Informatika pro moderní fyziky (9) zpracování dat se vzájemnými vazbami, složitější interaktivní dokument

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2013/2014
10. prosince 2013

Obsah

- procvičení zpracování dat
- stylování dokumentů s CSS
- vektorové obrázky na webu – SVG
- procvičení zpracování dat

Obsah

Zadání

- zpracovat data o docházce do práce z ledna 2013
- soubor lide.csv obsahuje čísla průkazů a jména
- soubor dochazka.csv obsahuje datum, číslo průkazu, čas příchodu, čas odchodu
- kolik hodin byl kdo v lednu v práci?
- které dny kdo chyběl?

Inspirace

- mám tady vztah číslo-člověk a pak záznamy číslo+časy
- mapa číslo-člověk je typický příklad použití hashe!
- pro agregaci záznamů ze dnů k lidem je také chytré použít hash (číslo průkazu jako klíč)
- pomněte paradigma `hash[key] ||= []`

Obsah

Co už máme

- grafy pro jednotlivé detektory ve formátu PNG
- mapy zóny, ale zatím neklikací (SVG)
- umíme načíst data o jednotlivých detektorech – víme, které pozice jsou v jednotlivé dny obsazeny

Co nám chybí

- mít schované i názvy souborů pro jednotlivé detektory
- vyrobit si HTML dokument, který by zobrazoval jednotlivé mapy
- doplnit interaktivitu do SVG obrázků
- zobrazovat po kliknutí do mapy správný graf

Další JS chytrosti

- v jQuery už známe `$('#id')`
- ale ve skutečnosti jde použít jakýkoli CSS selektor, takže třeba `$('p')`
- pokročilý CSS selektor – vnoření: `#my_list img` vybere všechny obrázky (`img`) které jsou uvnitř elementu s id `my_list`
- ... použiju v situacích, kdy chci schovat nějakou množinu elementů a pak jeden z nich zobrazit (tj. když mám hromadu obrázků a chci, aby byl vidět jen jeden)

Další CSS chytrosti

- normálně se jednotlivé elementy řadí pod sebe
- můžu místo toho použít tzv. floating, kdy se začnou elementy řadit nalevo nebo napravo
- efekt znáte např. z webových galerií, kde mi fotky vyplní celou šířku okna a jdou po řádcích
- `float:left`
- barvu pozadí nastavím např. `background-color:red` nebo `background-color:#ddddff`

Další SVG chytrosti

- kromě `rect` se bude hodit také `text`
- jako text se zobrazí obsah příslušného elementu
- opět použiju atributy `x`, `y` (levý dolní roh) a můžu přihodit `text-anchor="middle"`, aby to byl dolní prostředek
- pozor, text mi překryje čtvereček, takže budu muset zopakovat onclick!

Obsah

Zpracování cizích zdrojů na webu – HTML scraping

- dosud jsme zpracovávali pouze lokální, hezky formátovaná data
- i leckteré externí služby poskytují pěkná API ve formátech JSON nebo XML
- ale leckdy taky ne a zajímavé informace
- protože jsme chytré horákyně, naučíme se, jak data automaticky získat

Zadání úkolu

- protože po práci si chceme oddychnout a netrápit se přitom náročnou intelektuální činností, přečteme si rádi dobrý komiks
- a protože jsme přiměřeně cyničtí a také si chceme pocvičit angličtinu, přečteme si redmeat
- www.redmeat.com
- .. ale nechceme klikat a nechceme číst na internetu, takže si zhotovíme PDFko se všemi díly najednou
- zvládneme to snadno v LaTeXu, ale potřebujeme postahovat ty obrázky

Knihovny pro Ruby – příkaz require

- už umíme používat `require` pro lokální soubory
- také funguje pro *core library* – soubor knihoven dodávaných v rámci distribuce Ruby
- kromě toho můžu využít i další zdroj knihoven – ruby gems

Knihovny pro Ruby – *Ruby Gems*

- viz www.rubygems.org
- nepřeberné hromady
- require ‘rubygems’
- require ‘něco’

Správa knihoven – bundler

- nejjednodušší je instalovat knihovny někam “k sobě”
- `bundle -v`
- `bundle init`
- Gemfile – seznam potřebných gemů(knihoven)
- `bundle install -path vendor/bundle`

Gem nokogiri

- nejlepší nástroj na parsování XML/HTML pod Ruby
- nevýhoda – binární (C extension), takže mohou být potíže s instalací
- umí spoustu věcí, nám postačí přistupovat k elementům pomocí CSS selektorů (protože ty už umíme)
- každý element má metody `attributes` a `text`

Vypiš všechny odkazy

Otevřu si dokument (buď řetězec, nebo jakýkoli *stream*, což je například soubor) a pak můžu iterovat přes všechny elementy vyhovující danému selektoru:

```
doc = Nokogiri::HTML(File.open("redmeat.html"))
doc.css("a").each do |x|
  puts "#{x.text} => #{x.attributes['href']}"
end
```

Jak na to

- `http://www.redmeat.com/`, meat locker
- podíváme se na zdroják a koukáme – vidíme, že máme velké štěstí, protože tam jsou dva `ul` seznamy
- po rozkliknutí a inspekci vidíme, že se obrázek jmenuje stejně, což je výhra
- stačí tedy rozparsovat dokument se seznamem, tak si ho uložíme na disk

Stahování dat

Součást standardní knihovny – `open-uri`

```
require 'open-uri'
File.open(local_filename, 'wb') do |f2|
  open(remote_url, 'rb') do |f1|
    f2.write f1.read
  end
end
```

Relativní URL

- http://www.redmeat.com/redmeat/meatlocker/index.html
- relativní `../2013-12-03/index.html`
- → http://www.redmeat.com/redmeat/2013-12-03/index.html
- relativní `index-1.gif`
- → http://www.redmeat.com/redmeat/2013-12-03/index-1.gif

Problémy s formáty

- stažené obrázky jsou ve formátu GIF
- LaTeX umí z rastrů jen PNG a JPG
- potřeba převést – pro automatizaci je vhodný balík `ImageMagick`
- ale protože jsem nezařídil jeho nainstalování, tak to neuděláme

Vygenerovat PDF

- zase triviální, už to umíme, rychlá akce na deset minut!
- ERb šablona, použít `erb_compiler`
- seznam souborů vzít z `Dir["redmeat_*.png"]`
- co a jak s LaTeXem viz např. šestou přednášku

A to je vše, přátelé!