Day 11: Recipes I

Dates, Times, RE Tricks

Suggested Reading:
Perl Cookbook (2nd Ed.)

Chapter 3: Dates and Times
Chapter 6: Pattern Matching
(*especially* 2.1, 6.4, 6.6, 6.9, 6.15)

Homework Review

Homework Preview

Forecast Sample

```
...
<H1>Madison Forecast</H1>
Local Madison Forecast 349 AM CDT WED JUL 28 2010
<br><br><font size=+1><B>TODAY...</B></font>
MOSTLY CLOUDY WITH A 50 PERCENT CHANCE OF
THUNDERSTORMS IN THE MORNING...THEN MOSTLY SUNNY
IN THE AFTERNOON. HIGHS IN THE MID 80S. NORTHWEST
WINDS 5 TO 10 MPH. RAINFALL 0.05 TO 0.10 INCH.
PRECIPITATION DURATION OF UP TO ONE HOUR.
<br><br><font size=+1><B>TONIGHT...</B></font>
COOLER. MOSTLY CLEAR. LOWS IN THE LOWER
60S. NORTHWEST WINDS UP TO 10 MPH.
...
```

Dates and Times

What Is So Hard About This?

Dates

- Different calendars
- Historical calendar changes
- Y2K

Times

- UTC vs. time zones
- Daylight saving time
- Leap years
- Leap seconds
- Indiana (http://en.wikipedia.org/wiki/Time_in_Indiana)

Unix/POSIX/Epoch Time

Seconds since **1970 January 01 @ 0:00** (UTC)

(more or less)

Remaining challenge

Unix time <–> Other time formats

Standard Date/Time Functions

`localtime`	get local YMDHMS from Unix time
`gmtime`	get UTC YMDHMS from Unix time
`Time::Local::timelocal`	create Unix time from local YMDHMS
`Time::Local::timegm`	create Unix time from UTC YMDHMS
`POSIX::mktime`	create Unix time from local YMDHMS
`POSIX::strftime`	format a Unix time

Caution: Read perldoc pages!!!

```
my ($sec, $min, $hour, $mday, $mon, $year,
  $wday, $yday, $isdst) = localtime(time);
my $real_year = 1900 + $year;
```

Parsing Dates and Times

- Use regular expressions

```
if (m,(\d{1,2})/(\d{1,2})/(\d{4}),) {
    my $year = $3 - 1900;
    my $month = $1 - 1;
    my $mday = $2;
}
timelocal(0, 0, 0, $mday, $month, $year);
```

- Use CPAN's **Date::Manip** or **Date::Manip::Date**

Time Interval Calculations

- Use Unix time
  - Convert to Unix time
  - Do math in seconds
  - Convert back for display

```
use Time::Local qw/timelocal/;
my $start = timelocal(0, 0, 11, 12, 6, 110);
my $interval = time() - $start;
# now what?
```

- Use CPAN's **Date::Calc**

Timing Events

- Unix times: second-level resolution

```
my $start = time();
sleep(rand(10));
my $end = time();
my $duration = $end - $start;
```

- **Time::HiRes**: much higher resolution

```
use Time::HiRes qw/gettimeofday/;
my $start = gettimeofday();
sleep(rand(10));
my $end = gettimeofday();
my $duration = $end - $start;
```

Regular Expression Tricks

Greedy vs. Non-Greedy Matches

.* greedy match

```
/a.*z/        azimuth, dazzle, waltz, abuzz, a.*z
              a, z, apples, buzz, Azimuth
```

.*? non-greedy match

```
/a.*?z/       azimuth, dazzle, waltz, abuzz, a.*z
              a, z, apples, buzz, Azimuth
```

- Append **?** to *, **+**, **?**, **{}**
- Good for delimited text

```
my $html = '<img src="/icon.png" alt="icon">';
if ($html =~ m,<img\s+src="(.+?)",) {
    my $image_source = $1;           }
```

Matching Numbers

- What ***kind*** of numbers?

positive integer	`/^\d+$/`
integer	`/^-?\d+$/`
integer (leading + ok)	`/^[+-]?\d+$/`
decimal	`/^-?\d+\.?\d*$/`
decimal	`/^-?(?:\d+(?:\.\d*)?|\.\d+)$/`

- What about others?

```
1_234_567
$1,234.75
($1,234.75)
6.0221415e+23
```

Matching Across Lines I

- Why?

```
<connection><hostname>
  vdt-itb.cs.wisc.edu
  </hostname><port>
8080</port></connection>
```

- How?

```
open(SLURPY, $filename) or die "die: $!";
my $contents = join('', <SLURPY>);
close(SLURPY);

my $long_output = `some_verbose_command`;
```

Matching Across Lines II

`//m` Treat string as multiple lines

`^`, `$` match start, end of any line within string

`//s` Treat string as singe line

`.` matches newline (which normally it does not)

```
<connection><hostname>
  vdt-itb.cs.wisc.edu
  </hostname><port>
8080</port></connection>

my $text = slurp('config.xml');
$text =~ s{(<hostname>\s*).*?(\s*</hostname>)}
          {$1$host$2}im;
```

Glob as Regular Expression

<day-??-*.pl> versus **/^day-..-.*\.pl$/**

```
sub glob2re {
  my $regexp = shift;
  $regexp =~ s/\./\\./g;
  $regexp =~ s/\*/.*/g;
  $regexp =~ s/\?/./g;
  return '^' . $regexp . '$';
}

my $path_re_string = glob2re('*.pl');
my $path_regexp = qr/$path_re_string/;
find(\&find_handler, @directories);
sub find_handler {
    next unless $path_regexp;
```

Commenting Regular Expressions

`//x` Whitespace and comments allowed in RE
Both must be quoted with \ to be part of RE

```
$text =~ s{
    (                   # start of opening
      <hostname>        # open hostname element
      \s *              # maybe some whitespace
    )                   # end of opening
    . * ?               # capture hostname here
    (                   # start of closing
      \s *              # maybe some whitespace
      </hostname>       # end hostname element
    )                   # end of closing
}
{$1$host$2}imx;
```

Homework

Forecast Sample

```
...
<H1>Madison Forecast</H1>
Local Madison Forecast 349 AM CDT WED JUL 28 2010
<br><br><font size=+1><B>TODAY...</B></font>
MOSTLY CLOUDY WITH A 50 PERCENT CHANCE OF
THUNDERSTORMS IN THE MORNING...THEN MOSTLY SUNNY
IN THE AFTERNOON. HIGHS IN THE MID 80S. NORTHWEST
WINDS 5 TO 10 MPH. RAINFALL 0.05 TO 0.10 INCH.
PRECIPITATION DURATION OF UP TO ONE HOUR.
<br><br><font size=+1><B>TONIGHT...</B></font>
COOLER. MOSTLY CLEAR. LOWS IN THE LOWER
60S. NORTHWEST WINDS UP TO 10 MPH.
...
```

Weather Analysis, Part I

- Download current forecast
- Parse forecast timestamp and convert to Unix time
- Do you best to parse *at least* temperature forecasts