Informatika pro moderní fyziky (10) složitější interaktivní dokument, správa zdrojového kódu

František HAVLŮJ

e-mail: haf@ujv.cz

ÚJV Řež
oddělení Reaktorové fyziky a podpory palivového cyklu

akademický rok 2013/2014
17. prosince 2013

Obsah

- další procvičení zpracování dat - data s vazbami
- CSS selektory
- získávání informací z webu
- LaTex a ERb – opakování

Obsah

Zadání – připomenutí

- každý den data z 1-9 detektorů (`data/*.csv`)
- detektor má svoji polohu v AZ (VR-1 Vrabec, 8x8 čtvercových pozic) – včetně data je uvedena na prvním řádku CSV souboru
- je potřeba hezky zobrazit na každý den mapu AZ a grafy signálů z detektorů
- viz `html/document.html`

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 (hash! hash!)
- vyrobit si HTML dokument, který by zobrazoval jednotlivé mapy
- doplnit interaktivitu do SVG obrázků
- zobrazovat po kliknutí do mapy správný graf

Krok číslo jedna

- z datových souborů si vyrobit takovou datovou strukturu, která mi bude říkat, pro který den a na které poloze je který soubor
- tj. radím například takovýto hash: datum => poloha => soubor

Další kroky

- dopsat do mapy detektorů text - polohu
- vyrobit zatím verzi, kde si budu moci prohlížet jednotlivé mapy (zatím bez grafů)
- zobrazit grafy detektorů - zatím všechny
- dokončit – zobrazovat grafy

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! (musí být na textu i na čtverečku)

Obsah

Jak mít uložený zdrojový kód

- to, že mám na disku hromadu souborů a ‘jen tak’ je edituju, přináší spoustu problémů
- pokud udělám chybu/něco se rozbije, nemám se jak vrátit, nemám přehled o průběhu změn
- pokud chci mít víc verzí najednou (stabilní / vývojová), nemám jak zajistit konzistenci
- není jak se vracet ke starším verzím, problémy de facto nelze reproduktovat

Práce v týmu

- pokud na projektu pracuje víc lidí, stává se z problému noční můra
- jak dát dohromady
- jak zajistit, že všichni používají stejnou verzi?
- pro dokumenty už jsou různá více či méně dobrá řešení (slučování změn v MS Office, online nástroje jako Google Documents)

Jaké jsou možnosti

- nabízí se schovávat si každý týden verzi a pokusit se v tom udržet pořádek
- dbát na správná jména složek/souborů
- za žádnou cenu nemodifikovat verzovanou složku (read-only...)
- vést záznam o změnách, všechno podrobně dokumentovat

Jak se to dělá správně

- existují takzvané version control systems – VCS, které se o řešení tohoto problému postarají za nás
- základní princip: neukládám aktuální verze souborů, ale pouze *změny*
- k čemu to je – změny je možné mezi sebou **sčítat**, což je (po lehkém zamyšlení) naprosto geniální a řeší to všechno
- pokud mám paralelní změny (dva různí lidé něco změní), není to obvykle vůbec žádný problém, pokud
- můžu rekonstruovat libovolný bod v historii

Řešení konfliktů – merging

- pokud dva uživatelé udělají změnu zároveň, co se stane?
- když konflikt nenastal na stejném řádku stejného souboru, tak je to v pořádku
- v opačném případě to prostě musí vyřešit uživatel ručně (ale není to častá situace)
- spojení změn se ale vždycky musí dělat lokálně (na straně uživatele), aby se zamezilo sémantickému konfliktu

Paralelní větve – branches

- často se hodí mít víc verzí najednou
- jednak různé stupně experimentálnosti (stabilní, testovací, vývojová)
- jednak pokud chci přidat větší feature, tak to nechci míchat s hlavní větví vývoje
- VCS to většinou umí řešit – strom vývoje nemusí být lineární, ale může se větvit

Přehled VCS

- CVS – první standardní VCS, dnes již zcela překonaný
- SVN – moderní centralizovaný VCS, donedávna nejpoužívanější řešení
- Git – dnešní de facto standard, pokročilý distribuovaný VCS s geniální podporou branches
- Mercurial, Bazaar – méně obvyklé alternativy Gitu

Centralizovaný vs. distribuovaný VCS

- centralizovaný: mám jeden centrální repozitář, do kterého ukládají změny všichni uživatelé
- distribuovaný: každý uživatel má celý repozitář u sebe
- výhody – lokální commity (bez spojení se serverem!), lokální branche, bezpečnost (vysoká redundance)
- nevýhody – někomu to dělá problém pochopit a přijmout

Kdy je dobré používat VCS

- vlastně vždycky, pokud pracuju s plaintextovými soubory (tj. zejména zdrojové kódy, skripty)
- využití pro binární data (což se týká de facto i wordovského dokumentu...) je velice omezené
- – další důvod používat LaTeX! tam je samozřejmě všechno v pořádku
- výhody verzování (oproti tomu to “jen někde mít”) jsou nedozírné
- za sebe můžu říct, že asi úplně všechnu svoji práci mám v nějakém repo

Ukázka a příklad veřejných repozitářů

- http://github.com
- vhodné zejména pro open source projekty, ale umožňuje i soukromé repo (za poplatek)
- velmi hezké GUI (přehledy změn, grafy větvení, issue tracker, komunikace mezi vývojáři)
- samozřejmě není problém si nainstalovat repozitář (např. `gitolite`) někde u sebe na serveru

A to je vše, přátelé!