

Rapport

Aleksandar Mlinar

aleksandar.sasha.mlinar@gmail.com

ISGB13 - Portabla Format - Examination

I min examinationsuppgift har jag valt att bygga en tjänst (ett fiktivt företag som heter Internet Song Database) där man kan söka efter låtar på internet och få grundinfo om dessa. Musik är någonting jag älskar, jag spelar själv gitarr och det är ett ämne som man alltid kan forska mer i. Så valet mellan musikapp och en app om hur många grillplatser finns i Umeå kommun och hur dom är utspridda(hittade detta på opnadata.se) var ganska enkel. Trots att det låter lockande att göra en karta över grillplatser i och för sig.

Ja forskade väldigt mycket kring ämnet och hittade några alternativ som jag använde i musiktjänsten jag byggde. Jag har använt mig av JavaScript, HTML och CSS. Har verkligen försökt att hålla HTML-koden till minimum och manipulera DOM-en så mycket så möjligt ifrån JavaScript. Har använt mig av strängar, objekt, variabler etc. Applikationen jag byggde är en tjänst där man söker efter låtar och får låtnamn, artist och en bild på albumet låten finns på (json). Detta presenteras i ett rutnät. Klickar man på en av låtarna/rutorna visas det en ruta där man ser allt ovannämnda infot och på ett representativt sätt kan man spela upp en liten del av låten. Till det har jag byggt också en tjänst där man ser låttexter för låten man har valt. Det datasetet 'fetchar' jag ifrån chartlyrics.com(xml). Så man har sin favoritlåt, kan lyssna på en liten del av den och läsa låttexten.

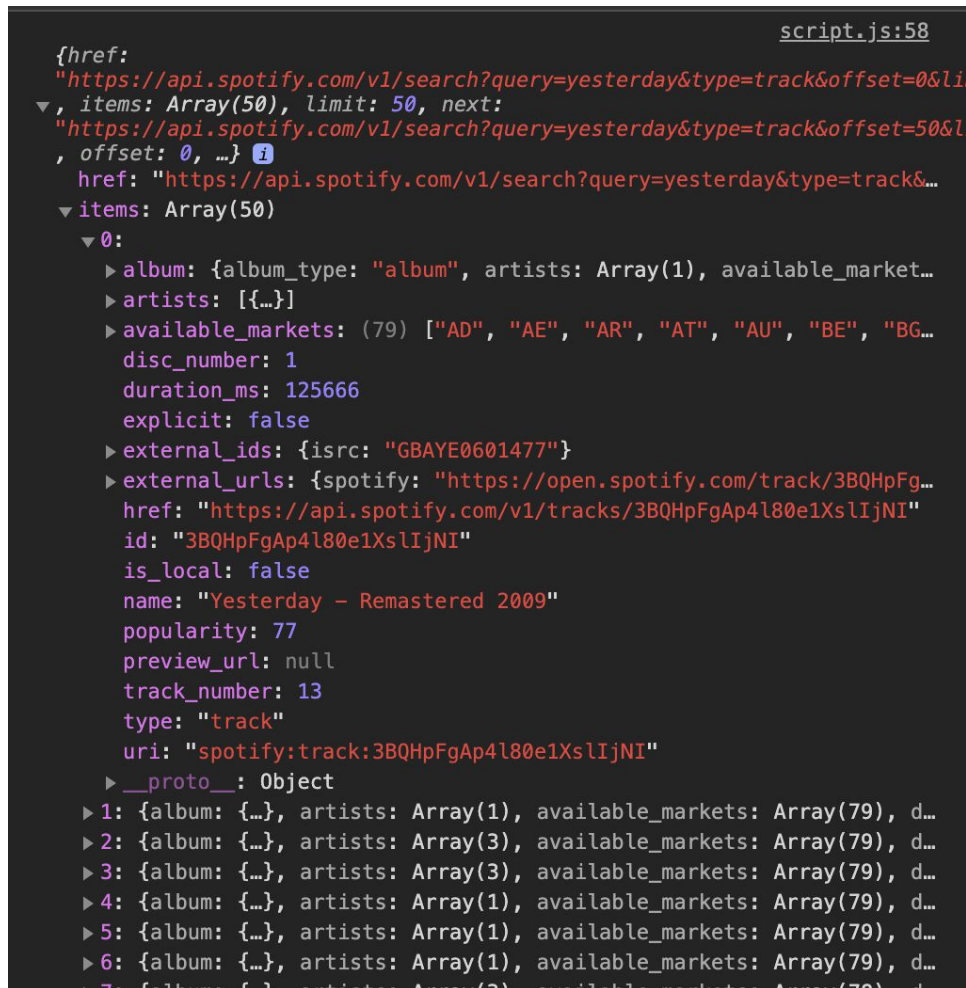
I början valde jag att arbeta med last.fms dataset. Jag visste att jag ville göra något sort av musik sökningsfunktion men visste inte från början vad exakt. Last.fm API verkade vara lätt att förstå och det liknade det som vi gjorde under andra föreläsningen på kursen så man kände sig hemma när man jobbade med deras applikationprogrammeringsgränssnittet. Jag ångrade mig dock efter ett tag eftersom man kunde inte fetch-a bilder till låtar (omslagsbilder för att kunna presentera data på ett attraktivt sätt). Ansåg snabbt att detta var ett stort estetiskt problem och jag valde Spotify istället. Som tur det så kunde jag återanvända koden och bygga på den befintliga jag skrev från början.

1. Spotify API och datasetet.

Till skillnad från Last.fms API var Spotifys mycket mer invecklad och detaljerad. Största problemet jag stötte var användning av tokens vilka går ut och man måste hämta nya efter det. Man kunde skriva kod för att göra en automatisk refresh men

detta var lite krångligt och något jag inte kunde ägna tid åt med tanke på att jag hade andra uppgifter att ta itu med. Det är det enda problemet jag inte kunde lösa inom denna tidsram och därför behöver man lägga in kod på nytt ifall man vill köra applikationen. Koden kan jag generera via min app på Spotifys hemsida.

Datasetet är väldigt lätt att arbeta med. Data är otroligt bra strukturerad och det var lätt att hitta info i objekten. Till exempel när man söker efter Beatles låt Yesterday får man denna struktur, se bilden nedan.



```
script.js:58
{href:
  "https://api.spotify.com/v1/search?query=yesterday&type=track&offset=0&li
  ", items: Array(50), limit: 50, next:
    "https://api.spotify.com/v1/search?query=yesterday&type=track&offset=50&li
    , offset: 0, ...}
  href: "https://api.spotify.com/v1/search?query=yesterday&type=track&...
  items: Array(50)
    0:
      album: {album_type: "album", artists: Array(1), available_market...
      artists: [{...}]
      available_markets: (79) ["AD", "AE", "AR", "AT", "AU", "BE", "BG...
      disc_number: 1
      duration_ms: 125666
      explicit: false
      external_ids: {isrc: "GBAYE0601477"}
      external_urls: {spotify: "https://open.spotify.com/track/3BQHfFg...
      href: "https://api.spotify.com/v1/tracks/3BQHfFgAp4l80e1XsIjNI"
      id: "3BQHfFgAp4l80e1XsIjNI"
      is_local: false
      name: "Yesterday - Remastered 2009"
      popularity: 77
      preview_url: null
      track_number: 13
      type: "track"
      uri: "spotify:track:3BQHfFgAp4l80e1XsIjNI"
      __proto__: Object
    1: {album: {...}, artists: Array(1), available_markets: Array(79), d...
    2: {album: {...}, artists: Array(3), available_markets: Array(79), d...
    3: {album: {...}, artists: Array(3), available_markets: Array(79), d...
    4: {album: {...}, artists: Array(1), available_markets: Array(79), d...
    5: {album: {...}, artists: Array(1), available_markets: Array(79), d...
    6: {album: {...}, artists: Array(1), available_markets: Array(79), d...
    7: {album: {...}, artists: Array(3), available_markets: Array(79), d...
```

Bild 1.1

När man söker igenom datasetet så är detta (bilden ovan) man stöter på först i konsolen. Man presenteras med objekt som sen har en array av 'items' och där har man lagrat resultat från sökningen. Inga avvikelser från standarden överhuvudtaget.

2. Chartlyrics.com API och datasetet.

Chartlyrics.com hade en väldigt enkelt API och datasetet. Anledningen varför jag valde detta set är att jag behövde bara använda låttexten och låtnamnet från datasetet. Efter att jag forskade en del så visade sig att detta var enklaste sättet att få fram data utan att behöva skapa applikation på deras hemsida för att i sin tur skaffa API-nyckel som man kunde använda. Sen var det bara att parse data från sträng till DOM dokumentet. Själva strukturen var inte ovanlig och jag fick lätt åtkomst till data. Man måste nog ha i åtanke att jag använde bara två stycken metadata att tyda/läsa ifrån. Dock så är väntetider på att data ska hämtas och läsas in är, måste jag säga, oacceptabelt varierande. Man får ibland data direkt och ibland efter 13-14 sekunder. Skulle nog inte kunna använda det för något betydligt större projekt.

3. JavaScript Object Notation och Extensible Markup Language.

En av dom största skillnaderna och självklart fördelen för JSON är att man använder mindre ord och är definitivt snabbare än XML vad jag har upplevt. JSON är mycket mer lättläst och har inte samma hierarki som XML har. Däremot så har XML stöd för namnrymder som användes som elementtyper och attributnamn vilket JSON inte har. Båda två är väldigt populära och jag tror att allt handlar om på vilket sätt man ska använda data är det som avgör vad man ska välja. För mig hade det nog varit svårt att hämta låtinfo med XML eftersom XML inte stöder arrayer. JSON däremot representerar sin data genom objekt och baseras på JavaScript så man kunde i väldigt stor utsträckning presentera data som arrayer, objekt osv. Väldigt mycket att bygga på som är lättåtkomligt enligt mig. En 'path' till Yesterday (som jag nämnde in början av rapporten) såg såhär ut: **musicData.items[].name**. Sen var det bara att bygga på, man kunde ju själv logiskt ana vad man skulle kunna skriva i fortsättningen för att kunna hämta ut album cover till exempel.