برای شما تدارک دیده شده است استفاده کنید .

15-4- جستجوی نام های دامنه (Whois) در ASP.NET

اگر به سایتهای ثبت دامنه مانند **www.Register.com** سر زده باشید، دیده اید که این سایتها امکاناتی برای جستجوی نام دامنه در اختیار شما می گذارند که به شما وجود یا عدم وجود آن نام را اطلاع

می دهند و اطلاعاتی نیز درباره فرد یا سازمان صاحب نام می دهد. در این بخش نحوه ی انجام این کار را با **ASP.NET** شرح می دهیم.

برای انجام عمل جستجو ابتدا باید یک ارتباط با سرورهایی که بانکهای اطلاعاتی آنها این مشخصات را دارند ارتباط برقرار کرد و با یک **Query**، اطلاعات مورد نظر را درخواست نمود. این ارتباط از نوع **TCP** و از طریق پورت ۴۳ انجام می گیرد. لیست کاملی از این سرورها را در آدرس

www.mit.edu/afs/athena/contrib/potluck/net-services/whois-servers.list

می توانید ببینید.

مشخص کردن فضا نامها

ابتدا فضا نامهای مورد نیاز را مشخص می کنیم:

```
Using System.Net.Sockets;
Using System.Text;
Using System.IO
Using System.Text.RegularExpressions;
```

با استفاده از **TcpClient** در **.NetFramework** می توانیم این ارتباط را با سرور مورد نظر برقرار نماییم. برای استفاده از این کلاس باید فضانام **System.Net.Sockets** را وارد کنیم. از دو فضانام دیگر نیز برای فرمت ورودی و خروجی خود استفاده می کنیم.

ایجاد و ارسال Query

ابتدا متغیرهای مورد نیاز را تعریف می کنیم:

```
string StrSvr, StrDomain, Resp;
TcpClient TcpClt;
byte[] ArrDomain;
Stream TcpStr;
StreamReader TcpStrRdr;
```

حال باید رشته ای را که می خواهیم به عنوان **Query** به سرور ارسال نماییم، ایجاد کنیم. برای این کار باید نام درخواستی کاربر را همراه پسوند مورد نظر(**Com،Org**...) را به سرور ارسال نماییم. توجه داشته باشید که هر کدام از سرورهای موجود در لیست فوق الذکر برای جستجوی پسوند خاصی می باشد. برای

مثال **whois.internic.com** تنها برای دامنه های **net**. و **com**. و **edu**. می باشد و اگر دامنه ای با پسوند **Info**. را با این سرور جستجو نمایید، جواب مطلوب را نخواهید گرفت.

بنابراین ابتدا باید پسوند مورد نظر کاربر و سپس سرور مربوطه را مشخص نماییم . در اینجا برای انتخاب پسوند از **DropDownList** استفاده کرده ام:

```
if (DDLstSuffix.SelectedItem.Value == ".COM" ||
DDLstSuffix.SelectedItem.Value == ".NET" ||
DDLstSuffix.SelectedItem.Value == ".EDU")
{
            StrSvr = "whois.internic.net";
            StrDomain = TxtDomainName.Text.Trim() +
DDLstSuffix.SelectedItem.Value + "\r\n";
}
else if (DDLstSuffix.SelectedItem.Value == ".ORG")
{
            StrSvr = "whois.publicinterestregistry.net";
            StrDomain = TxtDomainName.Text.Trim() +
DDLstSuffix.SelectedItem.Value + "\r\n";
}
else if (DDLstSuffix.SelectedItem.Value == ".BIZ")
{
           StrSvr = "whois.neulevel.biz";
            StrDomain = TxtDomainName.Text.Trim() +
DDLstSuffix.SelectedItem.Value + "\r\n";
}
else if (DDLstSuffix.SelectedItem.Value == ".INFO")
{
           StrSvr = "whois.afilias.info";
            StrDomain = TxtDomainName.Text.Trim() +
DDLstSuffix.SelectedItem.Value + "\r\n";
}
else if ((DDLstSuffix.SelectedItem.Value == ".IR") ||
(DDLstSuffix.SelectedItem.Value == ".CO.IR") ||
(DDLstSuffix.SelectedItem.Value == ".NET.IR") ||
(DDLstSuffix.SelectedItem.Value == "ID.IR"))
{
            StrSvr = "whois.nic.ir";
            StrDomain = TxtDomainName.Text.Trim() +
DDLstSuffix.SelectedItem.Value + "\r\n";
}
```

همانطور که می بینید، رشته ای که برای جستجو ارسال می شود به "**\r\n**" ختم می شود. اگر این رشته را در انتهای نام دامنه قرار ندهید، سرور نمی تواند جواب مطلوب را برگرداند.

اتصال به سرور

حال یک ارتباط باید با سرور مورد نظر برقرار نماییم. کلاس **TcpClient**، متدهایی برای اتصال، دریافت و ارسال اطلاعات دارد. به دو صورت می توانیم با یک سرور ارتباط برقرار کنیم:

۱ – ایجاد یک نمونه از کلاس **TcpClient** بدون هیچ پارامتری و سپس استفاده از متد **Connect**:

```
TcpClient TcpClt = new TcpClient();
TcpClt.Connect(StrSvr, 43);
```

۲ – ایجاد یک نمونه از کلاس **TcpClient** با پارامترهای آدرس سرور و پورت مقصد که می خواهید با آن ارتباط برقرار کنید. این **Constructor** ارتباط را به صورت اتوماتیک ایجاد می کند:

```
TcpClient objTCPC = new TcpClient(StrSvr, 43);
```

پس از آن با متد **GetBytes** کلاس **Encoding** (فضا نام **System.Text**) رشته **StrDomain** را به یک آرایه از نوع بایت می نویسیم.

```
ArrDomain = Encoding.ASCII.GetBytes(strDomain);
```

با استفاده از متد **GetStream** کلاس **TcpClient** ، یک **Stream** ایجاد می کنیم و از طریق آن به ارسال و دریافت اطلاعات می پردازیم:

```
TcpStr = TcpClt.GetStream();
```

با استفاده از این **Stream** می توانیم با متد **Write** یک آرایه از **Byte** را در **Stream** جاری نوشت :

```
TcpStr.Write(ArrDomain, 0, StrDomain.Length);
```

دریافت و نمایش نتیجه جستجو

یکی از راههای دریافت اطلاعات از سرور، ایجاد یک نمونه از کلاس **StreamReader** و سپس خواندن اطلاعات بر اساس یک **Encoding** مشخص می باشد.

```
TcpStrRdr = new StreamReader(TcpClt.GetStream(),
Encoding.ASCII);
```

قدم بعد، استفاده از متد **ReadToEnd** برای دریافت جواب سرور می باشد. این رشته شامل کاراکترهای خط جدید "**\n**" می باشد که باید با "**
" جایگزین شود. برای این کار از متد **Replace کلاس **RegEx** استفاده می کنیم ، سپس برای راحتی جستجو این رشته را به حروف کوچک تبدیل می کنیم :

```
Resp = Regex.Replace(TcpStrRdr.ReadToEnd(), "\n", "<br>");
Resp = Resp.ToLower();
```

اگر در این رشته، زیر رشته هایی مانند "**no match**" یا "**not found**" یا "**found no entries**" وجود داشته باشد، نام دامنه مورد نظر آزاد می باشد. در غیر اینصورت آن دامنه ثبت شده است. حال با استفاده از این مساله، نتیجه را به کاربر اعلام می کنیم. برای جستجوی این زیر رشته ها از متد **IsMatch** کلاس **RegEx** استفاده می کنیم. اگر در این متد زیر رشته مورد نظر در رشته موجود باشد True و در غیر اینصورت **False** برگشت داده می شود.

```
if (Regex.IsMatch(Resp, "no match") || Regex.IsMatch(Resp,
"not found") || Regex.IsMatch(Resp, "no entries found"))
{
            SearchRes = " دامنه مورد نظر موجود می باشد ";
            PnlOrder.Visible = true;
            PnlOrderOk.Visible = false;
}
else
{
            SearchRes = " دامنه مورد نظر موجود نمی باشد ";
            PnlOrder.Visible = false;
            PnlOrderOk.Visible = false;
}
```

در انتها نیز ارتباط را می بندیم.

```
TcpClt.Close();
```

خلاصه مطالب

فعالیت های تجاری یکی از مهم ترین کاربردهایی است که در یک شهر الکترونیک انجام می شود. خرید و فروش آنلاین در عصر دیجیتال باعث شده است که فعالیت های تجاری تا حد بسیار زیادی کم هزینه تر، آسان تر و با سطح اطمینان بالاتر انجام شوند . فعالیت های مرتبط با کسب و کار اگر به صورت سنتی انجام شوند هزینه های بسیار زیادی را در بر خواهند داشت . یکی از مهمترین این هزینه ها هزینه تبلیغات ، اجاره محل و انبار می باشد . به این معنی که در حالت فعلی خرید و فروش ، به دلایل تحمیل هزینه های بیان شده ، قیمت تمام شده کالا افزایش می یابد . در صورتی که اگر این معاملات از طریق اینترنت انجام شود ، قیمت ها کاهش یافته و هزینه های اشاره شده از میان برداشته خواهد شد. هر چند که این امکان نیز وجود دارد که بتوان در مواردی تخفیفات عمده ای را نیز برای مشتری منظور کرد . فروشگاه های آنلاین یکی از متداول ترین مکان های خرید و فروش در کشورهای پیشرفته می باشند . در اینگونه فروشگاه ها که بر روی انترنت قرار دارند مشخصات تمام اقلام وجود دارد و تنها کافی است که اقلام مورد نظر را انتخاب کرده و با استفاده از کارت اعتباری هزینه آن را پرداخت تا در اسرع وقت کالا برای مشتری فرستاده شود . در مورد فروش اقلام هم می توان مشخصات اقلام مورد نظر خود را بر روی اینترنت قرار داد تا در معرض دید عموم قرار گیرد و مشتریان در صورت تمایل آن ها را به صورت آنلاین سفارش دهند .

در سایت بهبازار ما محلی را برای عرضه محصولات دیگر شرکت ها ایجاد کرده ایم تا آن ها با تبلیغات خود در سایت بهبازار محصولات خود را به دیگران معرفی نمایند . در واقع ما به عنوان واسطه ای عمل می کنیم و پایگاه داده ای از محصولات متنوع طبقه بندی شده در اختیار بازدید کنندگان سایت قرار می دهیم .

در این مستندات ما به شرح چگونگی ساخت و ایجاد این سایت پرداخته ایم و از ابتدا تا انتها به صورت بهم پیوسته توضیحات لازم را ذکر کرده ایم .

به امید اینکه مفید واقع شود .

فهرست منابع

کتاب ها

۱. گام به گام با **ASP.NET 2.0** کریس هارت ، علیرضا انصاری، انتشارات ناقوس اندیشه، ۱۳۸۵.
۲. مرجع کامل **ASP.NET** با مروری بر **ASP** استفان والتر، بابک احترامی، دانش نگار، ۱۳۸۳.
۳. شیوه ارئه مطالب، سید محمد تقی روحانی رانکوهی، انتشارات جلوه ، پاییز ۸۳.
۴. برنامه نویسی شیء گرا با **ASP.NET2.0** عارف کریمی، ناقوس اندیشه، ۱۳۸۵.
۵. آموزش گام به گام **ASP.NET** مهندس جعفرنژاد قمی، انتشارات علوم رایانه، زمستان ۱۳۸۳ .
۶. آموزش گام به گام **ASP.NET**، داتی اندرو، مهندس سعید هراتیان – مهندس مهدی فلاح، ساحر، ۱۳۸۴ .
۷. آموزش گام به گام **ASP.NET**، داتی اندرو، مانی قاسم نیا همدانی، انتشارات ناقوس، ۱۳۸۳.
۸. امنیت در **ASP.NET 2.0** وحید نصیری، ناقوس اندیشه، ۱۳۸۶ .
۹. ۱۰۱ نکته و ترفند برای استفاده از **CSS** در طراحی وب، اندرو ریچل، امیرعباس عبدالعلی، ناقوس اندیشه، ۱۳۸۵.
۱۰. ۱۰۱ روش بازاریابی در اینترنت، سوزان سوبینی، فاطمه فرزانه – عبدالوهاب فخریاسری، ناقوس اندیشه، ۱۳۸۴.
۱۱. رازهای طراحی وب سایت، سید ابوالحسن تجملی محدث، ناقوس اندیشه، ۱۳۸۵.
۱۲. آموزش گام به گام **SQL**، مهندس عین الله جعفرنژاد قمی – مهندس رمضان عباس نژاد، علوم رایانه، بهار ۱۳۸۴.
۱۳. آموزش قدم به قدم **SQL Server 2000**، ریوردان ربکا، مانی قاسم نیا همدانی، ناقوس اندیشه، ۱۳۸۳.
۱۴. شهر الکترونیک، دکتر علی اکبر جلالی، مرکز انتشارات دانشگاه علم و صنعت ایران، ۱۳۸۲.

۱۵. آموزش گام به گام برنامه نویسی بانک های اطلاعاتی در **WEB**، جعفرنژاد قمی، علوم رایانه ، پاییز ۱۳۸۳.
۱۶. آموزش گام به گام **AJAX** ، حسین خوشرفتار منفرد، انتشارات نشر گستر، زمستان ۱۳۸۶.
۱۷. راه و رسم رونق تجارت با **Google**، برد هیل، مهیار کلانتری فرد، ناقوس اندیشه، ۱۳۸۶.

مقالات

۱. وب ۲ دنیایی بافته از مشارکت ، علیرضا صالحی ، ماهنامه شبکه - اسفند ۱۳۸۴ شماره ۶۳ .
۲. وب۲ - شوق یک جهان نو ، پرهام ایزدپناه ، ماهنامه شبکه - اسفند ۱۳۸۴ شماره ۶۳ .
۳. هفت مشخصه مهم وب۲ ، بهروز نوعی پور ، ماهنامه شبکه - اسفند ۱۳۸۴ شماره ۶۳ .
۴. ارسال ایمیل در **ASP.NET** ، بابک خالدیان ، وب سایت دات نت سورس .
۵. آموزش ساخت تصاویر امنیتی، سید محمد رضا فراحی ، وب سایت نت سورس .
۶. آموزش ساخت کامپوننت های وب در **ASP.NET 2.0**، بابک خالدیان ، وب سایت دات نت سورس .
۷. حفظ حالت در **ASP.NET** ، وحید نصیری .
۸. ساخت پروفایل در **WebHostingMatrix**، وحید نصیری .
۹. تجارت الکترونیک ، نوید زراعتی .
۱۰. تفاوت های اصلی **DataSet** و **DataReader** ، مجتبی صحرایی ، وب سایت دات نت سورس .
۱۱. محافظت از برنامه های خود در برابر **SQL Injection** ، بابک زواری، وب سایت دات نت سورس .
۱۲. خطاهای متداول در **ASP.NET**،وحید نصیری .
۱۳. ذخیره تصاویر در **SQL Server** ، اردوان دژپناه.
۱۴. کد کردن**Connection String** ، محمد رضا طاهری .
۱۵. روانشناسی سیستمهای کامپیوتری ، محمد حسین دالوند .
۱۶. نکات منفی وب سایت ها ، محمد توری .

وب سایت ها

۱۷. **www.sitepoint.com/article/build-whois-lookup-asp-net**
۱۸. **www.iranasp.net**
۱۹. **www.dotnetsource.com**
۲۰. **http://www.macromediax.com**
۲۱. **www.itiran.com**
۲۲. **www.barnamenevis.org**
۲۳. **http://www.nofa.ir**
۲۴. **http://www.sharghian.com**
۲۵. **www.behbazar.com**
۲۶. **www.persian-soft.com**
۲۷. **www.google.com**
۲۸. **www.asp.net**
۲۹. **www.codeproject.com**

www.itanalyze.om.۳۰
www.istgah.com.۳۱
www.parsdev.ir.۳۲
www.parslook.com.۳۳
www.yahoo.com.۳۴
www.msdn.com.۳۵
www.microsoft.com.۳۶

Payam Noor University

وزارت علوم ، تحقیقات ، فن آوری

E – commerice WebSite (BehBazar)

A Project Report

Presented to :

Behshahr Payam Noor University

In Partial Fulfillment of the Requierment for the degree of Bachelor of Science in

Computer Engine – SoftWare

Advisor :

DR – Mohammad Ebrahim Talebian

By :

Mohammad Jafari Foutami

Mohammad Shirdel

Seyed Fatemeh Hoseini

1386/10/10