Подробные решения
на восьми языках программирования

Включает учебное руководство



Регулярные выражения

Сборник рецептов

СИМВОЛ®

O'REILLY®

Ян Гойвертс
Стивен Левитан

Regular Expressions Cookbook

Jan Goyvaerts, Steven Levithan

O'REILLY®

Регулярные выражения
Сборник рецептов

Ян Гойвертс, Стивен Левитан

Символ®

Санкт-Петербург — Москва
2010

Ян Гойвертс, Стивен Левитан

Регулярные выражения Сборник рецептов

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Редактор	*Ю. Бочина*
Корректор	*С. Минин*
Верстка	*К. Чубаров*

Гойвертс Я., Левитан С.

Регулярные выражения. Сборник рецептов. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 608 с., ил.

ISBN 978-5-93286-181-3

Сборник содержит более 100 рецептов, которые помогут научиться эффективно оперировать данными и текстом с применением регулярных выражений. Книга знакомит читателя с функциями, синтаксисом и особенностями этого важного инструмента в различных языках программирования: C#, Java, JavaScript, Perl, PHP, Python, Ruby и VB.NET. Предлагаются пошаговые решения наиболее часто встречающихся задач: работа с адресами URL и путями в файловой системе, проверка и форматирование ввода пользователя, обработка текста, а также обмен данными и работа с текстами в форматах HTML, XML, CSV и др.

Данное руководство поможет как начинающему, так и уже опытному специалисту расширить свои знания о регулярных выражениях, познакомиться с новыми приемами, узнать все тонкости работы с ними, научиться избегать ловушек и ложных совпадений. Освоив материал этой книги, вы сможете полнее использовать все те возможности, которые предоставляет умелое применение регулярных выражений, и тем самым сэкономить свое время.

ISBN 978-5-93286-181-3
ISBN 978-0-596-52068-7 (англ)

© Издательство Символ-Плюс, 2010
Authorized translation of the English edition © 2009 O'Reilly. This translation is published and sold by permission of O'Reilly, Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 30.10.2009. Формат 70×100 $^{1}/_{16}$. Печать офсетная.
Объем 38 печ. л. Тираж 2000 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Предисловие 11

1. Введение в регулярные выражения 19

Определение регулярных выражений 19
Поиск с заменой с помощью регулярных выражений 25
Инструменты для работы с регулярными выражениями 27

2. Основные навыки владения регулярными выражениями 48

2.1. Соответствие литеральному тексту 49
2.2. Соответствие непечатным символам 52
2.3. Сопоставление с одним символом из нескольких 54
2.4. Сопоставление с любым символом 59
2.5. Сопоставление в начале и/или в конце строки 62
2.6. Сопоставление с целыми словами 68
2.7. Кодовые пункты Юникода, свойства, блоки и алфавиты 71
2.8. Сопоставление с одной из нескольких альтернатив 85
2.9. Группы и сохранение части совпадения 87
2.10. Повторный поиск соответствия с ранее совпавшим текстом 91
2.11. Сохранение и именованные части совпадения 93
2.12. Повторение части регулярного выражения определенное число раз 97
2.13. Выбор минимального или максимального числа повторений 100
2.14. Устранение бесполезных возвратов 104
2.15. Предотвращение бесконтрольных повторений 108
2.16. Проверка соответствия без включения его в общее соответствие 111
2.17. Совпадение с одной из двух альтернатив по условию 119
2.18. Добавление комментариев в регулярные выражения 122
2.19. Вставка текстового литерала в замещающий текст 124
2.20. Вставка совпадения с регулярным выражением в замещающий текст 128
2.21. Вставка части совпадения с регулярным выражением в замещающий текст 129
2.22. Вставка контекста совпадения в замещающий текст 133

3. Программирование с применением регулярных выражений 135

Языки программирования и диалекты регулярных выражений 135
3.1. Литералы регулярных выражений в исходных текстах 142
3.2. Импортирование библиотеки регулярных выражений 149
3.3. Создание объектов регулярных выражений 151
3.4. Установка параметров регулярных выражений 159
3.5. Проверка возможности совпадения в пределах испытуемой строки 168
3.6. Проверка совпадения со всей испытуемой строкой 175
3.7. Извлечение текста совпадения 181
3.8. Определение позиции и длины совпадения 188
3.9. Извлечение части совпавшего текста 194
3.10. Извлечение списка всех совпадений 202
3.11. Обход всех совпадений в цикле 208
3.12. Проверка полученных совпадений в программном коде 215
3.13. Поиск совпадения внутри другого совпадения 219
3.14. Замена всех совпадений 224
3.15. Замена совпадений с повторным использованием частей совпадений 232
3.16. Замена совпадений фрагментами, сгенерированными в программном коде 238
3.17. Замещение всех совпадений внутри совпадений с другим регулярным выражением 245
3.18. Замещение всех совпадений между совпадениями с другим регулярным выражением 247
3.19. Разбиение строки 253
3.20. Разбиение строки, сохранение совпадений с регулярным выражением 264
3.21. Построчный поиск 269

4. Проверка и форматирование 274

4.1. Проверка адресов электронной почты 274
4.2. Проверка и форматирование телефонных номеров 282
4.3. Проверка международных телефонных номеров 288
4.4. Проверка дат в традиционных форматах 290
4.5. Точная проверка дат в традиционных форматах 295
4.6. Проверка времени в традиционных форматах 300
4.7. Проверка даты и времени в формате ISO 8601 303
4.8. Ограничение возможности ввода алфавитно-цифровыми символами 308
4.9. Ограничение длины текста 312
4.10. Ограничение числа строк в тексте 317

4.11. Проверка утвердительных ответов ... 323
4.12. Проверка номеров социального страхования ... 325
4.13. Проверка номеров ISBN ... 328
4.14. Проверка почтовых индексов ... 337
4.15. Проверка почтовых индексов, используемых в Канаде ... 338
4.16. Проверка почтовых индексов, используемых в Великобритании ... 339
4.17. Поиск адресов, содержащих номер почтового ящика ... 339
4.18. Преобразование имен из формата «имя фамилия» в формат «фамилия, имя» ... 341
4.19. Проверка номеров кредитных карт ... 346
4.20. Европейские регистрационные номера плательщиков НДС ... 353

5. Слова, строки и специальные символы ... 361

5.1. Поиск определенного слова ... 361
5.2. Поиск любого слова из множества ... 364
5.3. Поиск похожих слов ... 367
5.4. Поиск любых слов, за исключением некоторых ... 371
5.5. Поиск любого слова, за которым не следует указанное слово ... 373
5.6. Поиск любого слова, которому не предшествует определенное слово ... 375
5.7. Поиск близко расположенных слов ... 379
5.8. Поиск повторяющихся слов ... 387
5.9. Удаление повторяющихся строк ... 389
5.10. Совпадение с полными строками, содержащими определенное слово ... 395
5.11. Совпадение с полными строками, не содержащими определенное слово ... 397
5.12. Удаление ведущих и завершающих пробельных символов ... 398
5.13. Замена повторяющихся пробельных символов единственным пробелом ... 402
5.14. Экранирование метасимволов регулярных выражений ... 403

6. Числа ... 409

6.1. Целые числа ... 409
6.2. Шестнадцатеричные числа ... 413
6.3. Двоичные числа ... 416
6.4. Удаление ведущих нулей ... 418
6.5. Числа в определенном диапазоне ... 419
6.6. Шестнадцатеричные числа в определенном диапазоне ... 427
6.7. Вещественные числа ... 429

6.8. Числа с разделителями групп разрядов 433
6.9. Римские числа 434

7. URL, пути и адреса в Интернете 438

7.1. Проверка адресов URL 438
7.2. Поиск адресов URL в тексте 442
7.3. Поиск в тексте адресов URL, заключенных в кавычки 444
7.4. Поиск в тексте адресов URL, заключенных в скобки 446
7.5. Преобразование адресов URL в ссылки 448
7.6. Проверка строк URN 449
7.7. Проверка универсальных адресов URL 452
7.8. Извлечение схемы из адреса URL 458
7.9. Извлечение имени пользователя из URL 460
7.10. Извлечение имени хоста из URL 462
7.11. Извлечение номера порта из URL 465
7.12. Извлечение пути из адреса URL 467
7.13. Извлечение строки запроса из URL 471
7.14 Извлечение фрагмента из URL 472
7.15. Проверка доменных имен 473
7.16. Сопоставление с адресами IPv4 476
7.17. Сопоставление с адресами IPv6 479
7.18. Проверка путей в Windows 495
7.19. Выделение элементов путей в Windows 498
7.20. Извлечение буквы устройства из путей в Windows 503
7.21. Извлечение имени сервера и разделяемого ресурса из пути в формате UNC 504
7.22. Извлечение имен папок из путей в Windows 506
7.23. Извлечение имени файла из пути Windows 508
7.24. Извлечение расширения имени файла из пути Windows 510
7.25. Удаление недопустимых символов из имен файлов 511

8. Разметка и обмен данными 513

8.1. Поиск тегов XML 521
8.2. Заменить тег <b> тегом 541
8.3. Удаление всех XML-подобных тегов, за исключением и 545
8.4. Сопоставление с именами XML 549
8.5. Преобразование простого текста в HTML добавлением тегов <p> и
 557
8.6. Поиск определенных атрибутов в XML-подобных тегах 560
8.7. Добавление атрибута cellspacing в теги <table>, где этот атрибут отсутствует 566

8.8. Удаление XML-подобных комментариев 569
8.9. Поиск слов в XML-подобных комментариях 574
8.10. Изменение разделителя, используемого в файлах CSV 579
8.11. Извлечение полей CSV из определенного столбца 584
8.12. Сопоставление с заголовком раздела в файле INI 588
8.13. Сопоставление с разделом в файле INI 589
8.14. Сопоставление с парами имя-значение в файле INI 591

Алфавитный указатель 593

Предисловие

В течение последнего десятилетия наблюдался неуклонный рост интереса к регулярным выражениям. На сегодняшний день все популярные языки программирования включают мощные библиотеки поддержки этого инструмента или даже имеют реализацию этой поддержки, встроенной непосредственно в сам язык. Многие разработчики с успехом используют поддержку регулярных выражений, предоставляя пользователям своих приложений возможности поиска или фильтрации данных с их помощью. Регулярные выражения присутствуют практически везде.

Было издано множество книг, помогающих освоить регулярные выражения. Большинство из них прекрасно справляются с задачей описания их синтаксиса, дополнительно предоставляя некоторые примеры и справочную информацию. Но пока еще не существует книги, которая демонстрировала бы решения, основанные на использовании регулярных выражений, примененные к широкому кругу практических задач, связанных с обработкой текста, возникающих в разнообразных интернет-приложениях. Мы, Стив и Ян, решили данной книгой восполнить этот пробел.

Мы стремились показать, как можно применять регулярные выражения в ситуациях, в которых тем, кто имеет недостаточный опыт работы с этим инструментом, может показаться, что их применить невозможно, а искушенные программисты могли бы посчитать, что для решения подобных задач их не стоило использовать.

Так как в настоящее время регулярные выражения получили повсеместную поддержку, они часто оказываются удобным инструментом, который может использоваться конечными пользователями даже без посредничества программистов. Да и сами программисты могут сэкономить время, используя регулярные выражения для решения задач извлечения и изменения информации, не тратя сил и времени на создание соответствующего процедурного программного кода или изучение описаний возможностей различных вспомогательных библиотек.

Проблемы, связанные с многообразием версий

Как и для любых других инструментов, популярных в мире информационных технологий, существует множество различных реализаций поддержки регулярных выражений, обладающих различной степенью совместимости. В результате это привело к появлению множества различных *диалектов* регулярных выражений, которые не всегда одинаково интерпретируют одно и то же регулярное выражение, если вообще способны интерпретировать его.

Во многих книгах отмечается наличие диалектов и указываются некоторые из их отличий. Но нередко какие-то диалекты в определенных ситуациях вообще не упоминаются – особенно если в этих диалектах отсутствуют какие-то возможности – тогда как можно было бы показать альтернативное решение или обходной путь. Это становится препятствием, когда приходится работать с различными диалектами регулярных выражений в различных приложениях или языках программирования.

Отдельные заявления, встречающиеся в литературе, такие как «в настоящее время все используют регулярные выражения в стиле языка Perl», к сожалению, выводят из рассмотрения широкий круг несовместимостей. Даже между пакетами реализации регулярных выражений «в стиле языка Perl» имеются существенные различия, к тому же сам язык Perl непрерывно развивается дальше. Чрезмерно упрощенное освещение проблемы может приводить к тому, что программист будет проводить часы бесплодных поисков в отладчике, вместо того чтобы исследовать особенности используемой реализации регулярных выражений. И даже когда он обнаружит, что та или иная особенность отсутствует в реализации, он не будет знать, как обойти этот недостаток.

Данная книга является первой, где на всем протяжении параллельно рассматриваются наиболее распространенные и обладающие богатыми возможностями диалекты регулярных выражений,.

Для кого написана эта книга

Издание предназначено для тех, кто постоянно работает с текстовой информацией на компьютере, то есть обрабатывает огромные массивы документов, работает с текстом в текстовых редакторах или разрабатывает программное обеспечение, реализующее анализ и управление текстовой информацией. Регулярные выражения являются превосходным инструментом для решения таких задач. Эта книга содержит все, что необходимо знать об этом инструменте. От читателя не требуется иметь какой-либо опыт работы с регулярными выражениями, потому что в книге объясняются даже самые основные их положения.

Если вы уже знакомы с регулярными выражениями, то найдете здесь множество подробностей, о которых часто умалчивают другие книги или электронные издания. Если вам уже приходилось сталкиваться с ситуацией, когда одно и то же регулярное выражение работает в одном приложении, но не работает в другом, вы найдете весьма ценным подробное описание семи диалектов наиболее распространенных регулярных выражений, представленное в этой книге. Мы организовали книгу как сборник рецептов, поэтому вы можете начинать чтение непосредственно с интересующей вас темы. А если вы прочтете это издание от корки до корки, вы станете мастером регулярных выражений мирового уровня.

Эта книга даст вам все необходимые сведения о регулярных выражениях, даже если вы не занимаетесь программированием. Если вам требуется использовать регулярные выражения в текстовом редакторе, в инструменте поиска или в каком-либо другом приложении, где имеется поле ввода с надписью «regex» (регулярное выражение), вы можете приступать к чтению этой книги, даже не обладая опытом программирования. В большинстве рецептов, что приводятся в книге, даны решения, основанные исключительно на применении одного или нескольких регулярных выражений.

Для программистов в главе 3 приводятся полные сведения, необходимые для реализации регулярных выражений в исходном программном коде. Данная глава предполагает, что читатель обладает знаниями об основных особенностях выбранного им языка программирования, но не требует наличия опыта использования регулярных выражений в исходном программном коде.

Охватываемые технологии

.NET, Java, JavaScript, PCRE, Perl, Python и Ruby – это не просто броские названия. Они представляют семь диалектов регулярных выражений, рассматриваемых в данной книге. Мы в равной степени уделим внимание всем семи диалектам. В особенности мы позаботимся о том, чтобы указать все несовместимости, существующие между ними, которые нам удалось обнаружить.

Глава, имеющая отношение к программированию (глава 3) , содержит листинги исходных текстов на языках C#, Java, JavaScript, PHP, Perl, Python, Ruby и VB.NET. Здесь также для каждого рецепта приводятся решения и их описания для всех восьми языков. Так как это приводит к тому, что в главе многое повторяется не один раз, вы легко можете пропускать обсуждения, связанные с языками программирования, которые вас не интересуют, – достаточно следить за тем, что относится к выбранному вами языку.

Организация книги

В первых трех главах описываются наиболее ценные инструменты и даются основные сведения, позволяющие начать использовать регулярные выражения. Каждая последующая глава представляет различные регулярные выражения, подробно исследуя какой-то один круг задач обработки текста.

Глава 1 «Введение в регулярные выражения» объясняет назначение регулярных выражений и знакомит с набором инструментов, способных облегчить их изучение, создание и отладку.

Глава 2 «Основные навыки владения регулярными выражениями» описывает каждый элемент и каждую особенность регулярных выражений, а также содержит рекомендации по их эффективному использованию.

Глава 3 «Программирование с применением регулярных выражений» демонстрирует приемы программирования и содержит листинги с исходными текстами на всех языках программирования, описываемых в этой книге, где используются регулярные выражения.

Глава 4 «Проверка и форматирование» содержит рецепты обработки информации, вводимой пользователем, такой как даты, номера телефонов и почтовые индексы, используемые в разных странах.

Глава 5 «Слова, строки и специальные символы» рассматривает наиболее типичные задачи обработки текста, такие как проверка наличия или отсутствия в строках определенных слов.

Глава 6 «Числа» показывает, как определять целые и вещественные числа при вводе в различных форматах.

Глава 7 «URL, пути и адреса в Интернете» демонстрирует, как анализировать и манипулировать строками, часто используемыми в Интернете и в системе Windows в процессе поиска.

Глава 8 «Разметка и обмен данными» охватывает приемы манипулирования текстом в форматах HTML, XML, CSV (comma-separated values – данные, разделенные запятыми) и конфигурационными файлами в формате INI.

Типографские соглашения

В этой книге используются следующие типографские соглашения:

Курсив

Курсивом выделяются новые термины, адреса URL, адреса электронной почты, имена и расширения файлов.

`Моноширинный шрифт`

Используется для представления листингов программ; элементов программного кода, таких как имена переменных или функций;

значений, возвращаемых в результате применения регулярных выражений, а также обрабатываемых элементов или текста, к которым применяется регулярное выражение. Это может быть содержимое поля ввода в приложении, файла на диске или содержимое строковой переменной.

Моноширинный курсивный шрифт

Используется для представления текста, который должен замещаться значениями, вводимыми пользователем, или для представления значений, определяемых контекстом.

‹Регулярное•выражение›

Так представляются регулярные выражения сами по себе или в том виде, в каком они вводятся в поле ввода в приложениях. Пробелы внутри регулярных выражений обозначаются кружочком, за исключением тех случаев, когда количество пробелов не имеет большого значения.

«Замещаемый•текст»

Так будет представлен текст, соответствующий регулярному выражению, который будет замещен при выполнении операции поиска с заменой. Пробелы в замещаемом тексте обозначаются кружочком.

Совпавший текст

Так будет представлен фрагмент текста, соответствующий регулярному выражению.

...

Многоточие внутри регулярных выражений указывает место, которое должно быть заполнено вами, прежде чем вы сможете воспользоваться ими. Сопутствующий текст поясняет, что должно быть подставлено вместо многоточия.

CR, LF и CRLF

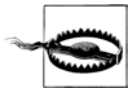
Обозначения CR, LF и CRLF в рамках представляют фактические разрывы текстовых строк, а не специальные экранированные последовательности, такие как \r, \n и \r\n. Такие строки могут создаваться в многострочных полях ввода нажатием клавиши Enter или при использовании многострочных строковых констант в исходных текстах программ, таких как строковые литералы в языке C#, или в строках, окруженных тройными кавычками, в языке Python.

↵

Эту стрелку вы можете увидеть на клавише Return или Enter, имеющейся на клавиатуре. Она говорит о том, что текст был перенесен исключительно для того, чтобы уместить его по ширине книжной страницы. При вводе такого текста не нужно нажимать клавишу Enter – просто продолжайте ввод в той же самой строке.



Этот значок отмечает советы, предложения и примечания общего характера.



Этот значок отмечает предупреждения и предостережения.

Использование программного кода примеров

Данная книга призвана оказать помощь в решении ваших задач. Вообще вы можете свободно использовать примеры программного кода из этой книги в своих приложениях и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить значительные отрывки программного кода. Например, если вы разрабатываете программу и используете в ней несколько отрывков программного кода из книги, вам не нужно обращаться за разрешением. Однако в случае продажи или распространения компакт-дисков с примерами из этой книги вам необходимо получить разрешение от издательства O'Reilly. Для цитирования данной книги или примеров из нее, при ответах на вопросы не требуется получение разрешения. При включении существенных объемов программного кода примеров из этой книги в вашу документацию вам необходимо будет получить разрешение издательства.

Мы приветствуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN. Например: «Regular Expressions Cookbook by Jan Goyvaerts and Steven Levithan. Copyright 2009 Jan Goyvaerts and Steven Levithan, 978-0-596-2068-7».

За получением разрешения на использование значительных объемов программного кода примеров из этой книги обращайтесь по адресу *permissions@oreilly.com*.

Safari® Books Online



Если на обложке технической книги есть пиктограмма «Safari® Books Online», это означает, что книга доступна в Сети через O'Reilly Network Safari Bookshelf.

Safari предлагает намного лучшее решение, чем электронные книги. Это виртуальная библиотека, позволяющая без труда находить тысячи лучших технических книг, вырезать и вставлять примеры кода, загружать главы и находить быстрые ответы, когда требуется наиболее верная и свежая информация. Она свободно доступна по адресу *http://safari.oreilly.com*.

Как с нами связаться

С вопросами и предложениями, касающимися этой книги, обращайтесь в издательство:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в Соединенных Штатах Америки или в Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Список опечаток, файлы с примерами и другую дополнительную информацию вы найдете на сайте книги:

http://www.regexcookbook.com

или

http://oreilly.com/catalog/9780596520687

Свои пожелания и вопросы технического характера отправляйте по адресу:

bookquestions@oreilly.com

Дополнительную информацию о книгах, обсуждения, Центр ресурсов издательства O'Reilly вы найдете на сайте:

http://www.oreilly.com

Благодарности

Мы выражаем свою признательность Энди Ораму (Andy Oram), нашему редактору из издательства O'Reilly Media, Inc., за то, что он сопровождал этот проект от начала и до конца. Мы также благодарим Джеффри Фридла (Jeffrey Friedl), Зака Грента (Zak Greant), Николая Линдберга (Nikolaj Lindberg) и Яна Морса (Ian Morse), за техническое рецензирование, что позволило сделать эту книгу более последовательной и точной.

1

Введение в регулярные выражения

Раз уж вы открыли эту книгу, то наверняка захотите ввести в свой программный код какие-то из несуразных строк, состоящих из круглых скобок и вопросительных знаков, которые вам встретятся в этих главах. Если вы готовы включиться в игру, будьте нашим гостем: практические регулярные выражения и их описания вы найдете в главах с 4 по 8.

Однако и начальные главы помогут вам сэкономить немало времени в будущем. Например, в этой главе будут представлены некоторые утилиты (часть из них создана Яном, одним из авторов книги), которые позволят вам проверять и отлаживать регулярные выражения перед включением их в программный код, где отыскать возможные ошибки будет намного труднее. Кроме того, начальные главы покажут вам, как использовать различные особенности и возможности регулярных выражений, способные облегчить вашу жизнь, дадут понимание регулярных выражений с точки зрения их оптимизации, покажут тонкие отличия в том, как обрабатываются регулярные выражения в разных языках программирования и даже в различных версиях одного и того же языка.

Мы приложили немалые усилия при подготовке этих базовых сведений в надежде на то, что вы начнете изучение с них или, если вы уже столкнулись с проблемами, связанными с использованием регулярных выражений, то приняли решение со всем этим разобраться.

Определение регулярных выражений

В контексте этой книги под *регулярными выражениями* понимаются определенные виды текстовых шаблонов, которые могут использоваться во многих современных приложениях и языках программирования. Вы можете использовать их, чтобы убедиться, что введенный текст вписывается в заданный шаблон; отыскивать внутри больших объемов текста фрагменты, соответствующие шаблону; замещать текст, соответствующий шаблону, другим текстом или переупорядочивать фраг-

менты совпавшего текста; разбивать блоки текста на списки более мелких фрагментов или даже делать не столь разумные вещи. Эта книга поможет вам точно понимать, что вы делаете, и избегать ошибок.

История появления термина «регулярное выражение»

Термин «регулярное выражение» пришел из математики и теоретической информатики, где он отражает свойство математических выражений, называемое *регулярностью*. Такие выражения могут быть реализованы программно с использованием детерминированных конечных автоматов (ДКА). ДКА – это механизм с конечным числом состояний, не поддерживающий возможность возврата (backtracking).

Текстовые шаблоны, использовавшиеся в самых ранних версиях инструментов *grep*, были регулярными выражениями в математическом смысле. Несмотря на то, что имя прижилось, современные регулярные выражения в стиле языка Perl фактически не являются регулярными выражениями в математическом смысле. Они реализованы с применением механизма недетерминированных конечных автоматов (НКА). Вы вскоре поближе познакомитесь с понятием возврата. Программистам-практикам достаточно усвоить из этого примечания следующее: высоколобым компьютерным теоретикам приходится мириться с тем, что чистые теоретические модели заменяются технологиями, более полезными в реальном мире.

При грамотном использовании регулярные выражения способны упростить задачи программирования и обработки текстов, а также обеспечить решение многих задач, которые вообще не могли бы быть решены без использования регулярных выражений. Вам могут потребоваться десятки, если не сотни, строк процедурного программного кода, чтобы извлечь все адреса электронной почты из документа, и такой код будет очень сложно написать и сопровождать. Однако с применением регулярных выражений, как показано в рецепте 4.1, решение этой задачи может быть реализовано всего лишь парой строк кода, а возможно, даже и одной строкой.

Но если попытаться взвалить слишком много на единственное регулярное выражение или задействовать регулярные выражения там, где они малопригодны, вы поймете, почему некоторые говорят[1]:

[1] Историю этой цитаты Джеффри Фридл прослеживает в своем блоге по адресу: *http://regex.info/blog/2006-09-15/247*.

> Когда кто-то, сталкиваясь с проблемой, думает: «А-а, понимаю, здесь поможет регулярное выражение», то в результате он получает две проблемы.

Вторая проблема, которую получают те, кто так думает, состоит в том, что они не читали справочное руководство, которое вы держите в руках. Продолжайте читать его. Регулярные выражения – это очень мощный инструмент. Если решаемая задача связана с обработкой или извлечением текста на компьютере, то владение регулярными выражениями поможет вам сэкономить массу времени.

Множество диалектов регулярных выражений

Итак, заголовком предыдущего раздела мы ввели вас в заблуждение. Мы не дали определение регулярных выражений. И мы не можем сделать этого. В мире не существует официального стандарта, который точно определял бы, какой текстовый шаблон является регулярным выражением, а какой – нет. Как вы можете себе представить, каждый, кто проектирует свой язык программирования, и каждый разработчик приложений, обрабатывающих тексты, имеет свое представление о том, какими должны быть регулярные выражения. Поэтому в результате мы имеем массу *диалектов* регулярных выражений.

К счастью, большинство проектировщиков и разработчиков оказываются достаточно ленивы. Зачем создавать нечто совершенно новое, когда можно скопировать то, что уже существует? В результате почти все современные диалекты регулярных выражений, включая и те, что рассматриваются в этой книге, ведут свою историю от языка программирования Perl. Мы называем эти диалекты *регулярными выражениями в стиле языка Perl*. Синтаксис этих диалектов прост и главным образом совместим, хотя и не полностью.

Писатели тоже ленивы. Единственное регулярное выражение мы обычно называем *regex* или *regexp*, а термин regexes используем для обозначения множественного числа.

Диалекты регулярных выражений не связаны жестко с какими-то определенными языками программирования. Языки сценариев обычно имеют свои собственные, встроенные диалекты регулярных выражений. Другие языки программирования опираются на использование библиотек, реализующих поддержку регулярных выражений. Некоторые библиотеки доступны для нескольких языков программирования, а некоторые языки программирования предоставляют возможность выбора из нескольких библиотек.

Эта вводная глава рассматривает диалекты регулярных выражений без всякой связи с языками программирования. Листинги с исходными текстами начнут появляться в главе 3; в разделе «Языки программирования и диалекты регулярных выражений» вы узнаете, с какими диалектами вам предстоит работать. А пока оставим в стороне все вопросы,

связанные с программированием. Инструменты, перечисленные в следующем разделе, обеспечивают простоту исследования синтаксиса регулярных выражений, «обучая в процессе работы».

Диалекты регулярных выражений, рассматриваемые в этой книге

Для этой книги мы отобрали наиболее распространенные диалекты регулярных выражений, используемых в наши дни. Все они являются диалектами *регулярных выражений языка Perl*. Некоторые диалекты имеют более широкие возможности по сравнению с другими. Но, как правило, когда два диалекта обладают одинаковым набором возможностей, они используют один и тот же синтаксис. Мы укажем на некоторые из досадных несовместимостей, с которыми нам пришлось столкнуться.

Все рассматриваемые здесь диалекты являются частью какого-либо языка программирования или библиотеки, которые продолжают активно разрабатываться. Из списка диалектов, что приводится ниже, вы узнаете, какие версии рассматриваются в этой книге. На протяжении всей книги мы будем упоминать диалекты без их версий, если представленные регулярные выражения одинаково работают во всех диалектах, что случается практически всегда. Помимо исправлений ошибок, проявляющихся в крайних случаях, диалекты обычно не изменяются, за исключением добавления новых особенностей и введения новых синтаксических конструкций, которые в предыдущих версиях трактовались как ошибка:

Perl

То, что язык Perl имеет встроенную поддержку регулярных выражений, является главной причиной их популярности в наши дни. В этой книге рассматриваются версии Perl 5.6, 5.8 и 5.10.

Многие приложения и библиотеки реализации регулярных выражений, которые, как утверждается, поддерживают регулярные выражения языка Perl или совместимы с ними, на самом деле просто используют регулярные выражения в стиле языка Perl. Они поддерживают синтаксис, похожий на синтаксис Perl, но не поддерживают тот же набор особенностей регулярных выражений. Наиболее вероятно, что они используют один из диалектов, что приводятся в списке ниже. Все эти диалекты являются диалектами в стиле языка Perl.

PCRE

PCRE – это библиотека реализации «Perl-Compatible Regular Expressions» (Perl-совместимых регулярных выражений), написанная на языке C Филиппом Хейзелем (Philip Hazel). Вы можете получить исходные тексты библиотеки по адресу: *http://www.pcre.org*. В этой книге рассматриваются версии PCRE с 4 по 7.

Несмотря на утверждение, что библиотека PCRE совместима с диалектом регулярных выражений языка Perl, что в отношении ее даже более справедливо, чем для других диалектов, тем не менее, она всего лишь поддерживает стиль языка Perl. Некоторые особенности, такие как поддержка Юникода, в ней отличаются, и она не допускает внедрение программного кода на языке Perl в регулярные выражения, как это допускается в самом языке Perl.

Благодаря свободной лицензии и надежности библиотека PCRE нашла применение во многих языках программирования и приложениях. Она встроена в язык PHP и включена в состав многих компонентов Delphi. Если заявлено, что приложение поддерживает «Perl-совместимые» регулярные выражения без указания конкретного используемого диалекта, наиболее вероятно, что используется библиотека PCRE.

.NET

Платформа Microsoft .NET Framework обеспечивает поддержку полноценного диалекта регулярных выражений в стиле языка Perl в виде пакета *System.Text.RegularExpressions*. В этом издании описываются версии .NET начиная с 1.0 по 3.5. Строго говоря, существует всего две версии пакета *System.Text.RegularExpressions*: 1.0 и 2.0. В версиях .NET 1.1, 3.0 и 3.5 классы Regex не претерпели никаких изменений.

Любые языки программирования, поддерживаемые платформой .NET, включая C#, VB.NET, Delphi for .NET и даже COBOL.NET, обладают полным доступом ко всем особенностям диалекта регулярных выражений .NET. Если приложение, разработанное на основе платформы .NET, обеспечивает поддержку регулярных выражений, можно быть совершенно уверенным, что оно использует диалект .NET, даже если заявлено, что используются «регулярные выражения Perl». Исключение составляет сама среда разработки Visual Studio (VS). Сама среда разработки VS по-прежнему использует устаревший диалект регулярных выражений, использовавшийся с самого начала, который вообще не является диалектом в стиле языка Perl.

Java

Java 4 – это первая версия, в которой присутствует встроенная поддержка регулярных выражений, реализованная в виде пакета java.util.regex. Эта реализация быстро затмила различные сторонние библиотеки для Java. Кроме того, что этот пакет является стандартным и встроенным, он также предлагает полноценную реализацию Perl-совместимого диалекта регулярных выражений и имеет превосходную производительность, сопоставимую даже с приложениями, написанными на языке C. В этой книге рассматривается пакет java.util.regex, входящий в состав Java 4, 5 и 6.

Если вы пользуетесь программным обеспечением, написанным на языке Java не более чем пару лет тому назад, любые регулярные выражения, поддерживаемые им, весьма вероятно используют диалект Java.

JavaScript

Название *JavaScript* используется в этой книге для обозначения диалекта регулярных выражений, определяемого стандартом ECMA-262 версии 3. Этот стандарт определяет язык программирования ECMAScript, который в различных веб-броузерах более широко известен под названиями JavaScript и JScript. Все броузеры: Internet Explorer с версии 5.5 по 8.0, Firefox, Opera и Safari – реализуют третью версию стандарта ECMA-262. Однако у каждого из броузеров имеются свои отклонения от стандарта. Мы будем указывать на эти отклонения в ситуациях, когда это будет иметь значение.

Если веб-сайт позволяет выполнять поиск или фильтрацию содержимого с помощью регулярных выражений, не вынуждая ожидать ответа веб-сервера, следовательно он использует диалект JavaScript регулярных выражений, который поддерживается броузерами на стороне клиента. Этот диалект используется даже в языках программирования Microsoft VBScript и Adobe ActionScript.

Python

Язык программирования Python обеспечивает поддержку регулярных выражений посредством модуля *re*. В этой книге рассматриваются версии Python 2.4 и 2.5. Поддержка регулярных выражений в языке Python остается неизменной уже в течение нескольких лет.

Ruby

Поддержка регулярных выражений в языке Ruby реализована как часть самого языка, так же как и в Perl. В этой книге рассматриваются версии Ruby 1.8 и 1.9. Версия Ruby 1.8 по умолчанию использует диалект регулярных выражений, реализация которого включена непосредственно в исходные тексты Ruby. Версия Ruby 1.9 по умолчанию использует библиотеку регулярных выражений Onigurama. Имеется возможность скомпилировать Ruby 1.8 с поддержкой библиотеки Onigurama, а Ruby 1.9 – с поддержкой старого диалекта регулярных выражений Ruby. В этой книге мы будем обозначать встроенный диалект Ruby как Ruby 1.8, а диалект Onigurama как Ruby 1.9.

Чтобы определить, какой диалект регулярных выражений Ruby используется у вас, попробуйте выполнить регулярное выражение `‹a++›`. Если поддерживается диалект Ruby 1.8, будет выдано сообщение об ошибке, потому что он не поддерживает захватывающие квантификаторы, тогда как в диалекте Ruby 1.9 этому выражению соответствует строка, состоящая из одного или более символов.

Библиотека Onigurama предусматривает обратную совместимость с диалектом Ruby 1.8, просто добавляя новые возможности, которые не влияют на работу существующих регулярных выражений. Разработчики даже оставили некоторые особенности, которые следовало бы изменить, например выражение `(?m)` означает «точка соответствует концу строки», тогда как в других диалектах для этой же цели используется выражение `(?s)`.

Поиск с заменой с помощью регулярных выражений

Поиск с заменой – это типичная задача, которая решается с применением регулярных выражений. Функция, реализующая поиск с заменой, принимает строку, в которой производится поиск, регулярное выражение и строку замены, которая будет замещать найденные фрагменты. Результатом функции является строка, где все участки, соответствующие регулярному выражению, заменяются замещающим текстом.

Даже при том, что замещающий текст не является регулярным выражением, тем не менее, допускается использовать специальный синтаксис с целью динамического создания замещающего текста. Как это делается, описывается в рецептах 2.20 и 2.21. Некоторые диалекты также поддерживают вставку совпавшего содержимого в замещающий текст, как показано в рецепте 2.22. В главе 3 в рецепте 3.16 показано, как с помощью программного кода генерировать различные замещающие тексты для каждого совпадения.

Множество диалектов определения замещающего текста

Многообразие идей, выдвигаемых различными разработчиками реализаций регулярных выражений, привело к появлению широкого диапазона диалектов регулярных выражений, каждый из которых обладает своими особенностями. История выработки синтаксиса определения замещающего текста также не является исключением. В действительности диалектов определения замещающего текста оказалось даже больше, чем диалектов самих регулярных выражений. Создание механизма регулярных выражений – задача не из легких. Большинство программистов предпочитают использовать один из существующих, а прикрутить функцию поиска с заменой к существующему механизму регулярных выражений значительно проще. В результате появилось множество диалектов определения замещающего текста для библиотек поддержки регулярных выражений, в которых отсутствуют собственные реализации функции поиска с заменой.

К счастью, все диалекты регулярных выражений, представленные в этой книге, обладают собственными возможностями выполнения по-

иска с заменой, за исключением PCRE. Этот недостаток PCRE усложняет жизнь тем программистам, которые используют диалекты, созданные на его основе. Свободно распространяемая библиотека PCRE не содержит встроенных функций, реализующих возможность замены. Поэтому все приложения и языки программирования, опирающиеся на использование PCRE, должны предоставлять собственные реализации функции поиска с заменой. Многие программисты пытаются копировать существующие реализации, но практически всегда делают это немного по-разному.

Ниже перечислены диалекты определения замещающего текста, рассматриваемые в этой книге. За дополнительной информацией о соответствующих им диалектах регулярных выражений обращайтесь к разделу «Множество диалектов регулярных выражений»:

Perl

Язык Perl обладает встроенной поддержкой замещения текста с помощью регулярных выражений, реализованной в виде инструкции `s/regex/replace/`. Диалект определения замещающего текста в языке Perl соответствует его диалекту регулярных выражений. В этой книге рассматриваются версии от Perl 5.6 по Perl 5.10. В последней версии была добавлена поддержка именованных обратных ссылок в замещающем тексте как дополнение к добавленной в регулярные выражения поддержке именованных сохранений.

PHP

В этой книге рассматривается реализация поиска с заменой на языке PHP в виде функции `preg_replace`. Данная функция использует диалект регулярных выражений PCRE и диалект PHP определения замещающего текста.

В других языках программирования, опирающихся на библиотеку PCRE, используются отличные от PHP диалекты определения замещающего текста. В зависимости от того, где черпали вдохновение разработчики того или иного языка программирования, синтаксис определения замещающего текста может быть похожим на диалект PHP или на любой другой диалект определения замещающего текста из представленных в этой книге.

В языке PHP имеется также функция `ereg_replace`. Эта функция использует другой диалект регулярных выражений (POSIX ERE) и иной диалект определения замещающего текста. Семейство функций `ereg` из языка PHP не будет рассматриваться в этой книге.

.NET

Пакет *System.Text.RegularExpressions* содержит несколько функций, реализующих поиск с заменой. Диалект определения замещающего текста в .NET соответствует диалекту регулярных выражений .NET. Все версии .NET используют один и тот же диалект определения замещающего текста. Новые особенности регулярных вы-

ражений, появившиеся в .NET 2.0, никак не повлияли на синтаксис определения замещающего текста.

Java

Пакет *java.util.regex* содержит собственные встроенные функции поиска с заменой. В этой книге рассматриваются версии Java 4, 5 и 6. Во всех этих версиях используется один и тот же синтаксис определения замещающего текста.

JavaScript

Под названием *JavaScript* в этой книге мы будем подразумевать как диалект определения замещающего текста, так и диалект регулярных выражений, определяемый стандартом ECMA-262 версии 3.

Python

Модуль *re* в языке Python реализует поиск с заменой в виде функции `sub()`. Диалект определения замещающего текста в языке Python соответствует диалекту регулярных выражений Python. В этой книге рассматриваются версии Python 2.4 и 2.5. Реализация поддержки регулярных выражений в языке Python остается неизменной на протяжении уже многих лет.

Ruby

Поддержка регулярных выражений в языке Ruby встроена непосредственно в сам язык, включая и функцию поиска с заменой. В этой книге рассматриваются версии Ruby 1.8 и 1.9. Версия Ruby 1.8 по умолчанию использует диалект регулярных выражений, реализация которого включена непосредственно в исходные тексты Ruby, тогда как версия Ruby 1.9 по умолчанию использует библиотеку регулярных выражений Onigurama. Имеется возможность скомпилировать Ruby 1.8 с поддержкой библиотеки Onigurama, а Ruby 1.9 – с поддержкой старого диалекта регулярных выражений Ruby. В этой книге мы будем обозначать встроенный диалект Ruby как Ruby 1.8, а диалект Onigurama как Ruby 1.9.

Для Ruby 1.8 и 1.9 используется один и тот же синтаксис определения замещающего текста, за исключением того, что в версии Ruby 1.9 добавлена поддержка именованных обратных ссылок в замещающем тексте. Именованные сохранения – это новая особенность, появившаяся в поддержке регулярных выражений Ruby 1.9.

Инструменты для работы с регулярными выражениями

Если у вас нет опыта программирования регулярных выражений, мы рекомендуем для начала поэкспериментировать с ними с помощью дополнительных инструментов и только потом включать их в программ-

ный код. В этой главе представлены примеры простых регулярных выражений, которые не содержат дополнительных экранированных последовательностей, те что необходимы при использовании в языках программирования (даже в языке командной оболочки UNIX). Эти регулярные выражения можно вводить непосредственно в поле поиска приложения.

В главе 3 описывается, как оформляются регулярные выражения в исходных текстах программ. Оформление регулярных выражений в виде строк еще больше осложняет их восприятие человеком, потому что к правилам экранирования последовательностей внутри регулярных выражений добавляются правила экранирования служебных символов в строках. Мы не будем касаться этой проблемы до рецепта 3.1. Как только вы разберетесь с основами регулярных выражений, вы сможете увидеть лес за частоколом обратных слэшей.

Инструменты, описываемые в этом разделе, кроме всего прочего обеспечивают возможность отладки, проверки синтаксиса и другие удобства, которые недоступны в большинстве сред программирования. По этой причине в процессе разработки регулярных выражений для своих приложений вы можете найти для себя удобным сначала создать сложное регулярное выражение в одном из этих инструментов и потом перенести его в свою программу.

RegexBuddy

Программа RegexBuddy (рис. 1.1) – это один из наиболее полнофункциональных инструментов, доступных на момент написания этих строк, позволяющий создавать, тестировать и реализовывать регулярные выражения. Он обладает уникальной возможностью имитировать все диалекты регулярных выражений, рассматриваемые в этой книге, и даже преобразовывать их из одного диалекта в другой.

Программа RegexBuddy была спроектирована и разработана Яном Гойвертсом (Jan Goyvaerts), одним из авторов этой книги. Работа над программой RegexBuddy сделала Яна экспертом в области регулярных выражений, а использование RegexBuddy помогло довести его соавтора Стивена до этапа передачи этой книги в издательство O'Reilly.

Если интерфейс программы (рис. 1.1) покажется вам перегруженным, то это только потому, что мы распахнули большую часть из имеющихся панелей, чтобы продемонстрировать обширные возможности RegexBuddy. По умолчанию все панели аккуратно свернуты в строку вкладок. Существует также возможность перемещать панели на второй монитор.

Чтобы опробовать какое-либо регулярное выражение из представленных в этой книге, просто введите его в поле редактирования, расположенное в верхней части окна RegexBuddy. Программа автоматически

подсветит синтаксис введенного регулярного выражения, ярко выделив возможные ошибки и непарные скобки.

По мере ввода регулярного выражения в панели Create (Создать) автоматически будет создаваться подробный отчет на английском языке. Двойной щелчок мышью на любом из описаний в дереве будет вызывать переход к редактированию соответствующей части регулярного выражения. Вы можете вставлять новые элементы в регулярное выражение вручную или, щелкнув на кнопке Insert Token (Вставить лексему), выбирать нужный элемент из меню. Например, если вы не помните достаточно сложный синтаксис позитивных опережающих проверок, можно попросить RegexBuddy вставить соответствующие символы.

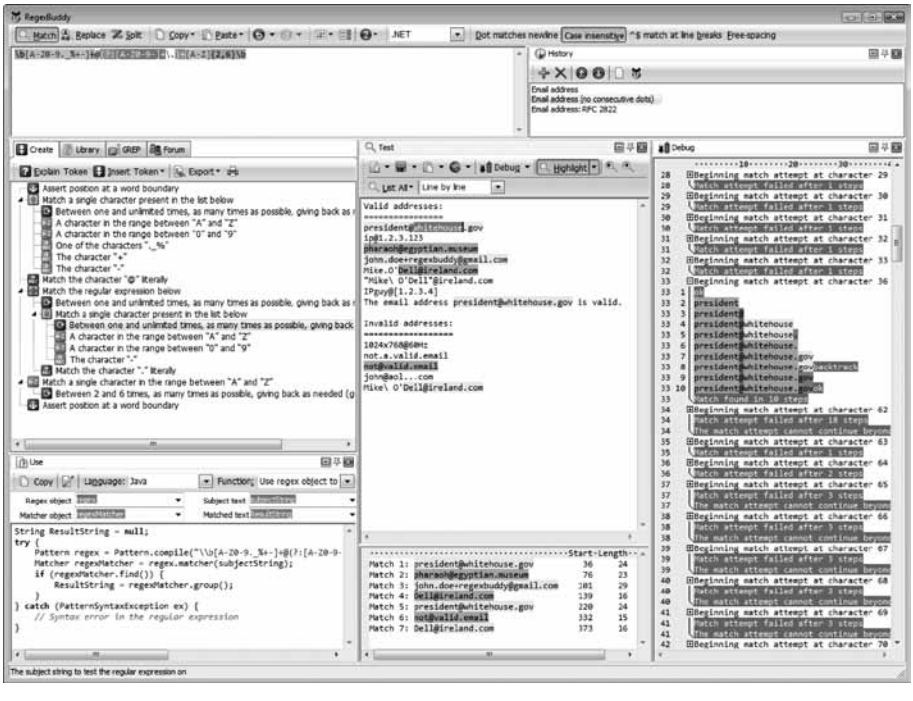


Рис. 1.1. *RegexBuddy*

После ввода или вставки некоторого примерного текста в панель Test (Проверка) при нажатой кнопке Highlight (Выделить) программа RegexBuddy автоматически выделит фрагменты текста, соответствующие регулярному выражению.

В число кнопок, которые вам наверняка придется использовать чаще всего, входят:

List All (Перечислить все)

Отображает список всех совпадений.

Replace (Заменить)

Кнопка Replace (Заменить), расположенная в верхней части окна, дает возможность ввести замещающий текст. Кнопка Replace (Заменить) в панели Test (Проверка) дает возможность увидеть, как будет выглядеть первоначальный текст после выполнения операции замены.

Split (Разделить) (кнопка в панели Test, а не в верхней части окна)

Интерпретирует регулярное выражение как разделитель и разбивает исходный текст на лексемы по совпадениям, найденным с помощью регулярного выражения.

Щелкните на любой из этих кнопок и в меню List All (Перечислить все) выберите пункт Update Automatically (Обновлять автоматически), чтобы обеспечить автоматическую синхронизацию результатов по мере редактирования регулярного выражения или текста примера.

Чтобы в точности увидеть, как работает (или не работает) регулярное выражение, следует щелкнуть в панели Test (Проверка) на выделенном совпадении или в месте, где обнаружено отсутствие совпадения, и затем щелкнуть на кнопке Debug (Отладка). Программа RegexBuddy переключится в панель Debug (Отладка) и по шагам продемонстрирует процесс поиска совпадения. Щелкнув мышью в пределах вывода отладчика, можно определить, какому фрагменту регулярного выражения соответствует текст, на котором был выполнен щелчок. Щелкнув мышью в пределах регулярного выражения, можно выделить фрагмент текста в панели отладчика, соответствующий данной части регулярного выражения.

В панели Use (Использование) можно выбрать предпочтительный язык программирования и функцию и тут же получить исходный программный код реализации регулярного выражения. Исходные тексты шаблонов для RegexBuddy полностью доступны для редактирования во встроенном редакторе шаблонов. Благодаря этому можно добавлять новые функции и даже новые языки программирования или изменять существующие.

Чтобы испытать работу регулярного выражения на большом объеме данных, можно переключиться в панель GREP и произвести поиск (возможно, с заменой) в любом числе файлов и каталогов.

Обнаружив регулярное выражение в исходном тексте сопровождаемой программы, можно скопировать его в буфер обмена, включая ограничивающие его кавычки или слэши, затем в окне программы RegexBuddy щелкнуть на кнопке Paste (Вставить), расположенной в верхней части, и выбрать стиль оформления строк для вашего языка программирования. После этого регулярное выражение появится в окне RegexBuddy как обычное регулярное выражение, без дополнительных кавычек и экранирующих последовательностей, необходимых для оформления

литералов строк. Чтобы создать в буфере обмена строку с соблюдением синтаксических правил, достаточно щелкнуть на кнопке Copy (Копировать), расположенной в верхней части, которую затем можно будет вставить в исходный текст программы.

По мере накопления опыта в панели Library (Библиотека) можно создавать собственную библиотеку регулярных выражений. Не забывайте добавлять подробное описание и тестовые примеры при сохранении регулярного выражения. Регулярные выражения могут выглядеть сложными для понимания даже опытных экспертов.

В случае неудачи при создании собственного регулярного выражения можно щелкнуть на вкладке панели Forum (Форум) и затем на кнопке Login (Зарегистрироваться). В случае если вы приобрели программу RegexBuddy на законных основаниях, на экране появится форма входа, заполнив которую и щелкнув на кнопке OK вы подключитесь к форуму пользователей RegexBuddy. Стивен и Ян часто бывают здесь.

Программа RegexBuddy работает под управлением операционных систем Windows 98, ME, 2000, XP и Vista. Пользователи Linux и Apple могут запускать ее под управлением VMware, Parallels, CrossOver Office и, с некоторыми усилиями, под WINE. Загрузить оценочную версию RegexBuddy можно по адресу *http://www.regexbuddy.com/RegexBuddyCookbook.exe*. Оценочная версия обладает полным набором функциональных возможностей в течение семи дней фактического использования, за исключением доступа к форуму.

RegexPal

RegexPal (рис. 1.2) – это веб-приложение, выполняющее тестирование регулярных выражений. Оно было написано Стивеном Левитаном (Steven Levithan), одним из авторов этой книги. Все, что необходимо для работы с этим приложением, это современный веб-броузер. Приложение RegexPal полностью написано на языке JavaScript. Вследствие этого оно поддерживает только диалект регулярных выражений JavaScript, реализованный в веб-броузере, используемом для доступа к веб-приложению.

Чтобы опробовать какое-либо регулярное выражение из этой книги, откройте страницу *http://www.regexpal.com* и введите регулярное выражение в поле с текстом «Enter regex here». Приложение RegexPal автоматически подсветит синтаксис введенного регулярного выражения, что сразу же позволит определить синтаксические ошибки. Приложение RegexPal учитывает проблемы несовместимости броузеров, которые могут нарушить ваши планы при разработке регулярных выражений для JavaScript. Если какая-то синтаксическая конструкция работает некорректно в некоторых броузерах, RegexPal выделит ее как ошибку.

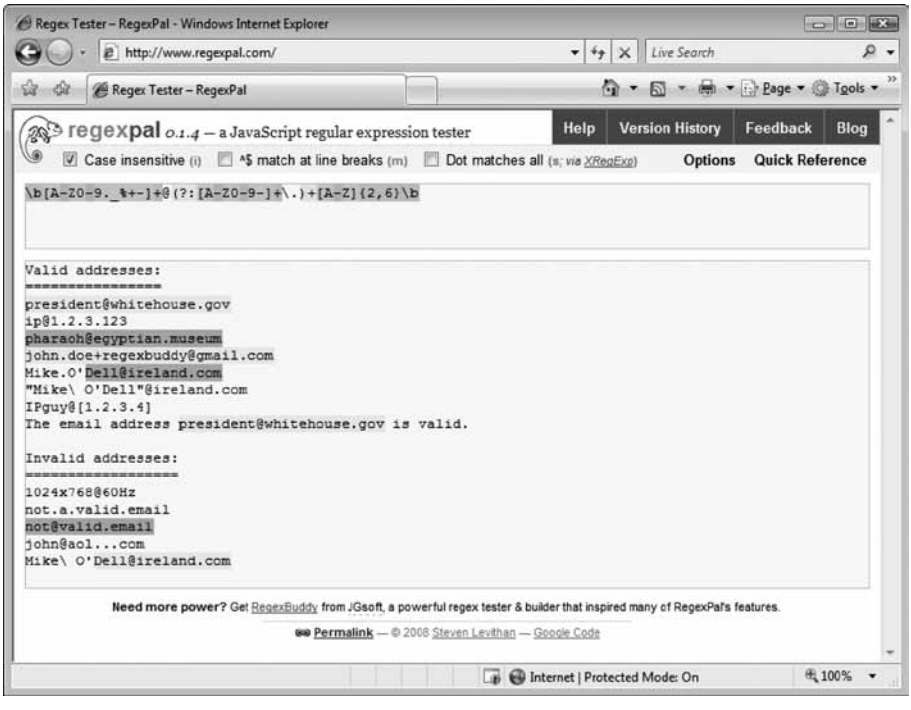


Рис. 1.2. *RegexPal*

Теперь можно ввести текст примера в поле с текстом «Enter test data here», после чего RegexPal автоматически выделит фрагменты текста, соответствующие регулярному выражению.

Здесь нет никаких кнопок, на которых требовалось бы было бы щелкать, что делает RegexPal одним из самых удобных веб-приложений тестирования регулярных выражений.

Другие примеры веб–приложений тестирования регулярных выражений

Создать простое веб-приложение для тестирования регулярных выражений достаточно просто. Если у вас имеются основные навыки веб-разработки, в главе 3 вы найдете все, что вам будет необходимо для создания такого приложения. Сотни людей уже сделали это, причем некоторые из них дополнили свои приложения дополнительными возможностями, которые делают их заслуживающими внимания.

regex.larsolavtorvik.com

Ларс Олав Торвик (Lars Olav Torvik) создал небольшое, но замечательное приложение тестирования регулярных выражений, доступное по адресу *http://regex.larsolavtorvik.com* (рис. 1.3).



Рис. 1.3. *regex.larsolavtorvik.com*

Для начала выберите требуемый диалект регулярных выражений, щелкнув на названии диалекта в верхней части страницы. Ларс предлагает выбор между диалектами PHP PCRE, PHP POSIX и JavaScript. PHP PCRE – это диалект PCRE, рассматриваемый в этой книге и используемый в языке PHP семейством функций `preg`. POSIX – это устаревший диалект с ограниченными возможностями, используемый семейством функций `ereg` в языке PHP, который не рассматривается в данной книге. В случае выбора JavaScript вы будете работать с реализацией JavaScript в вашем броузере.

Введите регулярное выражение в поле с меткой `Pattern` (Шаблон) и текст примера в поле `Subject` (Испытуемый текст). Спустя мгновение в поле `Matches` (Совпадения) появится текст примера с выделенными фрагментами, соответствующими регулярному выражению. В поле `Code` (Программный код) появится строка программного кода, которая применяет регулярное выражение к испытуемому тексту. Операция копирования и вставки этой строки в текстовом редакторе позволит вам

сэкономить время на преобразование вручную регулярного выражения в строковый литерал. Любая строка или массив, возвращаемые программным кодом, отображаются в поле Result (Результат). Так как при создании своего веб-приложения Ларс использовал технологию Ajax, результаты появляются всего через несколько мгновений для любого выбранного диалекта. Для работы с этим инструментом необходимо подключение к сети, так как интерпретатор PHP выполняется на сервере, а не в броузере.

Во второй колонке отображается список команд и параметров регулярных выражений. Перечень доступных команд и параметров зависит от выбранного диалекта регулярных выражений. В число команд обычно входят такие операции, как поиск совпадений, замена и разбиение. В число параметров входят такие, как чувствительность к регистру символов, а также ряд других параметров, характерных для реализации. Эти команды и параметры описываются в главе 3.

Nregex

http://www.nregex.com (рис. 1.4) – это простое веб-приложение, предназначенное для тестирования регулярных выражений, созданное Дэвидом Сераянгом (David Seruyange) на основе технологии .NET. Несмотря на то, что на сайте явно не говорится, какой диалект реализован, – это .NET 1.x, по крайней мере, на момент написания этих строк.

Расположение элементов на странице может немного сбивать с толку. Регулярное выражение следует вводить в поле с надписью Regular Expression (Регулярное выражение), а параметры выражения определяются с помощью флажков под ним. Испытуемый текст вводится в поле, расположенное ниже, где первоначально находится текст «`If I just had $5.00 then "she" wouldn't be so @#$! mad.`». Если предметом испытаний является веб-страница, можно ввести адрес URL в поле Load Target From URL (Загрузить испытуемый текст с адреса URL) и щелкнуть на кнопке Load (Загрузить), расположенной под полем ввода. Если предметом испытаний является файл, расположенный на жестком диске, можно щелкнуть на кнопке Browse (Просмотреть), отыскать требуемый файл и щелкнуть на кнопке Load (Загрузить), расположенной под полем ввода.

Испытуемый текст появится в поле Matches & Replacements (Совпадения и замены), расположенном в центре страницы, в котором будут выделены фрагменты, соответствующие регулярному выражению. Если ввести какой-либо текст в поле Replacement String (Строка замены), то будет отображен результат выполнения операции поиска с заменой. В случае наличия ошибки в регулярном выражении будет выведено многоточие (...).

Поиск соответствий регулярному выражению производится программным кодом платформы .NET, выполняющимся на стороне сервера, поэтому для работы с сайтом необходимо подключение к сети. Если автоматическое обновление страницы выполняется слишком медленно,



Рис. 1.4. *Nregex*

наиболее веротно, что это происходит из-за большого объема испытуемого текста. В этом случае отметьте флажок Manually Evaluate Regex (Производить вычисления вручную), расположенный выше поля ввода регулярного выражения. В результате на странице появится кнопка Evaluate (Вычислить), щелчок на которой будет приводить к обновлению содержимого поля Matches & Replacements (Совпадения и замены).

Rubular

По адресу *http://www.rubular.com* можно найти небольшое веб-приложение, выполняющее проверку регулярных выражений (рис. 1.5), которое использует диалект Ruby 1.8.

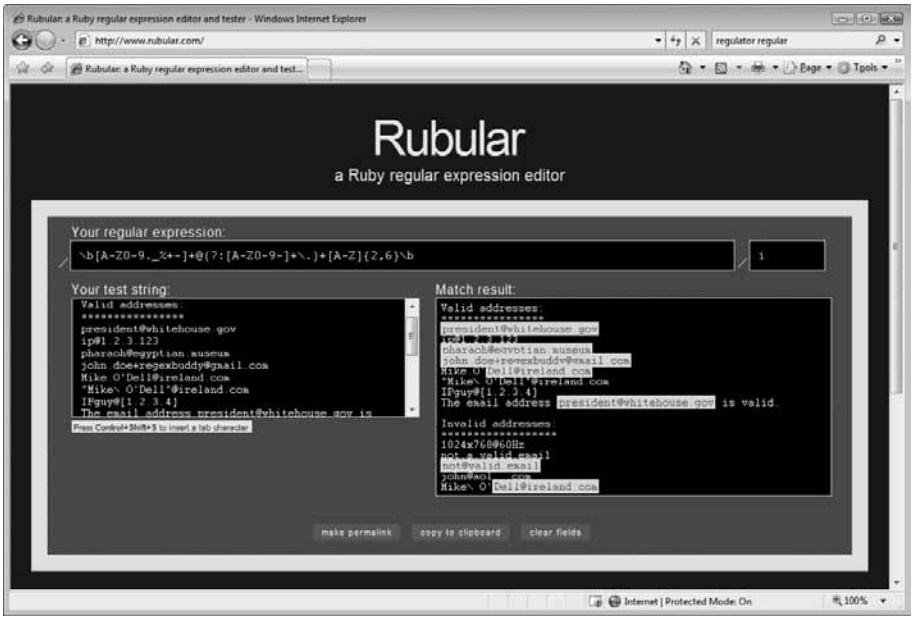


Рис. 1.5. *Rubular*

Регулярное выражение вводится в поле Your regular expression (Ваше регулярное выражение), расположенное между двумя символами слэша. Отключить чувствительность к регистру символов можно, введя символ `i` в небольшое поле ввода, находящееся правее второго символа слэша. При желании точно так же можно определить параметр «точка соответствует концу строки», введя символ m в то же самое поле. Комбинация символов im активизирует оба параметра. Эти соглашения могут показаться недружественными для тех, кто не знаком с языком Ruby, однако они соответствуют синтаксису `/regex/im`, используемому в исходных текстах программ на языке Ruby.

Введите или скопируйте испытуемый текст в поле Your test string (Ваша тестовая строка) и подождите мгновение. В поле Match result (Результат поиска соответствия) появится испытуемый текст с выделенными фрагментами, соответствующими регулярному выражению.

myregexp.com

Сергеем Евдокимовым (Sergey Evdokimov) было создано несколько приложений для разработчиков на языке Java, позволяющих тестировать регулярные выражения. На домашней странице *http://www.myregexp.com* (рис. 1.6) предоставляется доступ к веб-приложению тестирования регулярных выражений. Это Java-апплет, который выполняется броузером. При этом необходимо, чтобы на компьютере клиента была установлена среда выполнения Java 4 (или более поздней версии). Для оцен-

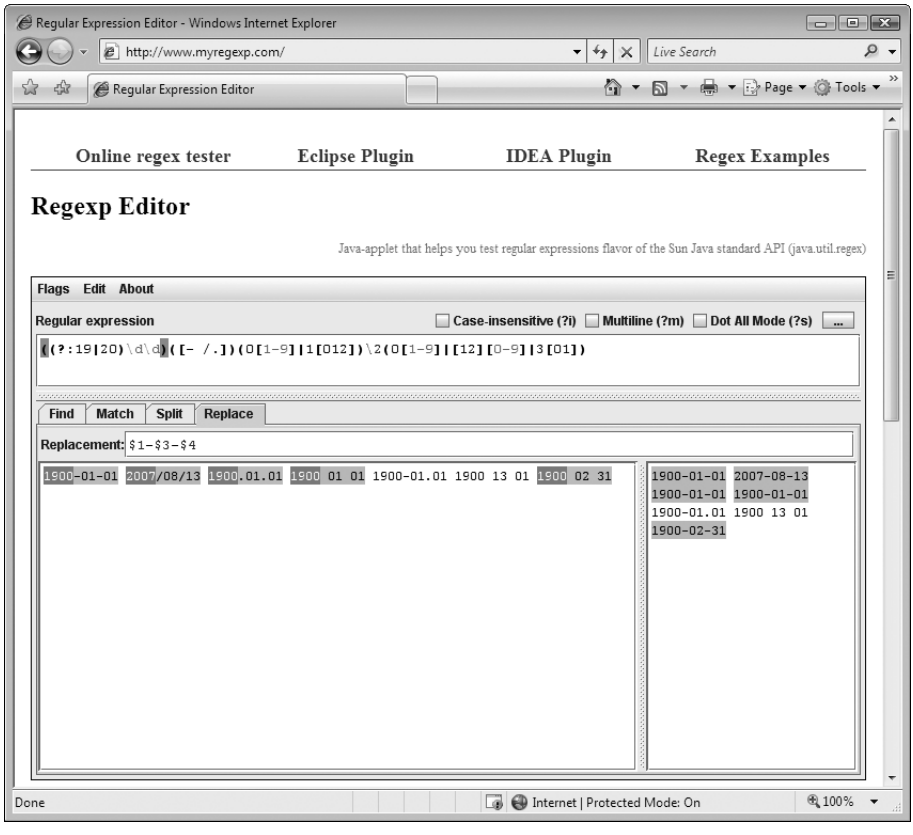


Рис. 1.6. *myregexp.com*

ки регулярных выражений используется пакет `java.util.regex`, который впервые появился в версии Java 4. Под диалектом регулярных выражений «Java» подразумевается именно этот пакет.

Регулярное выражение вводится в поле Regular Expression (Регулярное выражение). Желаемые параметры регулярного выражения устанавливаются с помощью меню Flags (Флаги). Кроме того, три параметра имеют собственные отдельные флажки.

Если необходимо проверить регулярное выражение, уже имеющееся в исходном программном коде на языке Java, достаточно просто скопировать строку целиком в буфер обмена, затем на странице *myregexp.com* щелкнуть на пункте меню Edit (Правка) и выбрать пункт Paste Regex from Java String (Вставить регулярное выражение из строки на языке Java). По завершении редактирования регулярного выражения в том же самом меню можно выбрать пункт Copy Regex for Java Source (Копировать регулярное выражение в формате строки на языке Java). В меню Edit (Правка) имеются похожие команды для JavaScript и XML.

Под полем ввода регулярного выражения имеются четыре вкладки, позволяющие производить разные виды испытаний:

Find (Поиск)

Выделяет все соответствия регулярному выражению, обнаруженные в испытуемом тексте. Эти соответствия отыскиваются с помощью метода `Matcher.find()` языка Java.

Match (Соответствие)

Проверяет, соответствует ли весь испытуемый текст целиком указанному регулярному выражению. Если соответствует, выделяется весь текст целиком. Этот тест демонстрирует работу методов `String.matches()` и `Matcher.matches()`.

Split (Разбиение)

Второе поле справа отображает массив строк, возвращаемых методом `String.split()` или `Pattern.split()` при использовании указанных вами регулярного выражения и испытуемого текста.

Replace (Замена)

После ввода замещающего текста в поле справа будет выведен текст, возвращаемый методом `String.replaceAll()` или `Matcher.replaceAll()`.

Другие приложения тестирования регулярных выражений, созданные Сергеем, можно отыскать с помощью ссылок, расположенных в верхней части страницы *http://www.myregexp.com*. Одно из них является модулем расширения для среды программирования Eclipse, а другое – модулем расширения для среды программирования IntelliJ IDEA.

reAnimator

Еще одно приложение – reAnimator *http://osteele.com/tools/reanimator* (рис. 1.7), созданное Оливером Стилом (Oliver Steele), вовсе не предназначено для оживления умерших регулярных выражений. Это лишь небольшой забавный инструмент, дающий графическое изображение конечных автоматов, которые используются механизмом регулярных выражений для поиска соответствия регулярному выражению.

Синтаксис регулярных выражений, поддерживаемых приложением reAnimator, весьма ограничен. Он совместим со всеми диалектами, рассматриваемыми в этой книге. Любое регулярное выражение, которое можно воспроизвести с помощью reAnimator, будет работать с любым из диалектов, представленных в книге, но это не означает, что обратное утверждение тоже верно. Это обусловлено тем, что регулярные выражения, поддерживаемые приложением reAnimator, являются регулярными в математическом смысле. Краткие пояснения по этому вопросу приводятся во врезке «История появления термина „регулярное выражение”» на стр. 20.

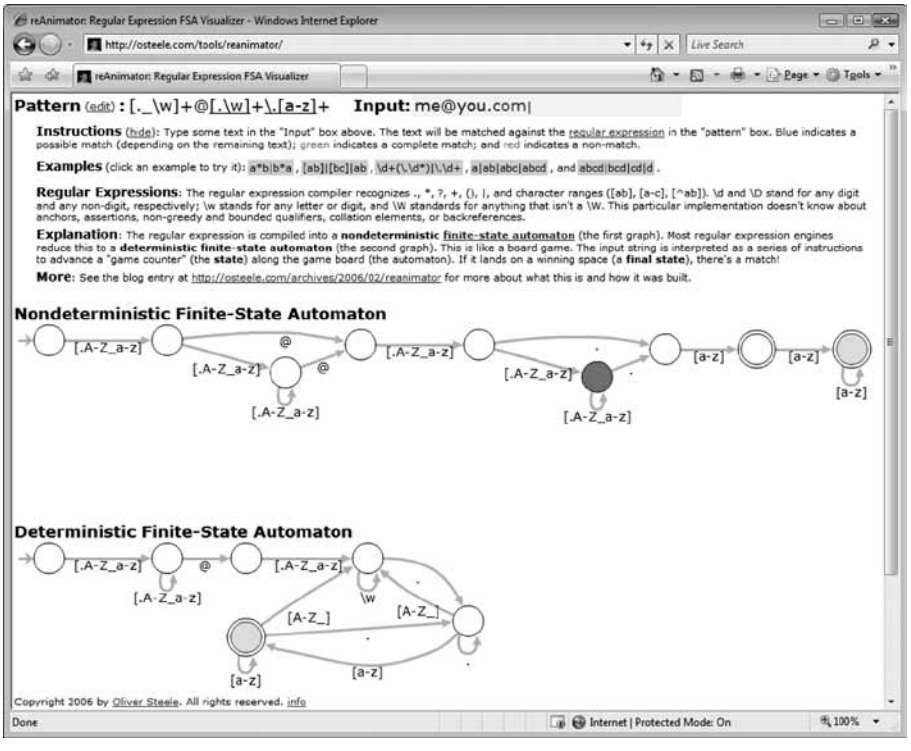


Рис. 1.7. *reAnimator*

Сначала необходимо перейти к полю Pattern (Шаблон) и щелкнуть на кнопке Edit (Правка). Ввести регулярное выражение в поле Pattern (Шаблон) и щелкнуть на кнопке Set (Установить). Затем следует медленно вводить испытуемый текст в поле Input (Ввод).

По мере ввода каждого символа цветные шарики будут перемещаться по изображению конечного автомата, отмечая конечную позицию, достигнутую автоматом к настоящему моменту. Голубые шарики показывают, что конечный автомат принял ввод, но при этом для достижения полного совпадения требуется дополнительный ввод. Зеленые шарики показывают, что ввод полностью соответствует шаблону. Отсутствие шариков говорит о том, что конечный автомат не может отыскать соответствие.

Приложение reAnimator обнаруживает соответствия, только если регулярному выражению соответствует вся строка целиком, как если бы она находилась между якорными элементами ‹^› и ‹$›. Это еще одна особенность выражений, которые являются регулярными в математическом смысле.

Примеры настольных приложений тестирования регулярных выражений

Expresso

Expresso (не следует путать с кофе espresso (эспрессо), богатым кофеином) – это приложение для платформы .NET, позволяющее создавать и тестировать регулярные выражения. Загрузить его можно по адресу *http://www.ultrapico.com/Expresso.htm*. Для его работы на компьютере должна быть установлена платформа .NET 2.0 или более поздней версии.

Для свободной загрузки доступна 60-дневная оценочная версия. По окончании оценочного периода необходимо зарегистрироваться, в противном случае программа Expresso перестанет работать. Регистрация выполняется бесплатно, но требует указать парням из Ultrapico свой адрес электронной почты. Регистрационный ключ высылается по электронной почте.

Внешний вид приложения Expresso приводится на рис. 1.8. Поле Regular Expression (Регулярное выражение), предназначенное для ввода регулярных выражений, постоянно находится в зоне видимости. Подсветка синтаксиса отсутствует. В поле Regex Analyzer (Анализатор регулярного выражения) автоматически создается краткий отчет о регулярном выражении на английском языке. Это поле также отображается постоянно.

В нижней части вкладки Design Mode (Режим проектирования) имеется возможность устанавливать такие параметры поиска совпадений, как Ignore Case (Игнорировать регистр символов). Большую часть экранного пространства занимает линия вкладок, где можно выбирать элементы для добавления в регулярное выражение. Те, кто имеет два монитора или один большой монитор, могут щелчком мыши на кнопке Undock (Отделить) отделить линию вкладок, преобразовав ее в плавающую панель. Кроме того, создавать регулярные выражения можно также во вкладке Test Mode (Режим тестирования).

Во вкладке Test Mode (Режим тестирования) следует ввести или вставить испытуемый текст в левый нижний угол и щелкнуть на кнопке Run Match (Выполнить поиск соответствий), чтобы получить в поле Search Results (Результаты поиска) полный список всех соответствий. В испытуемом тексте ничего не выделяется. Чтобы выделить совпадение в испытуемом тексте, следует щелкнуть на совпадении в наборе результатов.

Во вкладке Expression Library (Библиотека выражений) выводится список примеров регулярных выражений и список последних регулярных выражений. Испытуемое регулярное выражение добавляется в этот список всякий раз, когда выполняется щелчок на кнопке Run Match (Выпол-

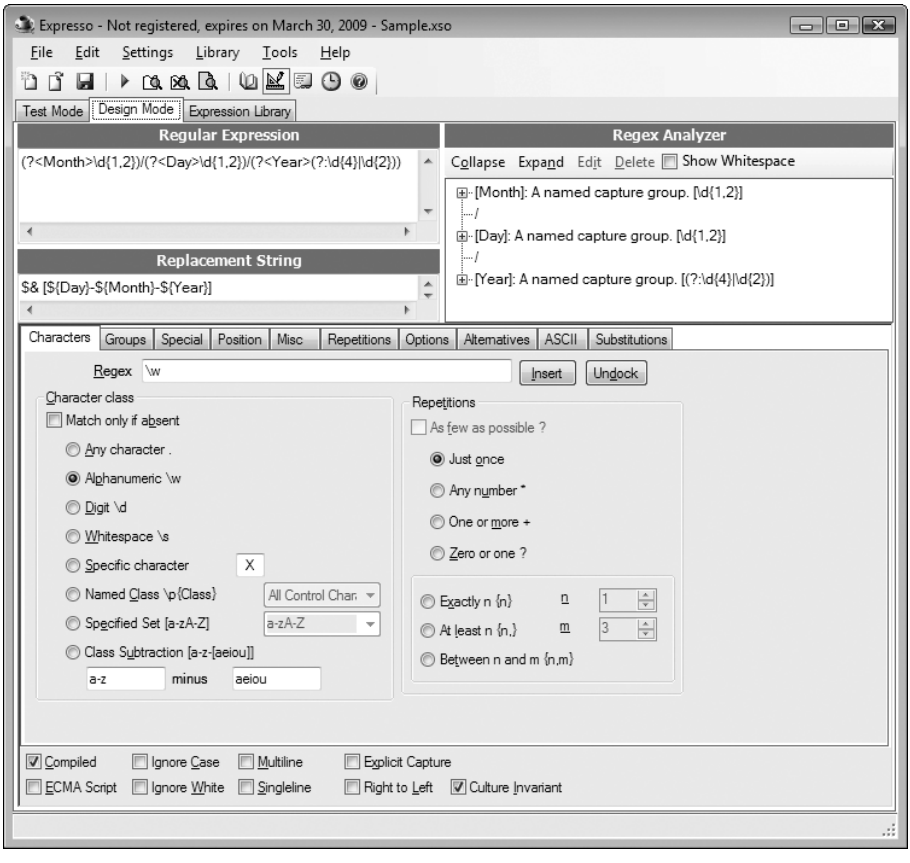


Рис. 1.8. *Expresso*

нить поиск соответствий). Библиотека допускает возможность редактирования посредством меню Library (Библиотека) в главном меню программы.

The Regulator

Приложение Regulator, которое можно загрузить по адресу *http://sourceforge.net/projects/regulator*, не может использоваться в дыхательных аппаратах для подводного плавания или в газовых баллонах – это еще одно приложение на платформе .NET, предназначенное для создания и тестирования регулярных выражений. Последняя версия приложения требует наличия установленной версии .NET 2.0 или более поздней. Кроме того, доступны для загрузки более старые версии приложения для .NET 1.x. Regulator – это открытое программное обеспечение, для приобретения которого не требуется платить деньги или проходить процедуру регистрации.

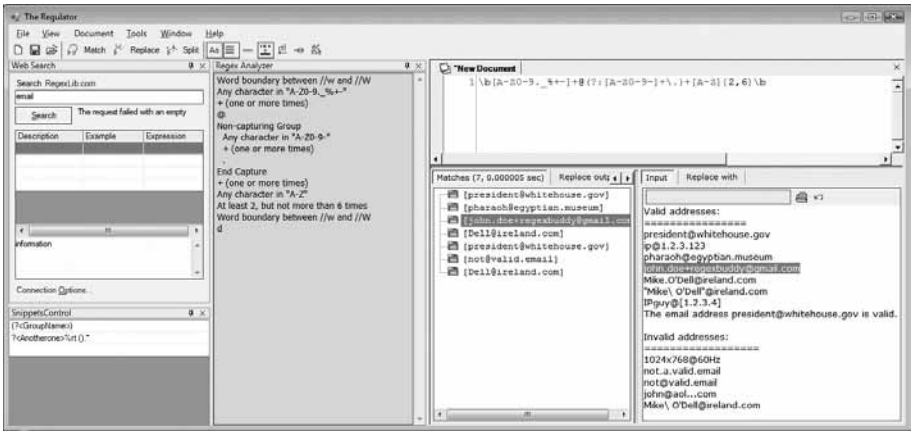


Рис. 1.9. *The Regulator*

Вся функциональность приложения Regulator доступна на единственном экране (рис. 1.9). Вкладка New Document (Новый документ) – это место, где вводится регулярное выражение. Подсветка синтаксиса выполняется автоматически, но синтаксические ошибки, присутствующие в регулярном выражении, не выделяются. Щелчком правой кнопки мыши вызывается контекстное меню, в котором можно выбирать добавляемые элементы регулярного выражения. Параметры регулярного выражения можно устанавливать с помощью кнопок на главной панели инструментов. Ярлыки на кнопках могут ввести в заблуждение, поэтому дожидайтесь появления всплывающих подсказок, чтобы определить назначение каждой кнопки.

Ниже и правее области ввода регулярного выражения имеется кнопка Input (Ввод), щелчок на которой приводит к появлению области, куда можно вставить испытуемый текст. Щелчок на кнопке Replace with (Заменить на) открывает область ввода замещающего текста, используемого в операции поиска с заменой. Ниже и левее области ввода регулярного выражения выводятся результаты работы регулярного выражения. Результат не обновляется автоматически – чтобы обновить результат, необходимо щелкнуть на кнопке Match (Соответствие), Replace (Заменить) или Split (Разбить) в панели инструментов. Фрагменты в испытуемом тексте не выделяются. Щелчок на совпадении в списке результатов приводит к выделению соответствующего фрагмента в испытуемом тексте.

В панели Regex Analyzer (Анализатор регулярного выражения) выводится простой отчет о регулярном выражении на английском языке, но этот отчет не обновляется автоматически и не является интерактивным. Чтобы обновить отчет, следует выбрать пункт Regex Analyzer (Анализатор регулярного выражения) в меню View (Вид), даже если панель анализатора видима. Щелчок мышью в панели анализатора приводит лишь к перемещению текстового курсора.