REST API pro sledování nasazení železničních vozidel

https://github.com/55p/train-locator

http://train-locator.appspot.com/

Pavel Dvořák

12. 5. 2014

Motivace:

Oběhy železničních vozidel jsou pravidelné, respektive definované jako pravidelné. Tohoto faktu využívají fotografové vozidel (galerie např. www.zelpage.cz, www.trainweb.cz aj., diskuze zejména www.k-report.net), kteří na základě pozorování dokáží odhadnout, na kterých vlacích bude dané vozidlo jezdit v příštích dnech. V oběžích je např. dáno, že souprava (lokomotiva, motorová jednotka, …), která například jede v pondělí na vlaku 783, přejde na vlak 792, poté na vlaky 789, 786 atd. Z toho lze také předvídat, že bude v úterý jezdit na vlacích 781, 794, 787 atd. Na základě pozorování z pondělka tak mohou lidé přizpůsobit svůj program na úterý, aby vyfotili nebo naopak věděli, že nemá smysl daný vlak fotit.

Volnou inspirací je systém používaný na adrese http://55p.8u.cz/sledovani/, který byl vytvořen v jazyce PHP. Možnosti tohoto systému jsou ovšem omezené, například definice oběhů je možná pouze ve zdrojovém kódu.

Popis entit:

Vlaky jezdí v tzv. *turnusových dnech (TD)*, několik TD tvoří tzv. *turnusovou skupinu (TS)*. V jedné *sledovací tabulce* může být více příbuzných TS, např. na základě stejných vozidel nebo jedné lokality výskytu. Kardinality všech vztahů jsou definováné **0 .. n – 1**.

Dělení aplikace:

Zdroje aplikace lze rozdělit na část **definiční** a **datovou.** Definiční část obsahuje zdroje pro CRUD operace sledovaček, TS, TD a vlaků. Datová část zahrnuje vkládání a přístup k informacím o vložených záznamech.

Podobně existují i dvě klientské verze: Jedna pro definice oběhů a druhá pro zobrazování uložených dat.

Míchání jazyků

Pro všechny proměnné jsou použity anglické názvy, vyjma označení druhu lokomotivy na vlaku. Odborné termíny označení „vlaková“, „přípřež“, „postrk“ a „řídicí vůz“ byly ponechány česky.

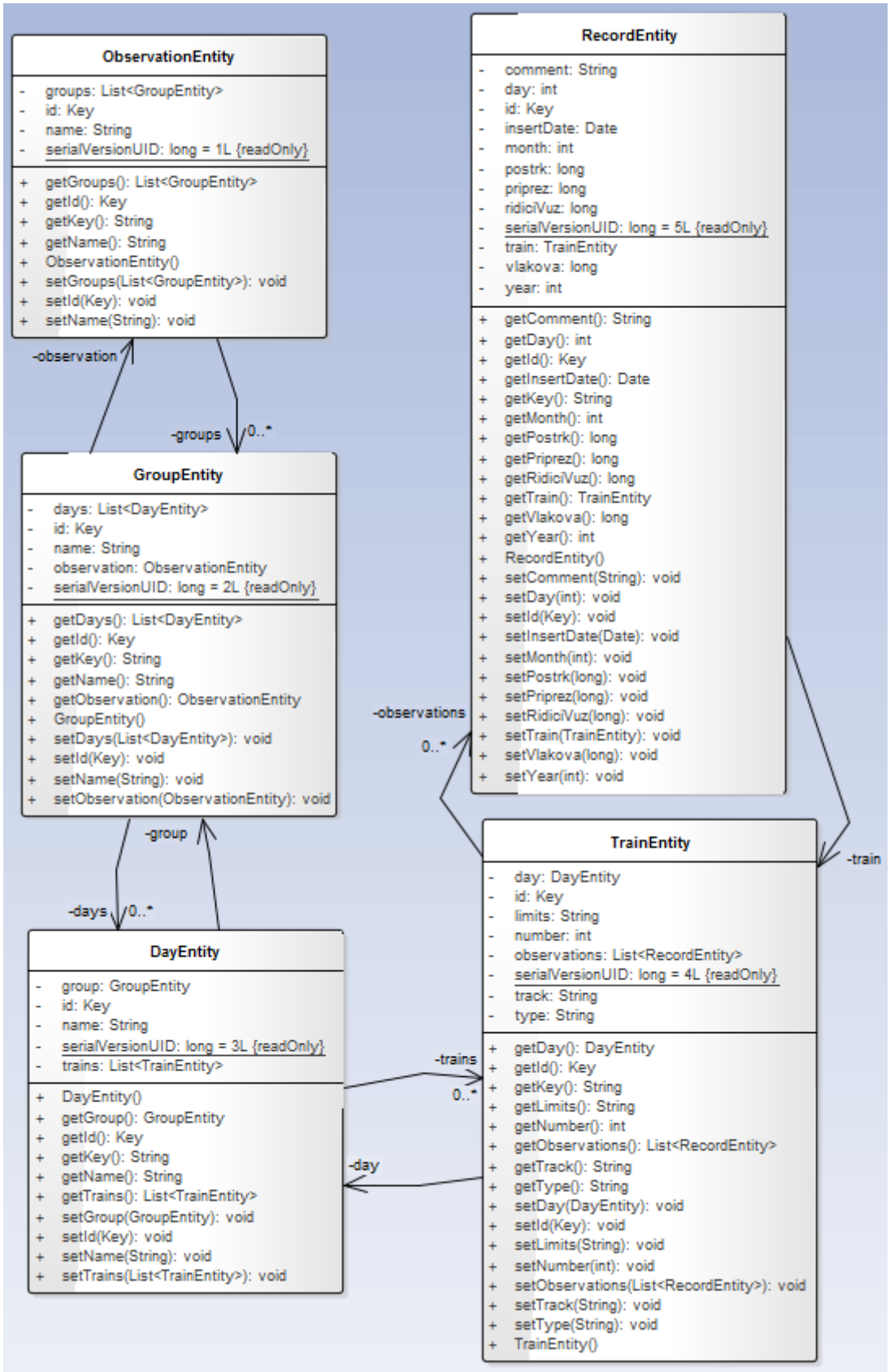
Filozofie aplikace

Jednotlivé záznamy nejsou mazány nebo upravovány, lze je pouze přepsat později vloženým. Zobrazovací aplikace pak mají zobrazit nejnovější záznam. Starší záznamy mohou být zobrazeny v administraci tabulky pro kontrolu.

Grafické rozhraní

Všechny zdroje podporují CORS, tedy Cross-Origin Resource Sharing. Díky tomu je možné vytvořit AJAXového klienta. Veškerá komunikace s REST API probíhá ve formátu JSON.

Entity v systému:



Zdroje – část definiční

/
GET: Vrátí odkazy na zdroje

/observation
GET: Vrátí sledovačky

POST: Nová sledovačka

/observation/{idSledovačky}
GET: Vrátí sledovačku

PUT: Úprava sledovačky

DELETE: Odstraní sledovačku

/observation/{idSledovačky}/group
GET: vrátí skupiny ve sledovačce

/observation/{idSledovačky}/data
GET: vrátí skupiny, dny a vlaky ve sledovačce

/group
GET: Vrátí všechny TS

POST: Vytvoří novou TS

/group/{idTS}
GET: Vrátí údaje o turnusové skupině

DELETE: Odstraní TS

PUT: Úprava TS

/group/{idTS}/day
GET: Vrátí TD v TS

/day
GET: Vrátí všechny TD

POST: Vytvoří nový TD

/day/{idTD}
GET: Vrátí údaje o TD

DELETE: Odstraní TD

PUT: Úprava TD

/day/{idTD}/train
GET: Vrátí vlaky v TD.

/day/{idTD}/data
GET: Vrátí sledovačku, její skupiny, dny a vlaky, ve které je den s idTD

/train

GET: Vrátí všechny vlaky.

POST: Vytvoří nový vlak.

/train/multiple

POST: Vytvoří více vlaků, přijímá list.

/train/{idVlaku}

GET: Vrátí vlak.

PUT: Upraví vlak.

DELETE: Smaže vlak.

/train/{idVlaku}/data

GET: Vrátí sledovačku, její skupiny, dny a vlaky, ve které je vlak s idVlaku.

/train/find/{čísloVlaku}

GET: Vyhledá vlaky se zadaným číslem.

Zdroje – část datová

/record/
POST: Vloží nový záznam, vlak je třeba specifikovat v datech.

/record/multiple
POST: Vloží více nových záznamů, přijímá list.

/train/{idVlaku}/add
POST: Vloží nový záznam pro zvolený vlak.

/record/observation/{idSledovačky}/
GET: Vrátí pozorování ve sledovačce pro dnešek.

/record/observation/{idSledovačky}/{year}/{month}
GET: Vrátí pozorování ve sledovačce pro zadaný měsíc.

/record/observation/{idSledovačky}/{year}/{month}/{day}
GET: Vrátí pozorování ve sledovačce pro zadané datum.

/record/group/{idTS}/
GET: Vrátí pozorování v TS pro dnešek.

/record/group/{idTS}/{year}/{month}
GET: Vrátí pozorování v TS pro zadaný měsíc.

/record/group/{idTS}/{year}/{month}/{day}
GET: Vrátí pozorování v TS pro zadané datum.

/record/day/{idTD}/
GET: Vrátí pozorování v TD pro dnešek.

/record/day/{idTD}/{year}/{month}
GET: Vrátí pozorování v TD pro zadaný měsíc.

/record/day/{idTD}/{year}/{month}/{day}
GET: Vrátí pozorování v TD pro zadané datum.

/record/train/{idVlaku}/
GET: Vrátí pozorování zadaného vlaku pro dnešek.

/record/train/{idVlaku}/{year}/{month}
GET: Vrátí pozorování zadaného vlaku pro zadaný měsíc.

/record/train/{idVlaku}/{year}/{month}/{day}
GET: Vrátí pozorování zadaného vlaku pro zadané datum.

JSON požadavky a odpovědi

Veškeré typy, které nejsou uvedeny, jsou řetězce. Při vytváření nebo úpravě zdrojů není nutné vyplňovat ID, toto je buď vygenerováno automaticky, nebo předáno v adrese.

Sledovačka:

id	ID sledovačky
name	jméno sledovačky

Turnusová skupina:

id	ID skupiny
name	jméno skupiny
observationID	ID sledovačky, v níž je tato turnusová skupina

Turnusový den:

id	ID dne
groupID	ID turnusové skupiny, v níž je tento turnusový den
name	označení dne

Vlak:

id	ID vlaku
dayId	ID turnusového dne
limits	omezení jízdy vlaku
number	číslo vlaku **[číslo]**
track	trasa vlaku
type	typ vlaku (Os/R/Sp/…)

Záznam:

id	ID pozorování
day	den pozorování **[číslo: 1-31]**
month	měsíc pozorování **[číslo: 1-12]**
year	rok pozorování **[číslo]**
trainNumber	číslo pozorovaného vlaku **[číslo]**
trainId	ID pozorovaného vlaku
vlakova	vlaková lok. **[číslo: 100000-999999]**
ridiciVuz	řídicí vůz vlaku **[číslo: 100000-999999]**
priprez	přípřežní lok. **[číslo: 100000-999999]**
postrk	postrková lok. **[číslo: 100000-999999]**
comment	poznámka k záznamu
insertDate	datum vložení záznamu **[timestamp]**

Při vložení záznamu není nutné vyplnit všechna pole:

- Pokud není vyplněno datum, považuje se datum odeslání za aktuální den.
- Pokud není vyplněn rok, považuje se rok za letošní
- Můžete zadat buď číslo vlaku, nebo ID vlaku. Pokud je nalezeno více vlaků daného čísla, budete vyzváni k upřesnění pomocí ID vlaku.
- Jednotlivá pole *vlakova*, *ridiciVuz*, *priprez* a *postrk* vyplňte, jen když jsou pro daný záznam potřeba. Pokud nejsou zadána, předpokládá se nulová hodnota, tj. nepřítomnost na vlaku.
- Datum vložení záznamu bude nastaveno automaticky při přijetí záznamu.
- Pokud není zadána poznámka, předpokládá se prázdný text.

Klientská část

Klientskou část tvoří AJAXový klient kompletně vytvořený v JavaScriptu. Ten se skládá z několika souborů:

- Listener.js, objekt Listener – jeho funkce slouží jako controllery na akce, obsahuje funkce pro práci s historií, onclick eventy, …
- Memory.js, objekt Memory – paměť používaná za běhu aplikace.
- Server.js, objekt Server – třída sloužící pro načtení dat, která nejsou dostupná v třídě Memory. Odpovědi serveru ukládá též do lokální paměti.
- Storage.js, objekt Storage – obalení API WebStorage.
- UI.js, objekt UI – funkce pro vypsání uživatelského rozhraní
- Object.js, objekty Observation, Group, Day, Train, Record a RecordMap. Třída RecordMap obsahuje pouze proměnné posílané v JSONu na server při vložení nového záznamu. Ostatní třídy slouží pro reprezentaci dat o oběhu a jejich instance jsou navzájem propojené.
- Canvas.js, objekt Canvas – funkce pro kreslení jezdící lokomotivy do elementu Canvas.
- Notify.js, objekt Notify – zobrazování notifikací, pokud nejsou povoleny, zobrazuje chyby alertem.

Ke klientské části patří ještě tři soubory: style.css s kaskádovými styly, prototype.js zajišťující doplnění funkcí do stávajících objektů a index.html obsahující kostru stránky a mikrodata.

Popis funkčnosti

Popis funkčnosti klientské části lze shrnout do několika bodů, které platí napříč celou aplikací:

- Třídy UI, Listener a Server pracují s instancí Memory.
- Třída Listener volá metody třídy Server a UI.
- Pokud jsou dostupná data v třídě Memory, zavolá Listener přímo UI. Jinak zavolá server a funkci třídy UI předá jako referenci. Tento callback je zavolán až po načtení a zpracování dat.
- Objekty Observation, Group, Day, Train, Record jsou obousměrně propojené, čehož se hojně využívá při výpisu tabulek sledování
- Aby byl i po stisku klávesy F5 načten předchozí stav aplikace, jsou potřebné údaje ukládány do URL. V URL jsou navzájem odděleny rozděleny znakem „!“.
- Všichni posluchači navěšení na události jsou ze třídy Listener (vyjma zpracování XHR, ty jsou ze třídy Server)

Další rozvoj

Aby mohla být aplikace považována za plnohodnotnou, je nutné doplnit mnoho částí rozhraní. Jedná se například o definici časové platnosti sledovaček, neboť oběhy se zpravidla mění každý rok při změně jízdního řádu. Trasy vlaků by měly být zadávány ve strojově zpracovatelném formátu. Stejná situace je i u omezení jízdy vlaků, neboť není výjimkou, že o víkendu platí úplně jiné oběhy, než v pracovní dny. Díky tomu by v daný den bylo možné zobrazit jen příslušné vlaky nebo zakázat vložení záznamu k vlaku, který v daný den nejede.

Další možnost rozvoje je výpočet předpokladů. Stroje zpravidla přecházejí do následujících TD, tj. z prvního do druhého apod., nalezneme ovšem nemnoho výjimek. Zde je situace s obtížnou definicí vstupních dat podobná jako u omezení jízdy:

Definice předpokladů a omezení jízdy vyžaduje velmi důkladnou přípravu, omezení mohou být složitá s mnoha vnitřními závislostmi. Příklad skutečného omezení jízdy vlaku v oběhu může být následující:

```
jede v úterý až pátek, nejede 22.IV., 1., 2., 8., 9.V., 1.VII.
- 29.VIII., 28., 29.X. a 18.XI.
```

nebo

```
jede v pondělí, 22.IV., 2., 9.V., 29.X., 18.XI., nejede
21.IV., 30.VI. až 25.VIII. a 17.XI.
```

Lze také zvažovat možnost definice pořadí turnusových skupin a turnusových dní ve sledovačce, zde ale v naprosté většina případů stačí lexikografické řazení. Vlaky lze (v případě strojového zadávání trasy) řadit dle času odjezdu z výchozí stanice.