Informatika pro moderní fyziky (10) Generování vektorových obrázků. Konfigurační soubory YML, formát JSON, použití cizích API

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2020/2021
7. prosince 2022

Obsah

- použití ERb šablon
- základy tvorby obrázků v SVG

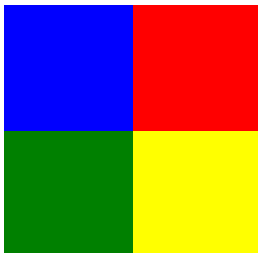
Obsah

Zadání dnešní úlohy

- pro zadanou textovou mapu AZ VR1 potřebuju udělat hezký obrázek
- co druh, to barvička, rozumně zacházet s odstíny (palivo různě modré, R/B/E tyče různě červené, zelené, fialové)

Jednoduchý příklad SVG

```
<svg width="320" height="320" xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect x="0.0" y="0.0" width="40.0" height="40.0" fill="blue" />
  <rect x="40.0" y="0.0" width="40.0" height="40.0" fill="red" />
  <rect x="0.0" y="40.0" width="40.0" height="40.0" fill="green" />
  <rect x="40.0" y="40.0" width="40.0" height="40.0" fill="yellow" />
</svg>
```



Co už máme

- vykreslení šachovnice v SVG
- včetně souřadnic

Co musíme udělat

- načtu konfiguraci AZ ze souboru třeba do 2D pole
- budu mít hash s barvičkama (#rrggbb formát je skoro nejjednodušší, #000000 černá, #0000ff modrá)
- vykreslím do SVG - barva podle typu elementu, text typ elementu

Barvy elementů

Nejjednoduší je pro začátek mít to v nějakém hashi ve zdrojáku:

```
colors = {
  "R1" => "#ff0000",
  "R2" => "#ff4400",
  ...
}
```

Hash je pro tyto účely ideálním řešením – snadno získám barvu pro daný element `colors[element]`

Co s variantou nedefinovaných elementů

- v posledním souboru jsou navíc neznámé typy XX a YY – vhodně vyřešte:
- a) náhodná barva `rand`, `rand(123)`, `"%02x" % 254`
- b) seznam barev pro neznámé typy
- vylepšení: načítejme barvy ze souboru! mohlo by to vypadat takto: (ale použijeme YAML - viz dále)

```
F4   ff0000
R1   00ffff
R2   00eeee
```

YAML - kamarád pro konfigurační soubory

- je trochu nuda pořád ručně načítat soubory a parsovat je, normálně se to tak nedělá – použiju standardizovaný formát souboru
- na konfigurační soubory je skvělý formát YAML (YML) – jednoduchý hash formátovaný odsazením (pozor - mezery, ne taby)

YAML - použití

- načtu standardní knihovnu `require "yaml"`
- parsování řetězce `YAML.load(File.read("config.yml"))`
- uložení dat `File.write("config.yml", data.to_yaml)`

YAML - kamarád pro konfigurační soubory

- přesuňte hash s barvami do konfiguračního souboru
- s hvězdičkou: RRGGBB, #RRGGBB i R,G,B

```
colors:
  F4: ff0000
  R1: 00ffff
  R2: 00eeee
```

Obsah

Ukládání strukturovaných dat

- často mám data ve formě struktury (kombinace hash+pole, různý stupeň vnoření)
- z různých důvodů můžu chtít data uložit na disk a pak je znovu načítat
- (zejména efektivita zpracování, případně data z externích/webových zdrojů)
- bylo by dobré mít možnost uložit a načíst rovnou celý hash
- odpověď jsou strukturované metaformáty - YAML, XML, JSON

Práce s JSON

- v Ruby je k mání knihovna – `require "json"`
- generování JSON: `hash.to_json`
- čtení JSON: `JSON[data]`

Příklad – výsledky běhu kolem rybníka

- dva soubory – `ages.csv`, `times.csv`
- chci v jednom skriptu (tasku) načíst, spárovat a uložit
- a v jiném už rovnou načíst zpracovaná data
- a vypsat tabulku výsledků včetně ročníků narození
- pozor! JSON nezná symboly, uloží se jako řetězce

Obsah

K čemu to?

- spousta informací na webu je poskytována ve strojově čitelné formě
- API – rozhraní mezi aplikacemi
- s využitím webových služeb naše možnosti exponenciálně rostou (počasí, doprava, mapy, atd atd.)
- spousta věcí se dá udělat jako *mashup* – sice nic neumím, ale umím to dát dohromady

Typy / formáty

- URL – rovnou dostanu např. obrázek po zadání správného URL
- XML – velmi obecný, ale komplikovaný formát (“vypadá jako HTML”)
- JSON – velmi jednoduchý a kompaktní formát, vyvinutý pro JS (v podstatě jen číslo, řetězec, pole, hash)

URL API – google maps

- stačí správně vymyslet URL a je to tím vyřešené
- pozor na usage limits (v produkci je nutné lokální cache...)
- QR platba:

  `http://qr-platba.cz/pro-vyvojare/restful-api/#generator-czech-image`

- Google Maps static API

Statická mapa

- Google Maps static API
- `https://developers.google.com/maps/documentation/static-maps/intro`
- API klíč – v praxi si musíte pořídit vlastní (ale nic to nestojí)
- AIzaSyDD3K0yUjHswlPn0PMSjjuuuyJdHph2JDY
- `https://maps.googleapis.com/maps/api/staticmap?center=Prague&zoom=13&size=600x300&maptype=roadmap&key=AIzaSyDD3K0yUjHswlPn0PMSjjuuuyJdHph2JDY`
- jednoduchý úkol: mapa s puntíkem ve vašem rodném městě

Jednoduchý mashup: mapa o-závodů

- ORIS API – http://oris.orientacnisporty.cz/API
- úkol: vypišme kalendář MTBO závodů v roce 2017
- `http://oris.orientacnisporty.cz/API/?format=json&method=getEventList&sport=3&datefrom=2019-01-01&dateto=2019-12-31`

Jak to dát dohromady

- stáhnu data a uložím (JSON?)
- vykreslím si mapu ČR
- přidám do ní puntíky podle potřeby
- nespletu pořadí zeměpisné šířky a délky
- nebudu se bát dlouhého a složitého url

Obsah

- vykresleme klikací mapu MTBO závodů
- použijeme mapy.cz
- chceme to ovšem trochu dotáhnout k dokonalosti – např. slučování závodů na stejném místě do jednoho bodu
- vygenerujeme mapy pro různé roky (třeba 2015 až 2019)
- když to budeme mít, jako bonus budeme vypisovat i vítězku ženské elitní kategorie (W21E)

Zdroj dat

- JSON API systému ORIS je na `http://oris.orientacnisporty.cz/API/?format=json&method=getEventList&sport=3&datefrom=2017-01-01&dateto=2017-12-31`
- API pro výsledky: `https://oris.orientacnisporty.cz/API/?format=json&method=getEventResults&eventid=2077`
- hodí se použít nějaký add-on do prohlížeče na JSON, je s tím pak míň práce
- alternativně si vyrobíme svůj hezkovypisovač s použitím `JSON.pretty_generate`

Lokální cache

- je pomalé a nešikovné stahovat pokaždé data, takže chceme lokální cache pro stažená data
- uvědomíme si, že vlastně nemusíme řešit JSON a stačí jen uložit data
- případně můžeme vytahat z dat to, co potřebujeme a uložit si to do JSONu už zpracované

Obsah

Mapa - HTML - JS - Erb

- použijeme příklad `mapa.html`
- ideální postup: nejdřív si potřebnou funkcionalitu (značka, vizitka ...) vytvořím ručně
- až pak to začnu řešit ve skriptu
- – protože když neumím JavaScript, tak je potřeba s tím zacházet trochu opatrně
- rozhodně použijme erb

Příklady JS pro to, co potřebujeme

- značky:
  `https://api.mapy.cz/view?page=markerlayer`
- s vizitkou:
  `https://api.mapy.cz/view?page=markercard`
- komplexnější vizitky:
  `https://api.mapy.cz/view?page=card`

Na co nezapomenout při zpracování dat

- budeme chtít sdružovat závody podle místa
- vzpomeneme si: pokud seskupuju cokoliv podle jednoznačného klíče, použiju hash
- získání výsledků: pomocí ID závodu `https://oris.orientacnisporty.cz/API/?format=json&method=getEventResults&eventid=2077`

A to je vše, přátelé!