مرجع کامل عبارات باقاعده

در سکوی .NET Framework

همراه با نحوه به‌کارگیری آن در دو زبان VB و C#

ترجمه و تالیف:

مهندس محمد محمدی پیروز

انتشارات پندارپارس

سرشناسه : محمدی پیروز، محمد، ۱۳۵۴ - گردآورنده، مترجم
عنوان و نام پدیدآور : مرجع کامل عبارات با قاعده در سکوی NET Framework. همراه با نحوه به کارگیری آن در دو زبان VB و #C/ ترجمه و تالیف محمد محمدی پیروز.
مشخصات نشر : تهران : پندار پارس، ۱۳۹۴.
مشخصات ظاهری : ۲۲۴ ص.: مصور، جدول.
شابک : ۱۸۰۰۰۰ ریال: 978-600-6529-80-6
وضعیت فهرست نویسی : فیپا
موضوع : متن‌پردازی
موضوع : مایکروسافت دات‌نت فریم‌ورک
موضوع : زبان‌های برنامه‌نویسی کامپیوتر
رده بندی کنگره : QA۹/۷۶ ۱۳۹۴ ۳م۲م/
رده بندی دیویی : ۵۲/۰۰۵
شماره کتابشناسی ملی : ۳۸۸۲۶۹۰

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ١٤، واحد ١٦ **www.pendarepars.com**

تلفن: ٦٦٥٧٢٣٣٥ - تلفکس: ٦٦٩٢٦٥٧٨همراه: ٠٩١٢٢٤٥٢٣٤٨ **info@pendarepars.com**

نام کتاب : مرجع کامل عبارات باقاعده در سکوی .NET Framework

ناشر : **انتشارات پندار پارس**

ترجمه و تالیف : محمد محمدی پیروز

چاپ نخست : **مرداد ٩٤**

شمارگان : **۵۰۰ نسخه**

طرح جلد : **سارا یعسوبی**

چاپ، صحافی : **روز**

قیمت : **۱۸۰۰۰ تومان** **شابک : ٦-٨٠-٦٥٢٩-٦٠٠-٩٧٨**

پیش‌گفتار

این کتاب، به شناخت یکی از زبان‌های مهم و پرکاربرد امروز دنیا به نام عبارات باقاعده (Regular Expressions) کمک می‌کند. عبارات باقاعده، خود به تنهایی امکان تولید نرم‌افزار و برنامه‌های کاربردی را ندارند، بلکه ابزاری هستند در اختیار دیگر زبان‌های برنامه‌ساز مانند C#، VB، PHP و غیره. از دستورها و امکانات این زبان در تولید کامپایلرها، صفحات وب هوشمند و دقیق و برنامه‌های کاربردی گوناگون در جهت تجزیه و تحلیل متون و رشته‌ها و جایگزینی آنها با دیگر متون یا رشته‌ها استفاده می‌شود.

این کتاب بیشتر بر مبنای استفاده از این زبان در جهت تجزیه و تحلیل رشته‌ها و متون در برنامه‌های کاربردی و کتابخانه‌های NET Freamwork. تولیدی شرکت مایکروسافت می‌باشد. هر چند با مطالعه این کتاب خواهید توانست حتی در صفحات وب و دیگر برنامه‌ها از ساختارهای تعریف شده برای عبارات باقاعده بهره برده و به اهداف خود برسید.

این کتاب در ۷ فصل به رشته تحریر در آمده است که با تعریف انواع کاراکترها، اجزای اصلی عبارات باقاعده شروع شده و به نحوه به کارگیری آن در زبان‌های گوناگون به‌ویژ دو زبان C# و VB خاتمه می‌یابد.

این کتاب از فصل‌های زیر تشکیل شده است:

۱. Regular Expressions به عنوان یک زبان

در این فصل به شما دیدی کلی از زبان برنامه‌نویسی از منظر عبارات باقاعده ارائه می‌شود.

۲. عناصر زبان Regular Expressions

در این فصل که شالوده اصلی یادگیری RegEx است، به تعریف و توضیح مجموعه‌ای از کاراکترها، عملگرها و سازنده‌هایی که در تعریف و تعیین عبارات باقاعده شما را یاری می‌کند، خواهیم پرداخت و با ارائه مثال‌هایی عملکرد آنها را برای شما تشریح خواهم کرد.

۳. کلاس‌های Regular Expression

در این فصل چگونگی استفاده از RegEx در NET Framework. را شرح می‌دهیم و مثال‌ها و کدهای نمونه‌ای به دو زبان C# و Visual Basic ارایه می‌دهیم.

٤. جزئیات رفتاری Regular Expression

این فصل شناخت درباره امکانات و رفتار NET Framework Regular Expressions. را ارائه می‌دهد.

٥. مثال‌های Regular Expression

شامل نمونه کدهایی است که در .NET Framework آمده و استفاده‌های شاخص از عبارات باقاعده را تشریح می‌کند.

۶. عبارات باقاعده در زبان‌های برنامه‌نویسی دیگر

این فصل بیشتر بر تفاوت رفتاری و استفاده از برخی علائم موجود در عبارات باقاعده در زبان‌های برنامه‌نویسی گوناگون اشاره دارد.

۷. مثال‌هایی از عبارات باقاعده برای مسائل رایج

هر چند تلاش در این بوده که این کتاب بدون اشتباه باشد اما از همه‌ی عزیزانی که این کتاب را مطالعه می‌کنند تقاضا دارم نظرات، پیشنهادها، اشتباه‌های تایپی، برنامه‌ای، محتوایی و هر آنچه که در بهبود مطالب این کتاب کارآمد می‌باشد به نشانی Piroozman@gmail.com ارسال نمایند.

محمد محمدی پیروز

فهرست

فصل نخست

Regular Expressions به عنوان یک زبان

۱-۱ پیش‌درآمد

رشته (string) بر اساس تعریف شرکت مایکروسافت، یک ساختار داده‌ای متشکل از سلسله‌ای از کاراکترهاست که معمولا نمایانگر متنی است که اشخاص می‌توانند بخوانند و text (متن یا متون) نیز داده‌هایی هستند که از کاراکترهای نمایانگر واژگان و نمادهای گفتار انسان تشکیل می‌شوند و معمولا شامل کاراکترهایی است که مطابق با استاندارد اسکی (ASCI) کد می‌شوند. این استاندارد، مقادیر عددی را به ارقام، حروف و نمادها اختصاص می‌دهد و برابر تعریفی دیگر، text در واژه پردازی و نشر کامپیوتری به بخش اصلی سند گفته می‌شود.

پس از تعریف بالا بیاید مسائلی را برای خود مطرح کنیم. فرض کنیم:

- ✓ می‌خواهیم در یک فایل متنی به دنبال واژه‌ی RegEx که بخشی از یک واژه نیست (مانند RegExpression) گشته و با واژه‌ی Regular Expression جایگزین کنیم.
- ✓ قصد داریم تمامی نشانی‌های الکترونیکی (ایمیل‌ها) موجود در متنی را یافته و آنها در صفحه وب خود تبدیل به یک لینک کنیم.
- ✓ صفحه‌ی وبی را ایجاد کرده‌ایم و بخش‌هایی را برای وارد کردن ایمیل و شماره تلفن مشتری‌ها در نظر گرفته‌ایم. می‌خواهیم مطمئن شویم که ایمیل وارد شده به شکل درست وارد شده است. برای نمونه، نشانی piroozman@gmail.com درست است ولی ایمیلی با فرمت piroozman@ درست نیست.
- ✓ قصد داریم متنی را در مکان مشخصی از فایل جست‌وجو کنیم. برای نمونه از شروع یک خط مشخص یا در پایان جلمه.

و بسیاری سناریوهای دیگر که می‌توان آنها را با عنوان دستکاری رشته‌ها (Manipulating Strings) نام برد.

حل اینگونه مسائل که در اصل، دستکاری رشته است ما را بر آن داشت تا دست به نگارش این کتاب بزنیم. این کتاب به گونه‌ای نوشته شده است که می‌تواند پاسخگوی افرادی باشد که می‌خواهند:

- ✓ با RegEx (مخفف Regular Expression یا عبارت باقاعده) آشنا شوند و در برنامه‌های خود به کار گیرند.
- ✓ مسائل پیچیده‌ی کار با الگوی تطبیق کاراکترها را در کوتاه‌ترین زمان ممکن حل کنند.
- ✓ متون خود را به شکل دلخواه و با هر نوع پیچیدگی که می‌خواهند ویرایش کنند.
- ✓ نحوه‌ی استفاده از RegEx و تغییر رفتار آنرا در .NET Framework یاد بگیرند.
- ✓ با نحوه‌ی به‌کارگیری RegEx در برخی زبان‌های برنامه‌نویسی دیگر نیز آشنا شوند.

نحو عبارات باقاعده ساده است. آن چیزی که ممکن است کمی دشوار به نظر برسد، تبدیل صورت مسئله به عبارت باقاعده است که تنها با تمرین و حل مسائل گوناگون می‌توانید از عهده‌ی حل آنها برایید. البته پس از خواندن این کتاب به سادگی خواهید توانست بیشتر مسائل را با کمی دقت و تمرین حل کنید.

۱-۲ تاریخچه

RegEx در سال ۱۹۵۰ به عنوان یک پروژه‌ی تحقیقاتی در حوزه‌ی ریاضیات ایجاد شد و چند سال بعد در سیستم عامل Unix و زبان Perl و ابزارهایی همچون grep در Unix استفاده شد و تا سال‌ها به طور انحصاری در Unix استفاده می‌شد. اما امروزه در بیشتر زبان‌های برنامه‌نویسی پشتیبانی می‌شود.

۱-۳ تعریف عبارات باقاعده

عبارات باقاعده روشی انعطاف پذیر برای کار با رشته‌ها و متون است. با استفاده از امکانات موجود در زبان عبارات باقاعده می‌توان در متون (حتی متون بسیار طولانی) عبارات خاصی را جست‌وجو، ویرایش، حذف یا جایگزین کرد. استفاده از عبارات باقاعده در برنامه‌هایی که تعامل فراوانی با رشته دارند بایسته و حتمی است.

یک مفهموم نام آشنا در بسیاری از سیستم‌های کامپیوتری، استفاده از کاراکترهای "جایگزین شونده‌ها" (wildcard) در الگوهای تطبیقی (pattern matching) می‌باشد. اگر قصد داشته باشید تمامی فایل‌های Microsoft Word را در یکی از شاخه‌های موجود رایانه خود بیابید از عبارت "*.doc"، برای جست‌وجو استفاده خواهید کرد. نماد ستاره (asterisk) به عنوان یک wildcardیی که می‌تواند هر شمار از کاراکترها را تطبیق دهد تفسیر خواهد شد.

در نوشتن برنامه‌ها یا صفحات وبی که با رشته‌ها و متون فراوان سر و کار دارند بیشتر وقت‌ها به چنین الگوهایی حتی با پیچیدگی‌های بسیار بیشتری نیاز خواهید داشت. عبارات باقاعده نیز به همین منظور ابداع و تولید شده است.

در اصل عبارات باقاعده برای " جست‌وجوی عبارات" یا "جست‌وجو و جایگذاری عبارات" استفاده می‌شوند.

جست‌وجو با RegEx

به طور مثال قصد داریم در یک متن به دنبال واژه‌ی Click بگردیم که کاراکتر نخست آن بزرگ بوده و بخشی از یک واژه‌ی (مانند MouseClick) نباشد. الگوی عبارت باقاعده استفاده شده برای حل این مسأله می‌تواند به صورت زیر باشد.

```
\bC[Ll][Ii][Cc][Kk]\b
```

الگوی بالا تمامی واژگانی همچون Click، CLiCk، ClIcK را که خود جزئی از یک واژه نباشند تطبیق می‌دهد.

در نمونه‌ای دیگر عبارت باقاعده‌ی "\s2000" تمامی رخدادهای رشته‌ی 2000 را که پیش از آن هر نوع کاراکتر فضای خالی (white space) مانند space یا tab قرار گرفته است شناسایی می‌کند.

یادداشت:

اگر نیاز به تطبیق کاراکترهای ویژه escape، مانند \s، دارید چنانچه از C++، C# یا JavaScript استفاده می‌کنید باید پیش از \s از یک "\" استفاده کنید، تا ماشین عبارت باقاعده (Regular Expression Engine) متوجه شود که کاراکتر "\" در "\s" یک کاراکتر معمولی (literal) است، وگرنه ماشین عبارت باقاعده با "\" و کاراکتر "s" در "\s" به عنوان دو عملگر مجزا رفتار خواهد کرد. اگر از VB استفاده می‌کنید مجبور به استفاده از "\" نمی باشید و اگر با زبان C# کار می‌کنید می‌توانید رشته‌های معمولی موجود در C# را به همراه یک پیشوند @ به‌کار ببرید و عمل escaping را غیر فعال کنید. به طور مثال ("\s2000"@). در ادامه، با escape کردن کاراکترها بیشتر آشنا خواهیم شد.

جست‌وجو و جایگذاری عبارات با RegEx

جست‌وجو با RegEx بسیار کاربردی و قدرتمند است. اما قدرت واقعی RegEx در جایگذاری متون یافت شده با مقداری دیگر است. برای نمونه جهت تبدیل نشانی‌های الکترونیکی به یک لینک، در آغاز نیاز دارید با استفاده از RegEx در متن به دنبال ایمیل‌ها گشته و پس از یافتن، آنها را با تگ <a> به لینک تبدیل کنید.

برای نمونه نشانی piroozman@gmail.com را به عبارت زیر تغییر دهید:

```
<a href="mailto:piroozman@gmail.com>piroozman</a>
```

مانند هر زبان دیگر، زبان عبارات باقاعده نیز، نحو، نمادها و دستورهای مختص به خود را دارد. RegEx توسط زبان برنامه‌نویسی مقصد پشتیبانی شده و زبان مقصد امکاناتی برای کار با نمادها و دستورهای آن فراهم می‌کند. بهتر است RegEx را یک زیر مجموعه از یک زبان برنامه‌نویسی در نظر بگیرید. هم‌اینک بیشتر زبان‌ها از RegEx پشتیبانی می‌کنند ولی برخی از نمادها و دستورهای RegEx مختص یک یا چند زبان برنامه‌نویسی ویژه هستند و در زبان دیگر کاربردی ندارند.

چگونه درستی RegEx نوشته شده را بیازماییم؟

برنامه‌هایی برای آزمایش عبارات باقاعده وجود دارند که از آنها می‌توانید برای بررسی درستی و عملکرد عبارات باقاعده استفاده کنید. یکی از این برنامه‌ها که به صورت رایگان عرضه می‌شود، Expresso نام دارد و می‌توانید با مراجعه به سایت http://www.ultrapico.com نرم‌افزار را دانلود و استفاده کنید. در بخش‌های بعدی با عملکرد این نرم‌افزار و امکاناتی که در اختیار قرار خواهد داد آشنا خواهید شد.

فصل دوم

عناصر زبان Regular Expressions

کیت تولید نرم‌افزار ویندوز، مجموعه گسترده‌ای از ابزارهای عبارت باقاعده را فراهم آورده است که شما را در ایجاد، مقایسه و تغییر رشته‌ها بسیار یاری می‌دهد و با استفاده از نمادها و دستورهای موجود در Regular Expression می‌توانید متون را به منظور جست‌وجو، ویرایش، حذف و جایگذاری زیررشته‌هایی در داخل متون پردازش کنید. این بخش شامل جزئیات مجموعه کاراکترها، عملگرها و ساختارهایی است که می‌توانید از آنها برای تعریف و تعیین عبارت باقاعده استفاده کنید.

آنچه در این فصل خواهیم خواند:

۱- کاراکترها

شامل مجموعه‌ای از فرار کاراکتر (Character escape) و فراکاراکترها[۱](Meta character) است. فرارکاراکتر نشانه‌ای برای تجزیه کننده (Parser) عبارت باقاعده می‌باشد و مشخص می‌کند که کاراکتر حاضر یک عملگر نبوده و بایستی به عنوان یک کاراکتر معمولی، در تطبیق تفسیر گردد. فراکاراکترها[۲](Meta character) نیز در یک عبارت باقاعده به عنوان یک عملگر شناسایی می‌شوند.

---

۱ - کاراکتری که در متن یک برنامه یا جریانی از داده‌ها گنجانده می‌شود، اطلاعاتی درباره دیگر کاراکترها ارائه می‌نماید. "\" یک مثال ساده است که وقتی در رشته‌های زبان برنامه سازی C به کار برده می‌شود، نشان می‌دهد که کاراکترهای بعدی بخشی از یک escape sequence هستند و به زبان C امکان می‌دهند تا یک کاراکتر غیر گرافیکی را نمایش دهد. Escape sequence دنباله‌ای از کاراکترها هستند که معمولا با کاراکتر Esc (اسکی 27 یا هگزادسیمال 1B) آغاز شده و یک یا چند کاراکتر اضافی پس از آن می‌آیند. این دنباله‌ها از توالی معمول کاراکترها « فرار» کرده و دستور العمل یا فرمانی را به یک وسیله یا برنامه صادر می‌کنند.

۲ - کاراکتری که در متن یک برنامه یا جریانی از داده‌ها گنجانده می‌شود، اطلاعاتی درباره دیگر کاراکترها ارائه می‌نماید. "\" یک مثال ساده است که وقتی در رشته‌های زبان برنامه سازی C به کار برده می‌شود، نشان می‌دهد که کاراکترهای بعدی بخشی از یک escape sequence هستند و به زبان C امکان می‌دهند تا یک کاراکتر غیر گرافیکی را نمایش دهد. Escape sequence دنباله‌ای از کاراکترها هستند که معمولا با کاراکتر Esc (اسکی 27 یا هگزادسیمال 1B) آغاز شده و

۲- کلاس‌های کاراکتر(Character Classes)
مجموعه‌ای از کاراکترهای عبارت باقاعده است که زیررشته‌ای را برای تطبیق تعریف می‌کند.

۳- شمارنده‌های تکرار (Quantifiers)
محدوده مشخصی از کمینه و بیشینه شمار دفعات تکرار تطبیق یک زیررشته را به وسیله الگوی عبارت باقاعده تعیین می‌کنند. در این بخش با تطبیق‌های از نوع حریص (greedy) و تنبل (lazy) نیز آشنا خواهید شد.

٤- اعلان‌های تجزیه ناپذیر طول- صفر (Atomic Zero-Width Assertions)
اعلان‌های طول- صفر باعث می‌شوند موفقیت یا شکست عمل تطبیق به موقعیت کنونی عبارت باقاعده تجزیه کننده در رشته ورودی بستگی پیدا کند.

٥- ساختارهای گروه‌بندی (Grouping Constructs)
ساختارهای گروه‌بندی موجب می‌شوند الگوی عبارت باقاعده گروه‌هایی از زیرعبارت‌ها را به دام بیاندازد (Capture).

٦- ساختارهای متناوب (Alternation Constructs)
اجرای قسمتی از عبارت باقاعده تنها در صورت محقق شدن یا نشدن بخش دیگری صورت می‌پذیرد.

۷- ساختارهای ارجاع به عقب (Backreference Constructs)
درباره‌ی تغییردهنده‌های ارجاع به عقب عبارات باقاعده بحث می‌نماید.

۸- جایگذاری (Substitutions)
ساختارهای خاصی که در الگوهای جایگذاری استفاده می‌شوند.

۹- ساختارهای گوناگون (Miscellaneous Constructs)
شامل دیگر ساختارهایی است که رفتار تطبیقی عبارت باقاعده را تغییر می‌دهند.

۱۰- گزینه‌های عبارت باقاعده (Regular Expression Options)
مجموعه‌ای از Optionها که موجب تغییر رفتار تطبیقی الگوی عبارت باقاعده می‌شود.

یک یا چند کاراکتر اضافی پس از آن می‌آیند. این دنباله‌ها از توالی معمول کاراکترها « فرار» کرده و دستور العمل یا فرمانی را به یک وسیله یا برنامه صادر می‌کنند.

۲-۱ انواع کاراکترها

۲-۱-۱ فرار کاراکترها

کاراکترها در زبان عبارت باقاعده به سه دسته تقسیم می‌شوند:

۱. کاراکترهای معمولی (Ordinary Characters)
۲. کاراکترهای ویژه (Special Characters)

این کاراکترها با یک "\" به کار می‌روند. مانند کاراکتر "\a" که یک بوق هشدار را تطبیق می‌دهد.

۳. فراکاراکترها (Meta Characters)

بیشتر عملگرهای مهم زبان عبارات باقاعده از نوع فراکاراکتر هستند و هر کدام مفهوم خاصی دارند. اینک اگر قصد داشته باشیم فراکاراکتری را در معنای واقعی خود به کار ببریم، دراصطلاح گفته می‌شود کاراکتر می‌بایست Escape یا فرار داده شود. بدین معنی که کاراکتر از منظور خاصی که ماشین زبان عبارت باقاعده ممکن است از آن برداشت کند فرار کرده و معنی واقعی خود را دهد.

کاراکترهایی که در معنای واقعی خود به کار می‌روند شامل کاراکترهای معمولی و ویژه می‌باشند و عبارتند از:

۱. همه‌ی کاراکترهای عادی که خود را تطبیق می‌دهند، به جز فراکاراکترهای (,) , [,] , { , } , ^ , $, + , - , # , ? , | , \ , . , *
۲. '\a' یک بوق را تطبیق می‌دهد (زنگ هشدار). \u0007(مقدار هگزادسیمال)
۳. '\b' در یک عبارت باقاعده اگر درون یک جفت [] قرارگیرد backspace (\u0008) را تطبیق می‌دهد.

یادداشت:
b\ کاراکتر ویژه‌ای است. اگراین کاراکتر در داخل [] قرار نگیرد به مرز واژه –مرز میان w\ (شامل تمام کاراکترهای حرفی، عددی و زیر خط alphanumeric) و W\(شامل تمام کاراکترها به جز alphanumeric) – اشاره خواهد کرد. در الگوهای جایگذاری b\ همیشه به مفهوم backspace است.

٤. '\t' یک Tab را تطبیق می‌دهد. \u0009

۵. '\r' کاراکتر Carriage Returns[۱] (CR) را تطبیق می‌دهد. \u000D
۶. '\v' کاراکتر Vertical Tab را تطبیق می‌دهد. \u000B
۷. '\f' کاراکتر Form Feed[۲] (LF) را تطبیق می‌دهد. \u000C
۸. '\n' کاراکتر Line Feed[۳] یا New Line را تطبیق می‌دهد. \u000A
۹. '\e' یک Escape را تطبیق می‌دهد (برابر کلید Esc). \u001B
۱۰. '\040' یک کاراکتر اسکی در مبنای 8 (اکتال) را تطبیق می‌دهد (دست بالا تا سه رقم)؛ شماره‌های بدون شروع با صفرارجاع به عقب هستند. اگر تنها تک رقمی باشند یا اگر با شماره گروه به دام افتاده‌ای (captured group) مرتبط باشند ارجاع به عقب محسوب می‌شوند (به فصل هفتم ساختارهای ارجاع به عقب مراجعه کنید). برای نمونه، کاراکتر 040\ به یک space اشاره دارد.
۱۱. '\x20' کاراکترهای بر مبنای 16 (یا هگزا دسیمال) با پیشوند x\ مشخص می‌شوند و دقیقا شامل دو رقم هستند. برای نمونه عبارت باقاعده x0A\ (کد اسکی ۱۰) برابر با n\ برای تشخیص خط جدید است.
۱۲. '\cC' کاراکترهای کنترلی اسکی را تطبیق می‌دهد. برای بررسی فشرده شدن کلید کنترل همراه با کلید دیگری در متن به کار می‌رود. برای نمونه، عبارت باقاعده cZ\ فشرده شدن کلیدهای control-Z را تطبیق می‌دهد.
۱۳. '\u0020' کاراکتر یونیکد[۴] (Unicode) که به شکل هگزادسیمال به کار رفته است را تطبیق می‌دهد و باید دقیقا شامل چهار رقم باشد.

یادداشت:

Perl از پنج کاراکتر فرار به شکل {####}x\ (که '####' دنباله‌ای از اعداد هگزادسیمال است) به منظور تعیین یونیکد استفاده می‌کند. .NET Framework از این ویژگی پشتیبانی نمی‌کند و به جای آن از ویژگی توضیح داده شده در شماره ۱۳ جهت یونیکد استفاده می‌کند.

---

[۱] – بازگشت به ابتدای خط: یک کاراکتر کنترلی که به کامپیوتر یا چاپگر فرمان می‌دهد تا به ابتدای خط جاری بازگردد. مقدار دسیمال این کاراکتر در مجموعه کاراکترهای ASCI، 13 (هگزادسیمال 0D) است. معادل با کلید Enter.

[۲] – فرمانی در چاپگرها. به چاپگر اطلاع می‌دهد تا به بالای صفحه بعد انتقال یابد. مقدار دسیمال کاراکتر ویژه این کار در مجموعه کاراکترهای اسکی 12 (مقدار هگزا دسیمال 0C) است. چون هدف این کاراکتر ایجاد یک صفحه جدید است کاراکتر page-eject نیز نامیده می‌شود.

[۳] – کاراکتر کنترلی که به کامپیوتر فرمان می‌دهد تا بدون حرکت دادن مکان نما یا هد چاپ به خط بعد انتقال یابد. معادل اسکی آن به صورت دسیمال 10 و هگزا دسیمال آن 0A است.

[۴] – استاندار رمزگذاری 16 بیتی که کنسرسیوم یونیکد در بین سال‌های ۱۹۸۸ و ۱۹۹۱ ابداع نمود. این استاندارد با استفاده از دوبایت برای هر کاراکتر، امکان ارائه تقریبا تمامی زبان‌های مکتوب جهان را با یک مجموعه از کاراکترها فراهم می‌نماید. به پیوست مراجعه کنید.

۲-۱-۲ فراکاراکترها

بیشتر عملگرهای مهم زبان عبارات باقاعده تک کاراکترهای unescaped (فراری داده نشده) هستند. فراکاراکترها در زبان عبارت باقاعده به عنوان عملگر محسوب شده و در معنای واقعی خود به کار نمی‌روند و برای فراری دادن آنها می‌بایست از کاراکتر \ پیش از آنها استفاده کرد. تعریف کوتاهی از بیشتر فراکاراکترهایی که در عبارات باقاعده به کار می‌روند، در زیر آمده است:

۱. '\' :این کاراکتر وقتی به همراه کاراکتر دیگری که به عنوان کاراکتر فرار تشخیص داده نمی‌شود به کار می‌رود آن را تطبیق می‌دهد. از این کاراکتر بیشتر برای تطبیق فراکاراکتر استفاده می‌شود. برای مثال، در تطبیق کاراکتر * که معادل هگزا دسیمال آن برابر با \x2A است می‌بایست از '*\' استفاده کرد.

۲. '.': هر کاراکتری به جز newline یا \n را تطبیق می‌دهد. عبارت باقاعده .do کلمات dos و dog را تطبیق می‌دهد. اگر نیاز به تطبیق خود کاراکتر "." باشد می‌بایست پیش از آن یک \ قرار دهیم (\.).

۳. '-' (خط تیره): برای تعیین یک سلسله کاراکتر پی‌درپی همراه با کاراکترهای [ ] به کار می‌رود. مانند [0-9] که معادل [0123456789] ارزیابی می‌شود. فراکاراکتر '-' تنها در صورت نمایان شدن میان نمادهای '[' و ']' به عنوان فراکاراکتر شناخته می‌شود و خارج از آنها به عنوان یک کاراکتر ساده در نظر گرفته می‌شود.

٤. '^' (هشتک):این فراکاراکتر دو کار انجام می‌دهد. اگر در آغاز یک مجموعه (مانند [^aeiou]) قرارگیرد مقصود تطبیق هر کاراکتر به جز کاراکترهای موجود در مجموعه است و در صورتی که در خارج از مجموعه و در آغاز عبارت باقاعده قرارگیرد منظور تطبیق از آغاز رشته است.

٥. '*' (ستاره): صفر یا چند کاراکتر (عبارت) پیش از خود را تطبیق می‌دهد.

٦. '+' : یک یا چند کاراکتر (عبارت) پیش از خود را تطبیق می‌دهد.

۷. '?':این فراکاراکتر کاربردهای گوناگونی دارد که مهمترین آن تطبیق صفر یا یک کاراکتر پیش از خود است.

۸. '{' , '}' : مهمترین کاربرد این کاراکترها در شمارنده‌های تکرار و تعیین کمینه و بیشینه دفعات تکرار با استفاده از قالبندی {n,m} می‌باشد.

۹. '$'(Dollar): برای تطبیق پایان متن و در الگوهای جایگزینی استفاده می‌شود.

۱۰. '|'(Vertical bar خط عمود): در ساختارهای متناوب به کار می‌رود.

۱۱. '(' , ')' : در ایجاد زیرعبارت‌ها، ساختارهای گروه‌بندی، ارجاع به عقب و . . . استفاده می‌شود.

۱۲. '#' (Number sign): از این فراکاراکتر برای درج توضیحات درون الگوهایی که نوشته‌اید می‌توانید استفاده کنید.

نگران نباشید! رفته رفته با مطالعه این کتاب با این فراکاراکتر و نحوه استفاده از آنها بیشتر آشنا خواهید شد و خواهید توانست به آسانی از آنها در الگوهایتان استفاده کنید.

چند مثال

فرض کنیم زمانی را در اسناد خود برای یافتن شواهدی که Elvis هنوز زنده است (alive) سپری کرده‌اید. این سناریو را با استفاده از عبارات باقاعده به سرانجام می‌رسانیم تا دریابیم که آیا Elvis هنوز alive است یا خیر.

elvis برای یافتن elvis

عبارت **elvis** یک عبارت باقاعده معتبر از هر نظر است که برای یافتن دقیق یک توالی از کاراکترها می‌توان به کار برد.

RegEx

elvis

Sample Text and Search Results

Some people believe that Elvis is alive today.
It takes all kinds, I suppose. Elvis was sometimes
Known as "Elvis the Pelvis". Here is ELviS with
Weird capitalization.

یادداشت:

در این مثال RegexOptions.IgnoreCase انتخاب شده است. در .NET به آسانی می‌توانید با تنظیم گزینه‌ای (Option) – به گزینه‌های عبارت باقاعده (Regular Expressions Options) مراجعه کنید – با عنوان "چشم پوشی از بزرگ و کوچک بودن کاراکترها"(To ignore the case of characters) عبارت‌هایی مانند "ELVIS", "Elvis" یا "eLvIs" را نیز تطبیق دهید. همچنین می‌توانید به جای استفاده از IgnoreCase Option از الگوی [Ee][Ll][Vv][Ii][Ss] یا استفاده از ساختارهای گروه‌بندی خاصی، برای یافتن هر شکل از واژه‌ی elvis استفاده کنید. براکت‌های باز و بستهٔ '[' و ']' در بیشتر مواقع برای جست‌وجوی عبارات بدون توجه به بزرگ و کوچک بودن حروف استفاده می‌شود. با استفاده از این دو فراکاراکتر '[' و ']' می‌توانیم مجموعه‌ای از کاراکترها را تعریف کنیم که تنها یکی از آنها می‌تواند به عنوان کاراکتر بعدی نمایان شود. برای اطلاعات بیشتر می‌توانید به فصل دوم بخش کلاس کارارکترها، ساختارهای گروه‌بندی و Regular Expression Options مراجعه کنید.

نکته: می‌توانید از متن نمونه زیر در انجام عملیات تطبیق و تست الگوها استفاده کنید:

Tutorial Sample Data
Here is a space followed by backslash"\ " :
Some people believe that Elvis is alive today.
It takes all kinds, I suppose. Elvis was sometimes
Known as "Elvis the Pelvis". Here is ELviS with
Weird capitalization.
Realistically, repeated words are easy to
to compose and hard to find.
The President of IRAN visited Iraq and Qatar,
but he did not visit Quebec.
<body>Here is some text</body>
A short paragraph
aabab
SSN=
Phone numbers: 555-1212, 800 325-3535,
650 555-1212, (800)325-3535, (650) 555-1212,
650) 555-1212.
Parentheses (ab(cd)ef) (g((hijk)l)m)n)o
SSN=123-45-6789
555-44-3333 42-33-7777
Zip=95070 55901-1234 90210 902101
IP Addresses: 123.456.789.012 255.243.190.101
12.1.34.2 12.1.34
Mississippi
One last phone number, filling the line.
555-1234

در نتیجه جست‌وجو دقت کنید. متأسفانه در مثال بالا می‌بینید که الگوی **elvis** پنج کاراکتر آخر واژه‌ی "Pelvis" را هم تطبیق داده است. برای جلوگیری از این حالت می‌بایست الگوی خود را به شکل زیر تغییر دهید:

\belvis\b واژه‌ی **elvis** را به عنوان یک واژه‌ی مستقل و نه بخشی از یک واژه می‌یابد.

پیش‌تر درباره کاراکتر ویژه \b توضیح داده شد. \b را می‌توان نشانه‌گذار واژه نیز نامید. این الگو را با مثال پیشین تست نمایید:

RegEx

\belvis\b

Sample Text and Search Results

Some people believe that Elvis is alive today.
It takes all kinds, I suppose. Elvis was sometimes
Known as "Elvis the Pelvis". Here is ELviS with
Weird capitalization.

نتیجه‌ی بهتری را به دست آوردیم اما نمی‌دانیم که elvis هنوز alive است یا خیر. فرض کنیم می‌خواهید تمامی خطوطی را که در آن پس از واژه‌ی elvis واژه‌ی alive آمده است را پیدا کنید. دراینجا دو فراکاراکتر "." و "*" به کمک شما خواهد آمد. نقطه (Period or Dot)، هر کاراکتری به غیر از خط جدید (\n یا New Line یا Line Feed) را تطبیق می‌دهد. کاراکتر ستاره "*" به معنای تکرار عبارت (یا کاراکتر) پیش از خود بوده و می‌گوید: " تضمین می‌کنم هر شماری که بایسته است از عبارت پیش از خودم را برای شما تطبیق دهم". پس "*." به معنای تطبیق هر شمار کاراکتر به غیر از خط جدید است. ترکیب دو کاراکتر "*." یک فرمت ساده برای ایجاد یک عبارت باقاعده به مفهوم " عبارت elvisیی را که پس از آن در همان خط واژه‌ی alive آمده است را پیدا کن" می‌تواند ایجاد کند. پس:

\belvis\b.*\balive\b متنی را که دارای واژه‌ی elive است و در پی آن alive است می‌یابد.

RegEx

\belvis\b.*\balive\b

Sample Text and Search Results

Some people believe that Elvis is alive today.
It takes all kinds, I suppose. Elvis was sometimes
Known as "Elvis the Pelvis". Here is ELviS with
Weird capitalization.

نتیجه مشخص شد. Elvis هنوز alive است. ما توانستیم با شمار کمی کاراکتر ویژه شروع به ساخت عبارت باقاعده مفید کنیم. این در حالی است که این کار پیش از این برای ما بسیار سخت و دشوار می‌نمود.

اجازه بدهید نمونه‌ی دیگری را امتحان کنیم:

فرض کنیم صفحه وب شما شماره تلفن‌های هفت رقمی را گردآوری می‌کند و می‌خواهید تشخیص بدهید که شماره تلفن وارد شده توسط مشتری در قالب صحیحی وارد شده است یا خیر (xxx-" xxxx" که "x" عدد است).

عبارت زیر کل متن را برای یافتن چنین رشته‌ای جست‌وجو می‌کند.

\b\d\d\d-\d\d\d\d شماره تلفن‌های هفت رقمی را پیدا می‌کند.

RegEx

\b\d\d\d-\d\d\d\d

Sample Text and Search Results

Phone numbers: 555-1212, 800 325-3535,

650 555-1212, (800)325-3535, (650) 555-1212,
650) 555-1212.
Mississippi
One last phone number, filling the line.
555-1234

هر "\d" به معنای "تطبیق یک عدد تک رقمی" است. کاراکتر"-" مفهوم خاصی نداشته و به عنوان یک کاراکتر عادی تفسیر شده و یک خط تیره (Dash) را تطبیق می‌دهد.

برای پرهیز از تکرارهای بی‌مورد و خسته کننده می‌توانیم از نمادگذاری‌های مختصرنویسی (Shorthand Notation) استفاده کنیم و در این مورد می‌توانیم عبارت باقاعده بالا را به شکل زیر به کار ببریم:

\b\d{3}-\d{4} شیوه‌ای بهتر برای یافتن شماره تلفن‌های هفت رقمی.

RegEx

\b\d{3}-\d{4}

Sample Text and Search Results

Phone numbers: 555-1212, 800 325-3535,

650 555-1212, (800)325-3535, (650) 555-1212,
650) 555-1212.
Mississippi
One last phone number, filling the line.

555-1234

"**{3}**" به همراه "**\d**" به معنای تکرار کاراکتر پیشین به شمار دقیقا سه بار است. نتیجه همان نتیجه مثال شماره ٤ است.

باید شماری از کاراکترها که معانی خاصی دارند را بشناسید. پیش از این با "**\b**"، "."، "*" و "**\d**" آشنا شدید. برای تطبیق هر کاراکتر فضای خالی (whitespace) مانند space، tab، newlines(\n) و Carriage Returns(**\r**) از "**\s**" استفاده کنید. "**\w**" هر کاراکتر alphanumeric را تطبیق می‌دهد. برای یادگیری بیشتر بهتر است چند نمونه دیگر را مطرح کنیم:

\ba\w*\b واژگانی را که با کاراکتر **a** شروع می‌شود می‌یابد.

RegEx

\ba\w*\b

Sample Text and Search Results

Try clicking the "Run Match" button (or F5) to see what
happens. After a successful match click in the results box
to highlight the matched text. Click the "Replace" button
using the "Dates" example. Notice in the results box that the
format of all the dates has been altered. Using the
"Expression Library" tab double-click another regular
expression and "Run Match."

این عبارت باقاعده به دنبال کاراکتر a که آغاز آن **\b**، پس از آن هر شمار کاراکتر alphanumeric (**\w***) و در پایان آن **\b** باشد می‌گردد.

\d+ رشته‌های عددی را تطبیق می‌دهد.

عملکرد فراکاراکتر "+" همانند "*" است با این تفاوت که دست کم به یک تکرار از کاراکتر یا عبارت پیش از خود نیاز دارد.

RegEx

\d+

Sample Text and Search Results

Netherlands Postal Codes:
1234 ZA
5678 KL5
0123 AZ -> Illegal number, must be 1000-9999

5432 ABQ -> Too many characters

\b\w{6}\b واژگان شش کاراکتری را پیدا می‌کند.

RegEx

\b\w{6}\b

Sample Text and Search Results

Try clicking the "Run Match" button (or F5) to see what happens. After a successful match click in the results box to highlight the matched text. Click the "Replace" button using the "Dates" example. Notice in the results box that the format of all the dates has been altered. Using the "Expression Library" tab double-click another regular expression and "Run Match."

Click "Show the Builder" (or Ctrl+D) to start designing your own regular expressions; or enter them directly in the regular expression box. Enter the text you want to search in the sample text box.

Expand nodes in the "Regex Analyzer" to see a description of the current regular expression. Right-click a node to edit the expression.

به نظر می‌رسد که وقت آن رسیده شما هم مثال‌ها و عبارت‌های خود را نوشته و درستی آنها را بیازمایید.

Expresso ۴-۱-۲

اگر باور دارید یادگیری عبارات باقاعده دشوار نیست، فرض من بر این خواهد بود که یا شما بسیار باهوش هستید یا اینکه از سیاره‌ی دیگری آمده‌اید. اشتباه نکنید، نحو می‌تواند هر شخصی را که پی‌درپی از عبارات باقاعده استفاده نمی‌کند گول بزند. این مسئله، همواره موجب ایجاد اشتباه‌های فراوانی می‌شود و این موضوع موجب شد تا شما را با یک ابزار ساده برای ایجاد و تست عبارات باقاعده آشنا کنیم. چنین ابزارهایی بسیار فراوانند ولی Expresso را به شما معرفی می‌کنیم. می‌توانید آخرین نسخه این نرم‌افزار را با مراجعه به نشانی http://www.ultrapico.com دانلود کنید.