Viewing Data with XForms

February 8, 2019

Steven Pemberton (/authors/steven-pemberton/)

Steven Pemberton continues his series on XForms with an explanation of how to view data in XForms

Introduction
I have a dataset of performances by a University choir stretching over 65 years or more. It is just a long list of concerts:

```
<concerts>
  <concert>...</concert>
  <concert>...</concert>
  <concert>...</concert>
   ...
</concerts>
```

A typical concert entry looks like this:

```
<concert>
   <year>1970</year>
   <month>5</month>
   <program>Duruflé – Requiem</program>
   <program>Mozart – Krönungsmesse</program>
   <where>Augustinuskerk te Amsterdam</where>
   <where>De Doelen te Rotterdam</where>
   <with>VU-orkest</with>
   <event>LP-opname: Mozart</event>
</concert>
```

I would like to be able to browse this data, but also easily answer questions like "How often have they performed something by Stravinsky?", "How often have they performed in the Concertgebouw", "What did they perform in 1960?", and so on.

Browsing

First to browse. We load the data:

```
<instance src="concerts.xml"/>
```

and then display it, which we'll do as a sort of table, one row per concert:

```
<group>
   <label>Concerts</label>
   <repeat ref="concert">
      ...
   </repeat>
</group>
```

For each concert, each group of entries (*date*, *program*, *where*, *with*, and *event*) will be displayed as a column by making the CSS `display` property of each group `inline-block`, so that the groups are displayed next to each other:

```
<group class="concert">
   <output class="when" value="concat(year, '-', month)"/>
   <group class="program">
      <repeat ref="program"><output class="line" ref="."/></repeat>
   </group>
   <group class="where">
      <repeat ref="where"><output class="line" ref="."/></repeat>
   </group>
   <group class="with">
      <repeat ref="with"><output class="line" ref="."/></repeat>
   </group>
   <group class="event">
      <repeat ref="event"><output class="line" ref="."/></repeat>
   </group>
</group>
```

Note that this doesn't require the sub-elements of the concerts to be in this order, or even adjacent; it just selects all sub-elements called `program` (for example) within a concert element, and displays them together.

Adding a row of titles above this using the same CSS class for the header titles ensures that they line up:

```
<group class="header">
   <output class="when" value="'when'"/>
   <output class="program" value="'what'"/>
   <output class="where" value="'where'"/>
   <output class="with" value="'with'"/>
   <output class="event" value="'why'"/>
</group>
```

which, with some suitable CSS, looks like this:

Concerts

when	what	where	with	why
1955-10	Hassler – Cantate Domino	Westerkerk te Amsterdam		75e Dies Natalis VU
1956-onb	Brahms – Alt-Rhapsodie	Onbekend		
1956-10	Onbekend	Amsterdam Rotterdam		
1957-2	Onbekend	Onbekend		
1957-5	Buxtehude – Cantate Distler – Christ, der Du bist der helle Tag	Raphaëlpleinkerk te Amsterdam		
1957-10	Onbekend	Concertgebouw, kleine zaal te Amsterdam		t.g.v. 77e Dies Natalis Studentencorps VU
1957-11	Buxtehude – Alles, was ihr tut mit Worten oder Werken J. S. Bach – Gott soll allein mein Herze haben Mozart – Ave Verum Corpus Distler – Christ, der Du bist der helle Tag Distler – Lobe den Herren Mendelssohn-Bartholdy – Hymne	Marcanti te Amsterdam	Bachorkest	t.g.v. VU-concertavond
1958-10	... – Cantate	Amsterdam Leeuwarden		
1958-11	Onbekend	Marcanti te Amsterdam	Bachorkest	t.g.v. VU-concertavond
1959-7	Onbekend	Tivoli te Utrecht		t.g.v. VU dagen

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/dataview/concerts1.xhtml)

We may as well fancy up the heading a little bit, and replace:

```
<label>Concerts</label>
```

with

```
<label>Concerts, <output value="min(concert/year)"/> - <output value="max(concert/year)"/>
</label>
```

Answering Questions

Nothing fancy, we'll just do a search-machine-like search on the data.

We create an instance for the search string:

```
<instance id="search">
   <data xmlns=""><q/></data>
</instance>
```

and an input control for it:

```
<input incremental="true" ref="instance('search')/q">
   <label>Search</label>
</input>
```

That's XForms 1.1. The newer XForms 2 allows you to say:

```
<input incremental="true" ref="instance('search')/q" label="Search"/>
```

The `incremental` attribute means that the value will be updated each time the input changes, meaning the search is done for every change.

Now the only other thing is to restrict the displayed concerts to those that match the search string.

Whenever you have a sequence of items, such as with `ref="concert"` above, you can select a subset of them using a filter: `ref="concert[`*condition*`]"`, selecting only those concerts that match the condition. This is just standard XPath, since the content of the `ref` attribute is just an XPath expression.

If we want only the concerts from 1975, we can write:

```
concert[year=1975]
```

If we want only the concerts that contain a piece composed by Bach, we write:

```
concert[contains(piece, 'Bach')]
```

If we want the concerts where *any* field contains "Amsterdam", we write

```
concert[contains(*, 'Amsterdam')]
```

In fact we can even say:

```
concert[contains(., 'Amsterdam')]
```

which means "any concert that contains the string "Amsterdam" anywhere (the "." means "self").

Finally if we want the concerts that contain the search string, we write

```
concert[contains(., instance('search')/q)]
```

So we replace:

```
<repeat ref="concert">
```

with that:

```
<repeat ref="concert[contains(., instance('search')/q)]]">
```

So this says "repeat over the concerts that contain the search string".

Here is the result. You can try it – type a year, or a composer, or whatever:

Concerts, 1955 - 2022
Search

when	what	where	with	why
1955-10	Hassler – Cantate Domino	Westerkerk te Amsterdam		75e Dies Natalis VU
1956-onb	Brahms – Alt-Rhapsodie	Onbekend		
1956-10	Onbekend	Amsterdam Rotterdam		
1957-2	Onbekend	Onbekend		
1957-5	Buxtehude – Cantate Distler – Christ, der Du bist der helle Tag	Raphaëlpleinkerk te Amsterdam		
1957-10	Onbekend	Concertgebouw, kleine zaal te Amsterdam		t.g.v. 77e Dies Natalis Studentencorps VU
1957-11	Buxtehude – Alles, was ihr tut mit Worten oder Werken J. S. Bach – Gott soll allein mein Herze haben Mozart – Ave Verum Corpus Distler – Christ, der Du bist der helle Tag Distler – Lobe den Herren Mendelssohn-Bartholdy – Hymne	Marcanti te Amsterdam	Bachorkest	t.g.v. VU-concertavond
1958-10	... – Cantate	Amsterdam		

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/dataview/concerts2.xhtml)

You may have noticed that the search string has to match exactly with the content. Let's make it so that the search is case-insensitive.

The function `lower-case` returns the lower-case version of its parameter, so that if `q` is `Mozart`, then `lower-case(q)` is `mozart`. (The `lower-case` function is from XForms 2. Previous versions use `translate(q, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz')` to achieve similar effect).

So in the repeat over the concerts, we replace

```
contains(., instance('search')/q)
```

with

```
contains(lower-case(.), lower-case(instance('search')/q))
```

which checks if a lower-case version of the element content contains the lower-case version of the search string.

Here it is:

Concerts, 1955 - 2022
Search (case-insensitive)

when	what	where	with	why
1955-10	Hassler – Cantate Domino	Westerkerk te Amsterdam		75e Dies Natalis VU
1956-onb	Brahms – Alt-Rhapsodie	Onbekend		
1956-10	Onbekend	Amsterdam Rotterdam		
1957-2	Onbekend	Onbekend		
1957-5	Buxtehude – Cantate Distler – Christ, der Du bist der helle Tag	Raphaëlpleinkerk te Amsterdam		
1957-10	Onbekend	Concertgebouw, kleine zaal te Amsterdam		t.g.v. 77e Dies Natalis Studentencorps VU
1957-11	Buxtehude – Alles, was ihr tut mit Worten oder Werken J. S. Bach – Gott soll allein mein Herze haben Mozart – Ave Verum Corpus Distler – Christ, der Du bist der helle Tag Distler – Lobe den	Marcanti te Amsterdam	Bachorkest	t.g.v. VU-concertavond

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/dataview/concerts3.xhtml)

There you have it. A useful application; about 35 lines of XForms.

Making it More General

Actually I have lots of similar files: the record of all computers I have owned, my bank accounts, the list of all talks I have given. These all have the same properties: a series of the same element, each with the same sub-elements.

I don't have to rewrite the above XForm each time I want to browse such files. I can make it general.

The trick is that in place of

```
<repeat ref="concert[contains(lower-case(.), lower-case(instance('search')/q))]">
```

to use

```
<repeat ref="*[contains(lower-case(.), lower-case(instance('search')/q))]">
```

which selects all top-level elements, regardless of what they are called. Then you do the same within each row:

```
<group class="row">
   <repeat ref="*">
      <output class="field" ref="."/>
   </repeat>
</group>
```

To supply the column headers we can just use the names of the elements, by taking the first row, and instead of outputting the value of the element, outputting its name:

```
<group class="header">
   <repeat ref="*[1]/*">
      <output class="field" value="name(.)"/>
   </repeat>
</group>
```

And right at the beginning, we can use the same trick to supply the top-level heading, by outputting the name of the root element:

```
<group>
   <label><output value="name(/*)"/></label>
```

Putting this together:

```
<group>
   <label><output value="name(/*)"/></label>
   <group>
      <input incremental="true" ref="instance('search')/q" label="Search (case-insensitive)"
      <output value="count(*[contains(lower-case(.), lower-case(instance('search')/q))])"/>
   </group>
   <group class="header">
      <repeat ref="*[1]/*">
         <output class="field" value="name(.)"/>
      </repeat>
   </group>
   <repeat ref="*[contains(lower-case(.), lower-case(instance('search')/q))]">
      <group class="row">
         <repeat ref="*">
            <output class="field" ref="."/>
         </repeat>
      </group>
   </repeat>
</group>
```

You'll see I have also added an output saying how many search results have been found:

```
<output value="count(*[contains(lower-case(.), lower-case(instance('search')/q))])"/> found
```

Here it is in action:

talks
Search (case-insensitive)
167 found

type	date	title	event	location
reviewed	2019-02-07	On the Specification of Invisible XML	XML Prague	Prague, Czech Republic
reviewed	2019-01-17	The Internet Effect	ISOC.nl New Year Event	Tolhuistuin, Amsterdam, NL
invited	2019-01-11	Internet@30	Freelance Factory	Pakhuis de Zwijger, Amsterdam, NL
keynote	2018-12-01	Gutenberg and the Internet	Media Art Festival	Leeuwarden, NL
keynote	2018-10-23	The Internet Effect	Cardiff Education Convention	Cardiff, Wales
invited	2018-10-06	4 uur 'snachts is de nieuw middernacht	Science Park Open Day	Amsterdam, NL
invited	2018-09-12	Invisible Markup	XML Summer School	St Edmund Hall, Oxford, UK
invited	2018-09-12	Declarative Applications	XML Summer School	St Edmund Hall, Oxford, UK
reviewed	2018-08-15	Fine-grained Access Control Framework for Igor, a Unified Access Solution to The Internet of Things	FIT 2018: the 4th International Workshop on the Future of the Internet of Things	Gran Canaria, Spain
reviewed	2018-08-02	XForms 2.0	Balisage	Washington DC, USA
invited	2018-08-01	On the Descriptions of	Balisage	Washington DC, USA

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/dataview/generic.xhtml)

Since this works with many different documents, we can now add a control to load different ones. We'll add an extra field to the `search` instance, to hold the name of the document we want to load:

```
<instance id="search">
   <data xmlns=""><q/><file/></data>
</instance>
```

and a control to select a different document:

```
<select1 ref="instance('search')/file"><label></label>
   <item><label>Talks</label><value>talks.xml</value></item>
   <item><label>Computers</label><value>computers.xml</value></item>
   <action ev:event="xforms-value-changed">
      <send submission="load"/>
      <setvalue ref="instance('search')/q"/>
   </action>
</select1>
```

Whenever the value changes, the following submission is activated, which loads the selected file, and replaces the data instance with it:

```
<submission id="load" resource="{instance('search')/file}" serialization="none"
            replace="instance" instance="data"/>
```

For this we need to add an `id` to the data instance:

```
<instance id="data" src="talks.xml"/>
```

Additionally when the value changes, the search string is reset.

Here it is in action:

talks

Talks

Search (case-insensitive)

167 found

type	date	title	event	location
reviewed	2019-02-07	On the Specification of Invisible XML	XML Prague	Prague, Czech Republic
reviewed	2019-01-17	The Internet Effect	ISOC.nl New Year Event	Tolhuistuin, Amsterdam, NL
invited	2019-01-11	Internet@30	Freelance Factory	Pakhuis de Zwijger, Amsterdam, NL
keynote	2018-12-01	Gutenberg and the Internet	Media Art Festival	Leeuwarden, NL
keynote	2018-10-23	The Internet Effect	Cardiff Education Convention	Cardiff, Wales
invited	2018-10-06	4 uur 'snachts is de nieuw middernacht	Science Park Open Day	Amsterdam, NL
invited	2018-09-12	Invisible Markup	XML Summer School	St Edmund Hall, Oxford, UK
invited	2018-09-12	Declarative Applications	XML Summer School	St Edmund Hall, Oxford, UK
reviewed	2018-08-15	Fine-grained Access Control Framework for Igor, a Unified Access Solution to The Internet of Things	FIT 2018: the 4th International Workshop on the Future of the Internet of Things	Gran Canaria, Spain
reviewed	2018-08-02	XForms 2.0	Balisage	Washington DC, USA

Source (https://homepages.cwi.nl/~steven/forms/examples/xmlcom/dataview/generic1.xhtml)

Adding more documents just involves adding new `<item>` elements to the `<select1>` control.

Even though this version of the application is more general, it is actually the same size as the first one.

It's worth pointing out that unlike the first version, this version requires each row of the data to have the same structure: all the sub elements present, and in the same order.

Related links

- An Introduction to XForms (/articles/2018/11/27/introduction-xforms/)