Assignment 13 (HF, major subject)

Due: Wed 03.02.2016; 12:00h (1 week)

Goals

After doing these exercises,

- You can handle AJAX requests declaratively
- Databinding is no longer a problem, in particular computed binding
- You'll know more about HbbTV

Task 1: HbbTV 2 Difficulty: Easy

Read this article: HbbTV 2.0 spec released by Chris Dziadul

Briefly answer the following questions:

a) What are the most anticipated features of HbbTV 2.0?
b) Is HbbTV capable of UltraHD? Which codec is mentioned for in this context?
c) Name 2 implications of "synchronization between TV, tablets and smartphones"
d) Is HbbTV 2.0 downward-compatible with older versions of HbbTV?

Put your solution into the folder 'task1'.

Task 2: Web Radio Browser Element Difficulty: Intermediate

Web radio is still alive and easy to use. Services like dirble.com allow you to browse stations and listen to your favorites. Also, they provide an API that allows developers to request a list of stations.

This API allows us to query a list of radios stations, but you need to get an API token first.

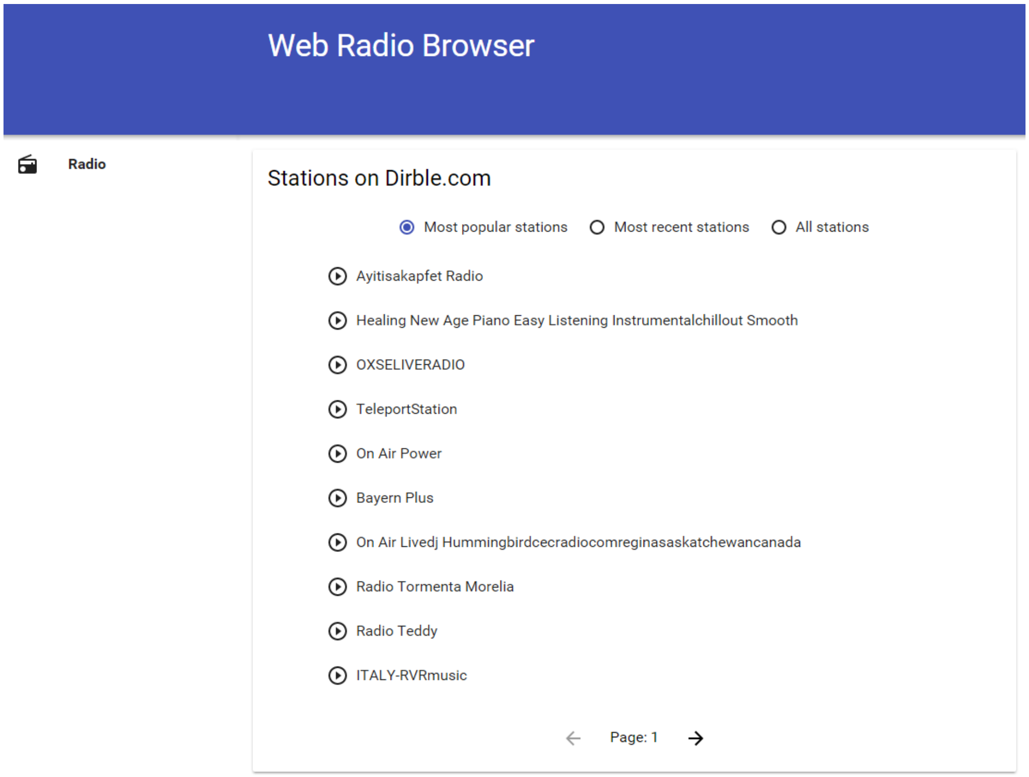
1. Sign up for dirble.com: https://dirble.com/users/sign_in
2. Create an API token: https://dirble.com/users/apikeys
3. Look at the documentation: https://dirble.com/api-doc

On GitHub, you'll find a working custom element <`webradio-browser`>. We suggest you first take a look at the entire code / mark-up that it contains.

Try to comprehend these points (you don't need to submit the answers, this is just for you):

- What does the `makeAjaxParams(…)` function actually do and where is it called?
- In `showPreviousPage()` and `showNextPage()` only the page number is updated. In other words, the code does not specifically mention that a new request to the API should be launched. Why does the element fetch a new list of stations nonetheless through AJAX?

- The `playRadio()` function uses the HTML5 `<audio>` element. Make sure you understand what's happening there.



Web Radio Browser

Radio

Stations on Dirble.com

● Most popular stations ○ Most recent stations ○ All stations

- Ayitisakapfet Radio
- Healing New Age Piano Easy Listening Instrumentalchillout Smooth
- OXSELIVERADIO
- TeleportStation
- On Air Power
- Bayern Plus
- On Air Livedj Hummingbirdcecradiocomreginasaskatchewancanada
- Radio Tormenta Morelia
- Radio Teddy
- ITALY-RVRmusic

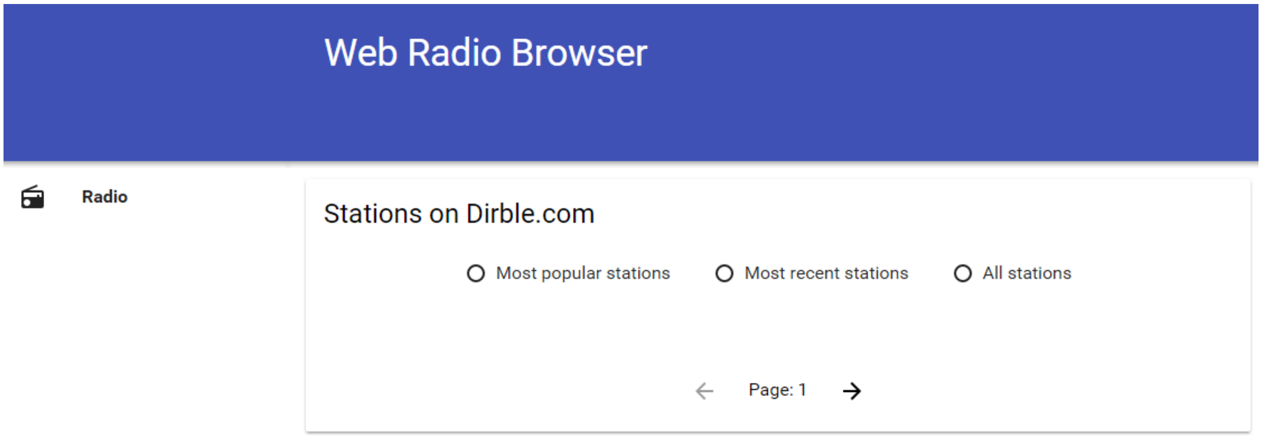
← Page: 1 →

In the screenshot you see three radio buttons for the most popular, most recent and all stations. So far, they don't provide much functionality – this is going to be your part. Our goal is to show a different list of stations depending on the selected radio button.

This is pretty easy, if you do this:

- Use databinding for the "`selected`" attribute of the radio group.
- Bind the value of the selected attribute to the "`url`" attribute of the iron-ajax element.

You'll notice that at first, no stations are displayed:



Web Radio Browser

Radio

Stations on Dirble.com

○ Most popular stations ○ Most recent stations ○ All stations

← Page: 1 →

You can solve this in XX ways:

- Option A: make the binding a property of the webradio-browser element. Then you can add a default value
- Option B: use the `ready()` lifecycle callback function. Then you can set an initial value for your bound variable there (see tutorial 13, slide 13).

That's it!

Good to know: A lot of stations won't play any music. We found that these work quite reliably:

- □ Ayitisakapfet Radio
- □ OXSELIVERADIO
- □ ITALY-RVRmusic
- □ Fresh Radio Leicester
- □ RFM Luxembourg
- □ rockSatelite-MadridONE
- □ 24/7 MJ Radio (our definite favorite)
- □ EMFA Radio

If you want to do more, you can try any of these ideas:

- Use a computed property / binding for the url attribute in the iron-ajax.
- Add a paper-input element that allows filtering the currently displayed stations
- Find another Web-Radio API on the Web and adjust the <webradio-browser> element so that it works with their API. We found these (but haven't tried them, yet):
  - http://www.streamfinder.com/internet-radio-api/
  - http://developer.iheart.com/Api/login
  - http://www.npr.org/api/stationFinder.php

Put your solution containing your bower.json, html and css files into the folder 'task2'. **Please do not submit the bower_components folder!**

Task 3: Send us your questions

Send the questions you would like to discuss in the final tutorial to tobias.seitz@ifi.lmu.de by Sunday, 31.01.2016.

Submission

Put the solution to all tasks into a ZIP folder. Please turn in your solution via UniWorX. You can form groups of up to three people. After the submission deadline, push your solution to our GitHub repository.