Elegant Pharo Code

Beautiful & Powerful One-liners, Expressions and Snippets

Writing computer software remains difficult and hard. Most computer code is hard to read and quite intimidating. This does not have to be the case.

Simple things should be simple, complex things should be possible—Alan Kay

Our software development environments should be designed in such a way that they **make it easy to read and to write code for day to day tasks**, for those problems that are solved.

Studying a list of example tasks—one-liners, expressions and snippets—is an excellent way to check out candidate programming languages. Here is a list with solutions implemented in Pharo—an immersive, live environment including a pure, object-oriented programming language focused on simplicity and immediate feedback.

> *See the '**Pharo Syntax**' section of the article Rediscovering the UX of the legendary HP-35 Scientific Pocket Calculator—A tutorial on Pharo, test and specification based immersive programming for a quick overview of the Pharo language.*

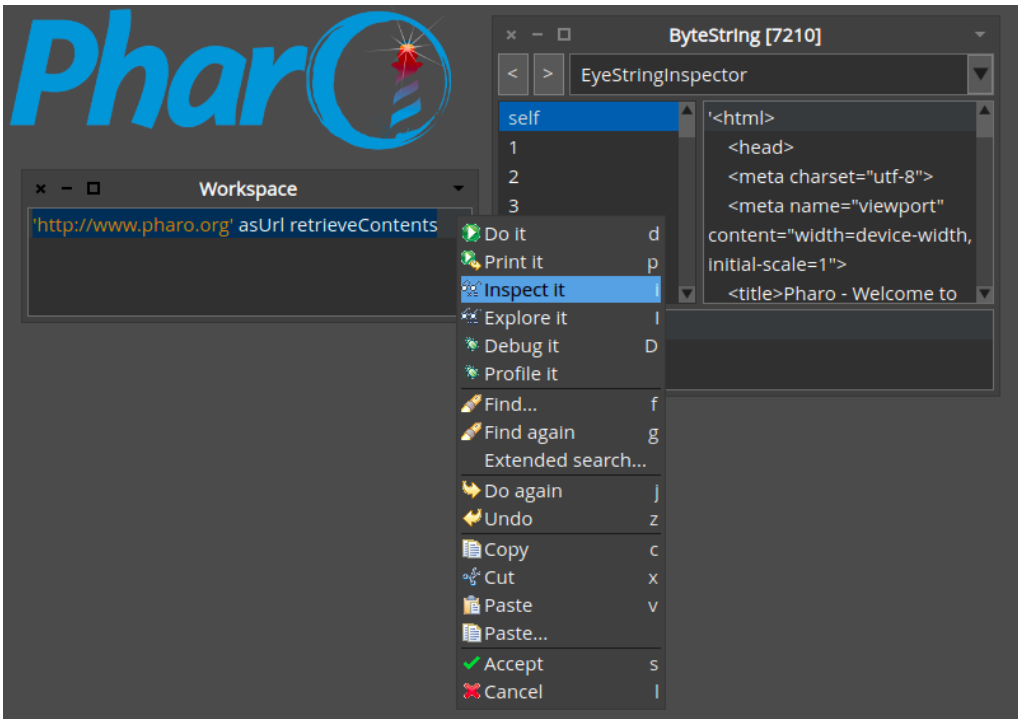
How would you tackle these problems using your favourite language ?

1. Get the HTML source of a web page

The internet is ubiquitous today. It should be dead simple to get the HTML source code of an arbitrary web page.

```
'http://www.pharo.org' asUrl retrieveContents
```

The given String is converted (parsed) to a URL which knows how to retrieve the contents (resource) it points to. When evaluated using the 'Inspect it' operation, an Inspector tool will open on the result, a String.



```
'http://www.pharo.org' asUrl retrieveContents
```

```
'<html>
  <head>
  <meta charset="utf-8">
  <meta name="viewport"
content="width=device-width,
initial-scale=1">
  <title>Pharo - Welcome to
```

2. Compute difference in days between two dates

Date or calendar arithmetic can be tricky. It should not be, given the proper abstractions.

```
('2014-07-01' asDate - '2013/2/1' asDate) days
```

Two Strings are parsed to Dates—note how the syntax of the dates allows for some flexibility. Subtracting two Dates results in a Duration, which knows how many whole days it compromises—515 days.

3. Decimal digit length of 42!

How many digits are there in the decimal representation of the factorial (!) of 42 (thus the product 1*2*3*4* … *42) ?

```
42 factorial decimalDigitLength
```

In Pharo there is no limit to Integers, they can become as large as needed, just like in real math. These Integers are also quite intelligent, they know their *decimalDigitLength* for example—in this case, 52 digits.

4. Set up an HTTP server that returns the current timestamp

Like client HTTP functionality, server HTTP functionality should be easy. Here is one expression to set up and run a local HTTP server that will reply to all requests with the current timestamp.

```
(ZnServer startDefaultOn: 8080)
  onRequestRespond: [ :request |
    ZnResponse ok: (ZnEntity with: DateAndTime now printString)
]
```

Try going to http://localhost:8080 if you want to know the current date and time, up to the nanosecond—as in 2014-07-07T17:55:03.638992+02:00

5. Split a string on dashes, reverse the order of the elements and join them using slashes

Much information in computers is represented as Strings. Being able to manipulate Strings in many different ways is thus important. Say we want to convert from a European to an American date format, textually.

```
$/ join: ($- split: '1969-07-20') reverse
```

The above example first splits a String using a dash as separator. Then the order of elements is reversed. Finally they are joined using a slash into a new String. The result is thus ‘20/07/1969’—the date of the first Moon landing.

6. Convert all JPG files in the current directory to PNG format

Many developers are more comfortable with there own programming language than with shell scripting. There is no reason not to try doing typical file manipulations from withing Pharo.

```
(FileLocator workingDirectory filesMatching: '*.jpg') do: [
:each |
  (each withExtension: 'png') writeStreamDo: [ :out |
    each readStreamDo: [ :in |
      (PNGReadWriter on: out) nextPutImage:
        (JPEGReadWriter on: in) nextImage ] ] ]
```

Step one is iterating over all files matching a wildcard. For each of these files, we create a new file with a 'png' extension. We open a write stream, *out*, to the new PNG file and a read stream, *in*, to the existing JPG file. We use two helper classes, PNGReadWriter and JPEGReadWriter, to respectively write and read the PNG and JPG formats.

7. Sum of the primes up to 64

We can use the built in prime sieve to give us a collection of the primes not larger than 64. Summing all numbers in a collection is just as easy.

```
(Integer primesUpTo: 64) sum
```

8. Extract a Unix format timestamp from the 5th to 8th byte of a byte array given in hex

Even though Pharo is a high level language, it excels at low level bit and byte manipulation. ByteArrays are often given as hexadecimal Strings. These can be used to create a ByteArray. Manipulating a ByteArray is no different from manipulating any other collection.

```
DateAndTime fromUnixTime:
  ((ByteArray readHexFrom: 'CAFEBABE4422334400FF')
      copyFrom: 5 to: 8) asInteger
```

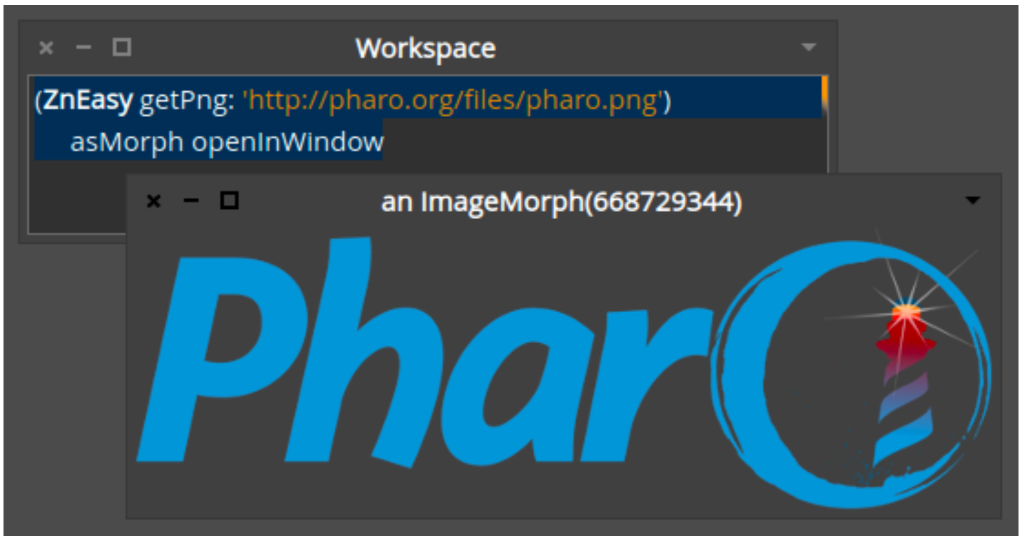
After making a sub selection, we convert the bytes in network order to an Integer. Unix time is defined as the number of seconds since midnight

January 1, 1970 UTC. The DateAndTime class knows how to use this offset to create a proper instance. The answer is 2006-03-23T05:33:56Z.

9. Show a PNG image retrieved from a URL

URLs are not only used to point to web pages, but to any kind of resources, including images. Retrieving such a PNG image and actually looking at it should be easy.

```
(ZnEasy getPng: 'http://pharo.org/web/files/pharo.png')
  asMorph openInWindow
```



```
(ZnEasy getPng: 'http://pharo.org/files/pharo.png')
   asMorph openInWindow
```

10. Verify that a fraction is close to ∏

Although the mathematical constant ∏ (pi) is an irrational number, approximately 3.141592653589793, there is a remarkably simple, easy to remember fraction that comes pretty close to it.

```
355 / 113 closeTo: Float pi
```

Pharo works with true Fractions (objects holding the simplified nominator and denominator, keeping the full precision using unlimited Integers) as well as with Floats. The *closeTo:* operation compares any two numbers using limited precision—using about 5 significant digits. The above expression thus return true.

11. Test whether one set is included in another one

Working with collections, like sets, should be simple and straightforward. Testing if one set is included in another one is such a simple operation.

```
#(a b c d e f) includesAll: #(f d b)
```

The above is correct, but maybe cheats a bit because all Collections in Pharo understand *includesAll:*—it is too easy.

```
(#(a b c d e f) asSet intersection: #(f d b) asSet)
  = #(b f d) asSet
```

Another approach is to use the mathematical property that if one set belongs to another, their intersection equals the the original one. In this case the constant literal Arrays are explicitly converted to Sets. Note how we changed the order of elements to emphasise that their order is irrelavant.

12. Show a digital clock

As a live environment, Pharo is able to show any object and will refresh its view when the object is changing. This allows us to easily construct a primitive digital clock.

```
Time now asValueHolder in: [ :clock |
  [ [
      1 second asDelay wait.
      clock value: Time now ] repeat ] fork.
  clock inspect ]
```

Here we use a ValueHolder, an object holding another one, to represent our clock. We fork a process that will update the clock's value every second. Note how we open the Inspector manually. This will show and refresh the contents inside the ValueHolder.

13. Average of the prime factors of 2^32—1

Showing off some of the built in functionality: computing prime factors and averaging lists of numbers. The answer being 65819/5 or 13163.8

```
(2**32 - 1) primeFactors average
```

14. Load XML Support using SCM from a public repository

It should be simple to load external libraries and frameworks from public repositories using good source code management (SCM) tools. Although this can be done using a UI tool, it should be easy in code as well.

```
Gofer it
  smalltalkhubUser: 'PharoExtra' project: 'XMLSupport';
  configuration;
  loadStable
```

This simple expression results in 'XML Support' being loaded while resolving all dependencies and versions—behind the scenes, a surprising number of subsystems are working together to make this possible.

15. Generate a random string

Here is one way to generate a String containing random letters—selected from the first 6 letters of the alphabet.

```
(String new: 32) collect: [ :each | 'abcdef' atRandom ]
```

An empty String of the proper size is created and transformed by changing each element (each Character) with a random one selected from a given, constant String, 'abcdef'.

16. Return the weekday of a date

A simple question that should have a simple solution—Tuesday in this case.

```
'2013/5/7' asDate dayOfWeekName
```

Converting (parsing) a String to a Date object gives us all we need: a Date knows its *dayOfWeekName,* among many other things.

17. Compute the standard deviation of a list of numbers

Mathematics can be beautiful and elegant, provided you can express things the right way. The corrected sample standard deviation is such a concept. It is the square root of the sample variance, which is the average of the squared deviations about the sample mean (about the average).

```
#(2 4 4 4 5 5 7 9) in: [ :input |
  ((input - input average) squared sum / (input size - 1)) sqrt
]
```

Note how the first subtraction, the average, the squaring and sum operate over the whole collection of numbers. The result being 2.138089935299395

Actually, this common operation is also built in:

```
#(2 4 4 4 5 5 7 9) stdev
```

18. Save the HTML source of a web page to a file

Quickly saving the HTML source of a web page to a file should be trivial.

```
'http://www.pharo.org' asUrl saveContentsToFile: 'page.html'
```

19. Basic grep on a file using a regular expression

Most programmers are familiar with *grep*, a Unix command line tool to quickly find the lines in a file or in files matching a pattern, often specified using regular expressions. Here is an implementation of a basic grep on a single file.

```
'^.*.jpg' asRegex in: [ :regex |
  '/tmp/foo.txt' asFileReference contents lines
    select: [ :line | regex matches: line ] ]
```

The pattern String is converted to an object that knows how to match lines. The file is opened by taking its *contents* and converting it to *lines* which are selected when they match.

20. Count the number of, or show the leap years between two years

Leap years can be tricky. How many leap years are there between 1914 and 1945? 8. Which are the leap years between 1895 and 1915 ? 1896, 1904, 1908 and 1912.

```
(1914 to: 1945) count: [ :each | Year isLeapYear: each ].
(1895 to: 1915) select: [ :each | Year isLeapYear: each ].
```

21. Find out which server is serving a web page

As a live environment, Pharo allows you to look at all objects involved in a computation. This includes the HTTP request and response objects. We can access the response object to find out which server is serving a particular web page—by consulting the identification String in the 'Server' header. Pharo.org is served by 'Zinc HTTP Components 1.0'.

```
(ZnClient new beOneShot; get: 'http://www.pharo.org'; response)
   headers at: #Server
```

In this case we are not interested in the result of the GET request itself, but in the response object. We also tell the HTTP client that it will only be used once—so that it can close the connections cleanly.

22. Encode the same string using Latin1, UTF-8 and UTF-16

A modern language should have Strings that can contain any Unicode character. Encoding these Unicode Strings as bytes for storage or transmission requires the use of an encoding. Pharo can encode/decode its Strings to/from bytes using multiple encodings. Here is how to compare the result of 3 such encoders.

```
#(latin1 utf8 utf16) collect: [ :each |
   (ZnCharacterEncoder newForEncoding: each)
      encodeString: 'Les élèves Français' ]
```

The result is an Array containing the following 3 ByteArrays, written here as hex strings:

```
4C657320E96CE87665732046726616EE7616973

4C657320C3A96CC3A87665732046726616EC3A7616973

004C00650073002000E9006C00E80076006500730020004600720061006E00E7
006100690073
```

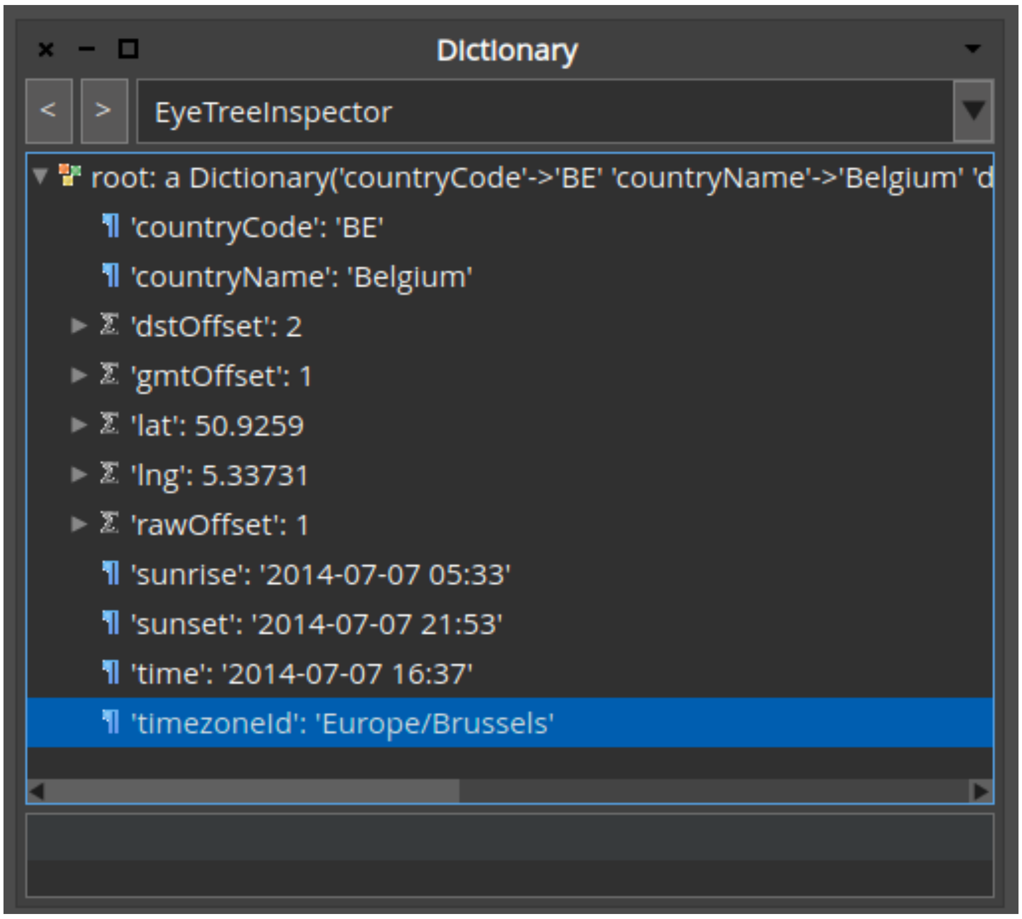
23. Get the timezone in effect in a location via a web service call

Web services or remote API's can be found everywhere today. It should be easy to access such services and interpret the results. Below is an example of calling a web service offered by http://geonames.org for computing the timezone in effect at a certain location given by longitude/latitude coordinates.

```
ZnClient new
  url: 'http://api.geonames.org/timezoneJSON';
```

```
    queryAt: #lng put: 5.33731;
    queryAt: #lat put: 50.9259;
    queryAt: #username put: 'demo';
    contentReader: [ :entity |
      NeoJSONReader fromString: entity contents ];
    get
```

To be able to run this snippet, you will have to load NeoJSON and get a free account for your own username. The resulting JSON will be parsed as a Dictionary.



```
root: a Dictionary('countryCode'->'BE' 'countryName'->'Belgium' 'd
  'countryCode': 'BE'
  'countryName': 'Belgium'
  'dstOffset': 2
  'gmtOffset': 1
  'lat': 50.9259
  'lng': 5.33731
  'rawOffset': 1
  'sunrise': '2014-07-07 05:33'
  'sunset': '2014-07-07 21:53'
  'time': '2014-07-07 16:37'
  'timezoneId': 'Europe/Brussels'
```

Programming vs Scripting

The examples given here should be considered as scripting and interactive exploration of the computational capabilities of Pharo. Designing and writing actual programs in Pharo is another story and is necessarily a bit more involved.

Furthermore, working in/with Pharo is a very interactive, immersive experience—involving code completion, syntax highlighting, code browsing, object inspection and debugging—that you have to try for yourself to appreciate fully.

Hopefully these examples have piqued your interest—there is much more to discover, learn and explore. Head over to http://pharo.org and get started. I can guarantee you that it will be interesting.

> *Feedback, remarks, comments ? Alternative solutions, other examples ? Feel free to contact me. Written by Sven Van Caekenberghe, July 2014*