

# Budowa aplikacji SaaS z użyciem pakietu opencpu

Przemysław Biecek

6 XII 2015

*Zróbmy sobie aplikacje SaaS!*

Za Wikipedią <sup>1</sup>

*Software as a Service (w skrócie SaaS, z ang. oprogramowanie jako usługa) – jeden z modeli chmury obliczeniowej, polegający na dystrybucji oprogramowania, w którym aplikacja jest przechowywana i udostępniana przez producenta użytkownikom przez Internet. Eliminuje to potrzebę instalacji i uruchamiania programu na komputerze klienta. Model SaaS przerzuca obowiązki zarządzania, aktualizacji, pomocy technicznej z konsumenta na dostawcę. W efekcie użytkownik oddaje kontrolę nad oprogramowaniem dostawcy i obowiązek zapewnienia ciągłości jego działania.*

Dzisiaj będziemy pracować z lokalną instalacją serwera opencpu. Do poważniejszych aplikacji potrzebowalibyśmy instalacji na zdalnym serwerze. Instrukcje dotyczące tego jak można pobrać i zainstalować wersję serwerową i lokalną, znajdują się na stronie projektu opencpu <sup>2</sup>.

*Ale o co chodzi?*

Zacznijmy od przykładu. Uruchomimy lokalnie serwer na porcie 4848.

```
library(opencpu)
opencpu$stop()
opencpu$start(4348)
```

Po włączeniu pakietu na porcie 4348 podłącza się serwer OpenCPU. Można go zobaczyć, np. sprawdzając jakie porty są otwarte. Na OSX można to zrobić poleceniem Network Utility.

*Testowa praca z serwerem*

Interface do testowania aplikacji jest dostępny pod adresem `http://localhost:4348/ocpu/test/`

Używając tego interfejsu możemy przećwiczyć podstawowe tryby pracy z serwerem.

*Normalna praca z serwerem*

Używając komunikacji protokołem GET/POST możemy wywołać następujące akcje



<sup>1</sup> [https://pl.wikipedia.org/wiki/Software\\_as\\_a\\_Service](https://pl.wikipedia.org/wiki/Software_as_a_Service)

<sup>2</sup> <https://www.opencpu.org/download.html>

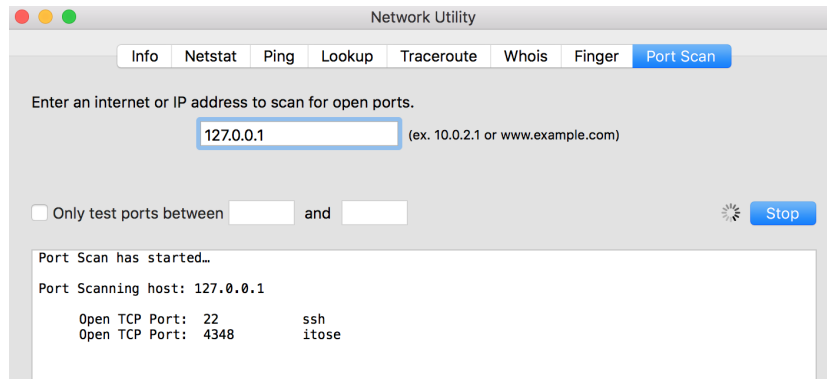


Figure 1: Program Network Utility pokazuje serwer R nasłuchujący na porcie 4348.

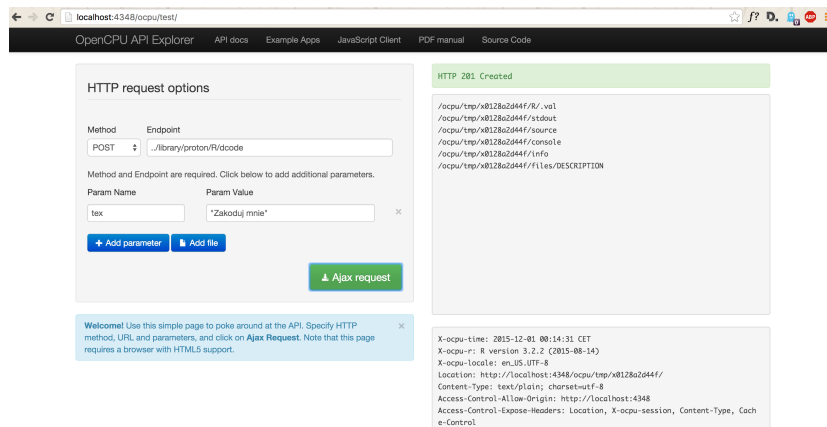


Figure 2: Testowa aplikacja pozwalająca na interaktywne odpytywanie serwera opencpu przez protokoły GET i POST.

- Informacja o obiektach udostępnionych przez pakiet `http://localhost:4348/ocpu/library/stats/`
- Lista funkcji i zbiorów danych eksportowanych przez pakiet `http://localhost:4348/ocpu/library/stats/R` `http://localhost:4348/ocpu/library/stats/data`

### *Komunikacja z serwerem*

Komunikację klienta R z serwerem opencpu przez protokół GET/POST można przeprowadzić:

- z użyciem przeglądarki internetowej (protokół GET),
- używając linii poleceń, wywołując np. polecenie `curl http://localhost:4348/ocpu/library/stats/data/`
- używając R i np funkcji `GET()` i `POST()` z pakietu `httr`

```
library(httr)
httr::GET("http://localhost:4348/ocpu/library/stats/data")
httr::GET("http://localhost:4348/ocpu/library/stats/R/rnorm")
httr::GET("http://localhost:4348/ocpu/library/proton/R/dcode")
```

### *Formatowanie danych*

Server openCPU automatycznie konwertuje wyniki na różne formaty, w zależności od wskazanego sufiksu.

Przykładowy, zbiór danych o kotach i ptakach z konwersją wyników do formatów json/csv/rda.

- `http://localhost:4348/ocpu/library/PogromcyDanych/data/koty_ptaki/print`
- `http://localhost:4348/ocpu/library/PogromcyDanych/data/koty_ptaki/json`
- `http://localhost:4348/ocpu/library/PogromcyDanych/data/koty_ptaki/csv`
- `http://localhost:4348/ocpu/library/PogromcyDanych/data/koty_ptaki/rda`

### *Formatowanie funkcji*

Podobnie dla funkcji dostępna jest metoda `print`<sup>3</sup> oraz dokumentacja dostępna przez API

<sup>3</sup> `http://localhost:4348/ocpu/library/stats/R/rnorm`

- `http://localhost:4348/ocpu/library/stats/man/rnorm/`
- `http://localhost:4348/ocpu/library/stats/man/rnorm/text`
- `http://localhost:4348/ocpu/library/stats/man/rnorm/html`
- `http://localhost:4348/ocpu/library/stats/man/rnorm/pdf`

## Zdalne wykonywanie funkcji

Protokół GET służy pobieraniu danych oraz ciał funkcji. Aby wywołać zdalnie funkcję należy wykorzystać protokół POST.

Argumenty dla funkcji można podawać jako dodatkowe argumenty.

```
(tmp <- httr::POST("http://localhost:4348/ocpu/library/stats/R/rnorm",
  body = list(n = "5")))
```

Po zdalnym wywołaniu funkcji na serwerze powstaje zapis sesji z wynikami przetwarzania.

- Lista obiektów w sesji `http://localhost:4348/ocpu/tmp/x0abff4dc2d/`
- Lista obiektów R zapisanych w sesji `http://localhost:4348/ocpu/tmp/x0abff4dc2d/R/`

Możemy te wyniki odczytać w R na wiele sposobów.

```
# odczytywanie wyników parsując wynik funkcji
# print
(tmp <- httr::GET("http://localhost:4348/ocpu/tmp/x0abff4dc2d/R"))
rawToChar(tmp$content)

# odczytywanie formatu rda przez load
load(url("http://localhost:4348/ocpu/tmp/x0abff4dc2d/R/.val/rda"))
.val

# odczytywanie formatu csv przez readLines
readLines(url("http://localhost:4348/ocpu/tmp/x0abff4dc2d/R/.val/csv"))
```

## Zdalne wykonywanie wykresów

Wynikiem zdalnego wywołania funkcji mogą być nie tylko obiekty R, ale też grafika w formacie png, pdf, svg

```
httr::POST("http://localhost:4348/ocpu/library/graphics/R/hist",
  body = list(x = "[0,1,1,1,1,5]", col = "'black'"))
```

Odwołać do wyników można się przez link

`http://localhost:4348/ocpu/tmp/x0fadb8920d` `http://localhost:4348/ocpu/tmp/x0fadb8920d/graphics/1/pdf`

## Sesja

Wynik wywołania funkcji - cała sesja - może być pobrana jako jeden plik

```
http://localhost:4348/ocpu/tmp/x0ba6b83f97/zip
Wynik przetwarzania może być wyświetlony
http://localhost:4348/ocpu/tmp/x0ba6b83f97/console/text
```

### *Własne pakiety*

Wywoływanie zdalnej funkcji ma pewne zalety. Ale szczególnie użyteczne jest udostępnianie własnych pakietów z funkcjonalnościami do zdalnego wywoływania.

Dowolne funkcjonalności można udostępniać zamykając je w funkcje nowych pakietów

```
# równoważnie
proton:::dcode("Zakoduj to")
# równoważnie
http:::POST("http://localhost:4348/ocpu/library/proton/R/dcode",
  body = list(tex = "'Zakoduj to'"))

# pobieramy wyniki v1
load(url("http://localhost:4348/ocpu/tmp/x050588d598/R/.val/rda"))
.val

# pobieramy wyniki v2
readLines("http://localhost:4348/ocpu/tmp/x050588d598/R/.val/csv")
```

### *Przekazywanie argumentów do funkcji*

Wywołując zdalnie funkcję możemy wskazać wartość, wyrażenie lub wynik z innego wyrażenia.

```
# argumenty przekazane jako wartości
http:::POST("http://localhost:4348/ocpu/library/stats/R/rpois",
  body = list(n = "1", lambda = "10"))

# przekazanie kodu R jako argumentu
http:::POST("http://localhost:4348/ocpu/library/stats/R/rexp",
  body = list(n = "sqrt(9)", rate = "1"))

# przekazanie wyników jednej funkcji jako
# argumentu drugiej
http:::POST("http://localhost:4348/ocpu/library/stats/R/rexp",
  body = list(n = "x01bbf1e095", rate = "1"))
```

### Serwis *public.opencpu.org*

Zdalne wykonanie funkcji można przećwiczyć na serwisie udostępnionym przez *public.opencpu.org*.

```
res <- http::POST("https://public.opencpu.org/ocpu/library/stats/R/rexp",
  body = list(n = "2+12", rate = "1"))

# Wyniki można odczytać i wykorzystać jak w
# przypadku poprzednich analiz
# https://public.opencpu.org/ocpu/tmp/x06921c10f6/R
```

### Jak wywołać większy fragment kodu

Używając funkcji identity możemy skrypt do wykonania przesłać jako argument funkcji `identity()`

```
http::POST("https://public.opencpu.org/ocpu/library/base/R/identity",
  body = list(x = "coef(lm(speed~dist, data=cars))"))

# Wyniki można odczytać i wykorzystać jak w
# przypadku poprzednich analiz
# https://public.opencpu.org/ocpu/tmp/x044bc7b4f3/R
```

### Linki

Instrukcja jak zainstalować pakiet *opencpu* na serwerze <sup>4</sup> oraz szersza dokumentacja pakietu dostępna jest na stronie projektu *opencpu* <sup>5</sup>.

<sup>4</sup> <https://www.opencpu.org/download.html>

<sup>5</sup> <https://www.opencpu.org/api.html#api-root>

### Zadanie

W zespołach dwuosobowych należy przygotować API do usługi, która (każdy zespół robi jedną z niżej wymienionych rzeczy):

- A. Dla wskazanej marki samochodu wyznacza tempo spadku ceny w kolejnych latach i rysuje tę zmianę na wykresie.
- B. Dla wskazanego rocznika i przebiegu wyznacza średnią cenę aut o danych parametrach.
- C. Dla wskazanej marki samochodu i rocznika wyznacza 5 najtańszych aut z danej marki i danego rocznika.
- D. Spróbuj zdalnie wywołać tę funkcję ze swojego komputera i komputera kolegi.