Урок 6

Соответствие позиций

Теперь вы знаете, как установить соответствие между всеми типами символов во всевозможных типах комбинаций и повторений в любом положении в тексте. Однако иногда необходимо установить соответствие в определенных местоположениях в пределах блока текста, для чего требуется установить соответствие позиций, — именно это объясняется в данном уроке.

Использование границ

Соответствие позиций позволяет указать, где в строке текста должно произойти совпадение. Чтобы понять потребность в соответствии позиций, рассмотрим следующий пример:

Текст

```
The cat scattered his food all over the room.
```

(Кот разбросал свою еду по всей комнате.)

Регулярное выражение

```
cat
```

Результат

```
The cat scattered his food all over the room.
```

(Кот разбросал свою еду по всей комнате.)

Анализ

Шаблон `cat` (кот) соответствует всем вхождениям `cat`, даже вхождению `cat` в слово `scattered`. На самом деле иногда как раз это и нужно, но более чем вероятно, что требуется совсем другое. Если вы хотите в результате поис-

ка заменить все вхождения слова cat на слово dog, то при таком поиске все закончилось бы следующей ерундой:

```
The dog sdogtered his food all over the room.
```

(Пес s**dog**tered свою еду по всей комнате.)

Это заставляет нас учитывать границы и использовать специальные метасимволы для определения позиции (или границы) перед шаблоном и после него.

Границы слова

Символ \b указывает границу слова. Таким образом, \b соответствует началу или концу слова.

Чтобы продемонстрировать использование \b, рассмотрим предыдущий пример снова, на сей раз с указанием границ:

Текст

```
The cat scattered his food all over the room.
```

(Кот разбросал свою еду по всей комнате.)

Регулярное выражение

```
\bcat\b
```

Результат

```
The cat scattered his food all over the room.
```

(Кот разбросал свою еду по всей комнате.)

Анализ

Перед словом cat и после него есть пробел, и слово вместе с пробелами соответствует шаблону cat

> Так чему же именно соответствует \b? Механизмы регулярных выражений не понимают английский или любой другой язык, и потому они не могут определить границы слова. Символ \b просто соответствует позиции между символом, который является обычно частью слова (слова состоят из алфавитно-цифровых символов и символа подчеркивания, т.е. представляют собой текст, который состоит из символов, соответствующих \w) и чем-нибудь, что не может быть частью слова (текст, который состоит из символов, соответствующих \W).

Важно помнить, что для того, чтобы найти слово целиком, \b должен использоваться и перед, и после текста, с которым будет установлено соответствие. Рассмотрим следующий пример:

Текст

```
The captain wore his cap and cape proudly as he sat
listening to the recap of how his crew saved the men from
a capsized vessel.
```

(Капитан гордо сидел в кепке и плаще, слушая краткое повторение рассказа о том, как его команда спасла экипаж опрокинувшегося судна.)

Регулярное выражение

```
\bcap
```

Результат

```
The captain wore his cap and cape proudly as he sat
listening to the recap of how his crew saved the men from
a capsized vessel.
```

(Капитан гордо сидел в кепке и плаще, слушая краткое повторение рассказа о том, как его команда спасла экипаж опрокинувшегося судна.)

Анализ

Шаблон \bcap соответствует любому слову, которое начинается с cap, и потому были найдены четыре слова, причем три из них не являются словом cap.

Далее приведен тот же самый пример, но только \b стоит в конце слова:

Текст

```
The captain wore his cap and cape proudly as he sat
listening to the recap of how his crew saved the men from
a capsized vessel.
```

(Капитан гордо сидел в кепке и плаще, слушая краткое повторение рассказа о том, как его команда спасла экипаж опрокинувшегося судна.)

Регулярное выражение

```
cap\b
```

Результат

```
The captain wore his cap and cape proudly as he sat
listening to the recap of how his crew saved the men from
a capsized vessel.
```

(Капитан гордо сидел в кепке и плаще, слушая краткое повторение рассказа о том, как его команда спасла экипаж опрокинувшегося судна.)

Анализ

Выражение `cap\b` соответствует любому слову, которое заканчивается на `cap`, и потому были найдены два совпадения, включая и то слово, которое не является словом `cap`.

Но если нужно найти только слово `cap`, правильным шаблоном является только `\bcap\b`.

На самом деле символ `\b` не соответствует какому-либо символу; он соответствует позиции. Поэтому длина строки, которая находится шаблоном `\bcat\b`, равна трем (`c`, `a` и `t`), а не пяти символам.

Чтобы указать нечто, не соответствующее границе слова, используйте `\B`. В следующем примере метасимволы `\B` помогают определить местонахождение дефисов с лишними пробелами вокруг них:

Текст

```
Please enter the nine-digit id as it appears on your
color - coded pass-key.
```

(Пожалуйста, введите девятицифровый идентификатор, который имеется на вашей цветокодированной отмычке.)

Регулярное выражение

```
\B-\B
```

Результат

```
Please enter the nine-digit id as it appears on your color -
coded pass-key.
```

(Пожалуйста, введите девятицифровый идентификатор, который имеется на вашей цветокодированной отмычке.)

Анализ

`\B-\B` соответствует дефису, который окружен символами границы слова. Дефисы в `nine-digit` и `pass-key` не соответствуют шаблону, но дефис в `color - coded` будет найден.

- Как указывалось в уроке 4, “Использование метасимволов”, метасимволы на верхнем регистре обычно отрицают функциональ□4□д(де)□воерх

В уроке 3, “Соответствие набору символов”, вы узнали, что крыша ^ используется для того, чтобы отрицать набор. Как же она может использоваться также и для указания начала строки?

Метасимвол ^ имеет несколько значений. Этот метасимвол отрицает набор, только если находится в наборе (т.е. заключен в квадратные скобки [и]) и является первым символом после открывающей [. Вне набора и в начале шаблона ^ соответствует началу строки.

Чтобы продемонстрировать использование границ строк, рассмотрим следующий пример. Правильные (допустимые) XML-документы начинаются с <?xml> и, вероятно, имеют дополнительные атрибуты (например, номер версии, как в <?xml version="1.0" ?>). Ниже мы просто проверяем, является ли текст XML-документом:

Текст

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

Регулярное выражение

```
<\?xml.*\?>
```

Результат

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

Анализ

Шаблон, казалось, сработал. <\?xml соответствует <?xml, .* соответствует любому другому тексту (нуль или больше экземпляров .), а \?> соответствует ?> в конце тега.

Но это очень грубая проверка. Рассмотрим следующий пример: тот же шаблон используется для того, чтобы найти текст, которому предшествует лишний текст перед открытием XML.

Текст

```
This is bad, real bad!
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

(Это плохо, очень плохо!..)

Регулярное выражение

```
<\?xml.*\?>
```

Результат

```
This is bad, real bad!
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap=http://xml.apache.org/xml-soap
```

(Это плохо, очень плохо!..)

Анализ

Шаблон `<\?xml.*\?>` соответствует второй строке текста. И хотя тег, открывающий XML, действительно может быть помещен во второй строке текста, этот пример определенно недопустим (и обработка такого текста как XML-документа может вызвать всевозможные проблемы).

Необходимо проверить, что открывающий XML-тег на самом деле является первым текстом в строке, и именно это должен сделать метасимвол ^:

Текст

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

Регулярное выражение

```
^\s*<\?xml.*\?>
```

Результат

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://tips.cf"
xmlns:impl="http://tips.cf" xmlns:intf="http://tips.cf"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
```

Анализ

Первый символ ^ соответствует началу строки, поэтому ^\s* соответствует началу строки, после которого следует нуль или большее количество пробельных символов (таким образом обрабатываются допустимые пробелы, позиции табуляции и концы строк перед открытием XML-документа). Следовательно, весь шаблон ^\s*<\?xml.*\?> соответствует открывающему XML-тегу с любыми атрибутами, причем он правильно обрабатывает также и пробельные символы.

Шаблон ^\s*<\?xml.*\?> тоже работал, но только потому, что XML-документ, приведенный в этом примере, не завершен. Если бы использовался весь XML-документ, вы бы увидели пример работы жадного квантора. Это, вероятно, был бы самый убедительный пример того, когда нужно использовать .*? вместо просто .*.

Знак $ используется во многом аналогичным способом. Этот знак помогает проверить, что после закрывающего тега </html> на Web-странице ничего нет:

Регулярное выражение

```
</[Hh][Tt][Mm][Ll]>\s*$
```

Анализ

Наборы используются для каждого из символов H, T, M и L (чтобы обработать любую комбинацию символов верхнего и нижнего регистров), а \s*$ соответствуют любому пробельному символу, за которым следует конец строки.

Шаблон ^.*$ — синтаксически правильное регулярное выражение, которое почти всегда будет находить соответствие, и потому оно совершенно бесполезно. Попробуйте догадаться, чему соответствует данное выражение и когда оно не будет находить соответствие.

Использование многострочного режима

Обычно ^ соответствует началу строки, а $ — концу строки. Однако есть исключение, а точнее, способ изменить такое поведение этих символов.

Во многих реализациях регулярных выражений имеются специальные метасимволы, которые изменяют поведение других метасимволов; один из них — (?m). Этот метасимвол допускает использование многострочного режима. Многострочный режим вынуждает механизм регулярных выражений обрабатывать конец строки как разделитель строк, и в этом режиме ^ соответствует началу строки или началу после конца строки (т.е. началу новой строки), а $ соответствует концу строки или концу после конца строки.

Метасимвол (?m) (если он, конечно, применяется) должен быть помещен в самое начало шаблона, как показано в следующем примере, в котором регулярное выражение помогает определить местонахождение всех комментариев в блоке кода, написанном на JavaScript:

Текст

```
<SCRIPT>
function doSpellCheck(form, field) {
    // Make sure not empty
    if (field.value == '') {
        return false;
    }
    // Init
    var windowName='spellWindow';
    var

spellCheckURL='spell.cfm?formname=comment&fieldname='+field.
        name;
    ...
    // Done
    return false;
}
</SCRIPT>
```

Регулярное выражение

```
(?m)^\s*//.*$
```

Результат

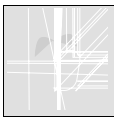
```
<SCRIPT>
function doSpellCheck(form, field) {
    // Make sure not empty
```

```
    if (field.value == '') {
        return false;
    }
    // Init
    var windowName='spellWindow';
    var

spellCheckURL='spell.cfm?formname=comment&fieldname='+field.
        name;
    ...
    // Done
    return false;
}
</SCRIPT>
```

Анализ

Выражение `^\s*//.*$` соответствует началу строки, за которым следует любой пробельный символ, за ним в свою очередь следует `//` (начало комментария в JavaScript), за которым размещается любой текст, после чего следует конец строки. Однако использованный нами шаблон соответствовал бы только первому комментарию (и то только если бы это был единственный текст на странице). Модификатор `(?m)` в `(?m)^\s*//.*$` заставляет шаблон обрабатывать концы строк как разделители строк, и потому были найдены все комментарии.



Выражение `(?m)` не поддерживается многими реализациями регулярных выражений.

В некоторых реализациях регулярных выражений также поддерживается метасимвол `\A`, соответствующий началу строки, и метасимвол `\Z`, соответствующий концу строки. Если эти метасимволы поддерживаются, то ведут себя они очень похоже на `^` и `$`, но в отличие от `^` и `$`, они не модифицируются метасимволом `(?m)` и поэтому не будут работать в многострочном режиме.

Резюме

Регулярные выражения могут соответствовать любым блокам текста или тексту в определенной позиции в строке. Символ `\b` соответствует границе слова (а `\B` — его противоположность — все делает полностью наоборот). Метасимволы `^` и `$` отмечают границы строк (начало строки и конец строки, соответственно), хотя когда используется модификатор `(?m)`, `^` и `$` будут также соответствовать строкам, которые начинаются или заканчиваются в конце строки (т.е. там, где стоит символ разрыва строки).

✧✧✧