



ALEJANDRO TARDÍN

Liferay Headless

Present & Future

 Liferay® **DEVCON**

Demo time!

What could possibly go wrong?

Scan this QR Code



<https://devcon-headless.ngrok.dev/vote>

 Liferay® **DEVCON**

Results

Let's see what you voted

Modelling data

Objects

Related objects:

Poll -> Option -> Vote

Account	company	Yes	May 18, 2023, 2:58:05 PM
User	company	Yes	May 18, 2023, 2:58:05 PM
Commerce Product Group	company	Yes	May 18, 2023, 2:58:05 PM
Commerce Product	company	Yes	May 18, 2023, 2:58:06 PM
Commerce Order	company	Yes	May 18, 2023, 2:58:06 PM
Poll Vote	company	Yes	May 19, 2023, 4:16:22 PM
Poll Option	company	Yes	May 19, 2023, 4:16:22 PM
Poll	company	Yes	May 19, 2023, 4:16:23 PM

20 Items ▾ Showing 1 to 9 of 9 entries.

Importing the data

Import / Export Center

UI to import & export data using batch engine.

Still behind a dev feature flag (COMMERCE-8087).

Located under *Applications* -> *Import / Export* -> *Import / Export Center*.

The screenshot shows the 'Import' configuration page in Liferay. It is divided into several sections:

- Import Settings:** Includes a 'Name' field with the value 'definitions', a 'Template' dropdown, and an 'Entity Type' dropdown set to 'ObjectDefinition (v1_0 - Liferay)'. There is a 'Download a Sample File for This Entity: Download' button. A checkbox 'Stop the import on Error' is checked. The 'Import Strategy' is set to 'Only Add New Records' and the 'Update Strategy' is set to 'Overwrite Records'.
- File Settings:** The 'Upload a File from My Computer' radio button is selected. The 'File (.json, .jsonl)' field contains 'definitions.json'.
- Import Mappings:** A table with columns 'Destination Field', 'Source File Field', and 'Preview'. It lists several fields under the heading 'OPTIONAL FIELDS':

Destination Field	Source File Field	Preview
accountEntryRestricted	accountEntryRestricted	false
accountEntryRestrictedObjectFieldName		
active		
defaultLanguageId		

Importing the poll

In this case we imported the poll through [this JSON file](#).

The poll options are created in the same file thanks to [LPS-153117](#).

```
{
  "description": "What's your favourite Zelda game?",
  "externalReferenceCode": "zelda",
  "name": "Zelda",
  "options": [
    {
      "externalReferenceCode": "none",
      "value": "I don't play Zelda 😞"
    },
    {
      "externalReferenceCode": "the-legend-of-zelda",
      "value": "The Legend of Zelda 🗡️"
    },
    {
      "externalReferenceCode": "zelda-ii-the-adventure-of-link",
      "value": "Zelda II: The Adventure of Link 🗡️"
    }
  ]
}
```

Batch Engine API

- Used by Import / Export Center.
- Supported for every Headless API by default.
- Asynchronous.

```
curl \  
--location 'http://localhost:8080/o/headless-batch-engine/v1.0/import-task/{className}'\  
--header 'Content-Type: application/json'\  
--data '[  
  {  
    "prop": "value1",  
  },  
  {  
    "prop": "value2",  
  },  
  ...  
]'
```

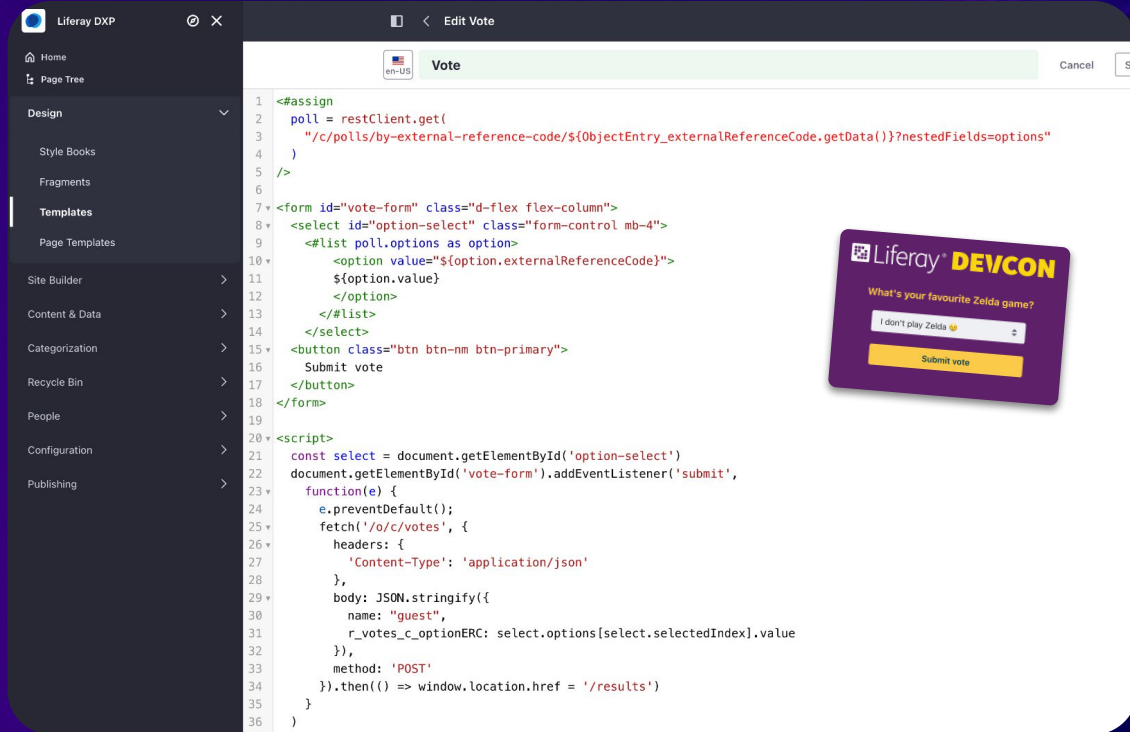

Rendering vote options

Server side rendering

Using information templates and the `restClient` variable.

Using Liferay Objects Headless APIs to get all the options by external reference code.

Getting related elements (options) via `nestedFields`.



The screenshot shows the Liferay DXP editor interface for editing a 'Vote' page. The left sidebar contains navigation options like Home, Page Tree, Design, Templates, Site Builder, Content & Data, Categorization, Recycle Bin, People, Configuration, and Publishing. The main editor area displays the following code:

```

1 <#assign
2 poll = restClient.get(
3   "/c/polls/by-external-reference-code/${ObjectEntry_externalReferenceCode.getData()}?nestedFields=options"
4 )
5 />
6
7 <form id="vote-form" class="d-flex flex-column">
8 <select id="option-select" class="form-control mb-4">
9   <#list poll.options as option>
10    <option value="${option.externalReferenceCode}">
11      ${option.value}
12    </option>
13  </#list>
14 </select>
15 <button class="btn btn-nm btn-primary">
16   Submit vote
17 </button>
18 </form>
19
20 <script>
21 const select = document.getElementById('option-select')
22 document.getElementById('vote-form').addEventListener('submit',
23   function(e) {
24     e.preventDefault();
25     fetch('/o/c/votes', {
26       headers: {
27         'Content-Type': 'application/json'
28       },
29       body: JSON.stringify({
30         name: "guest",
31         r_votes_c_optionERC: select.options[select.selectedIndex].value
32       }),
33       method: 'POST'
34     }).then(() => window.location.href = '/results')
35   }
36 )

```

Overlaid on the right side of the editor is a preview of the rendered page. It features the Liferay DEVCON logo at the top, followed by the question "What's your favourite Zelda game?". Below the question is a text input field containing "I don't play Zelda" and a dropdown arrow. At the bottom of the form is a yellow "Submit vote" button.

restClient template variable

- Added to every template by default.
- Can call any Headless GET endpoint.
- The request is performed server side.

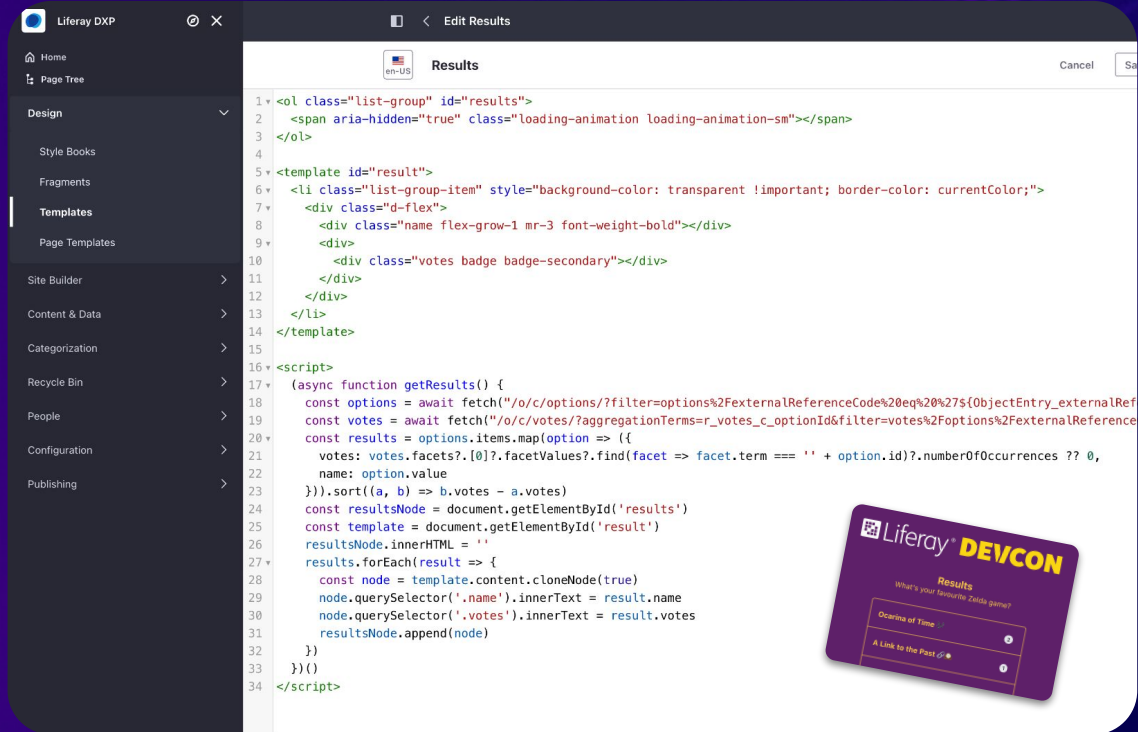
```
<#assign  
  poll = restClient.get(  
    "/c/polls/by-external-reference-code/zelda?nestedFields=options"  
  )  
</>
```

Displaying results

Client side rendering

Filtering by related objects
(all the votes for a specific
poll) thanks to [LPS-165819](#).

Using [aggregationTerms](#) to
count the votes.



The screenshot shows the Liferay DXP 'Edit Results' interface. On the left is a navigation sidebar with options like Home, Page Tree, Design, Style Books, Fragments, Templates, Page Templates, Site Builder, Content & Data, Categorization, Recycle Bin, People, Configuration, and Publishing. The main area displays the 'Results' page with a code editor showing the following code:

```

1 <ol class="list-group" id="results">
2   <span aria-hidden="true" class="loading-animation loading-animation-sm"></span>
3 </ol>
4
5 <template id="result">
6   <li class="list-group-item" style="background-color: transparent !important; border-color: currentColor;">
7     <div class="d-flex">
8       <div class="name flex-grow-1 mr-3 font-weight-bold"></div>
9       <div>
10        <div class="votes badge badge-secondary"></div>
11      </div>
12    </li>
13  </template>
14
15 <script>
16 (async function getResults() {
17   const options = await fetch("/o/c/options/?filter=options%2FexternalReferenceCode%20eq%20%27%7BobjectEntry_externalRef
18   const votes = await fetch("/o/c/votes/?aggregationTerms=_votes_c_optionId&filter=votes%2Foptions%2FexternalReference
19   const results = options.items.map(option => {
20     votes: votes.facets?.[0]?.facetValues?.find(facet => facet.term === '' + option.id)?.numberOfOccurrences ?? 0,
21     name: option.value
22   })).sort((a, b) => b.votes - a.votes)
23   const resultsNode = document.getElementById('results')
24   const template = document.getElementById('result')
25   resultsNode.innerHTML = ''
26   results.forEach(result => {
27     const node = template.content.cloneNode(true)
28     node.querySelector('.name').innerText = result.name
29     node.querySelector('.votes').innerText = result.votes
30     resultsNode.appendChild(node)
31   })
32 })()
33 </script>

```

On the right, a preview of the rendered results page is shown. It features the Liferay DEVCON logo, the title 'Results', and the question 'What's your favourite Zelda game?'. Below the question, there are two items listed: 'Ocarina of Time' with 3 votes and 'A Link to the Past' with 1 vote.

Filtering by related objects

- Uses odata complex data types.
- Supports odata operators (contains, startsWith...).
- Name of the complex type = name of the relationship.

```
filter=votes/options/externalReferenceCode eq 'zelda'
```



Demo GitHub Repo:

[4lejandrito/liferay-devcon-headless-2023](https://github.com/4lejandrito/liferay-devcon-headless-2023)

 Liferay® **DEVCON**

And now what?



The future of Liferay
Headless APIs

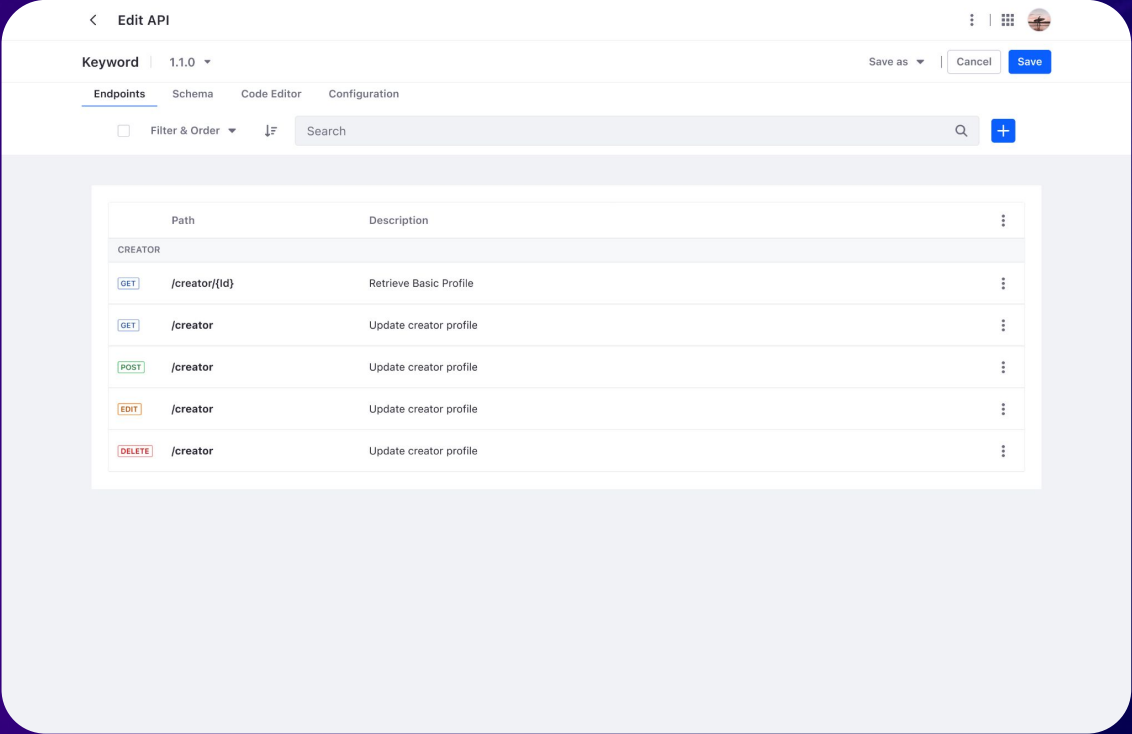
API Builder!

Manage your API

Provide built-in, scalable alternative to REST builder!

Define and manage your API for your Object definitions.

Portable API configurations to deploy in different instances.

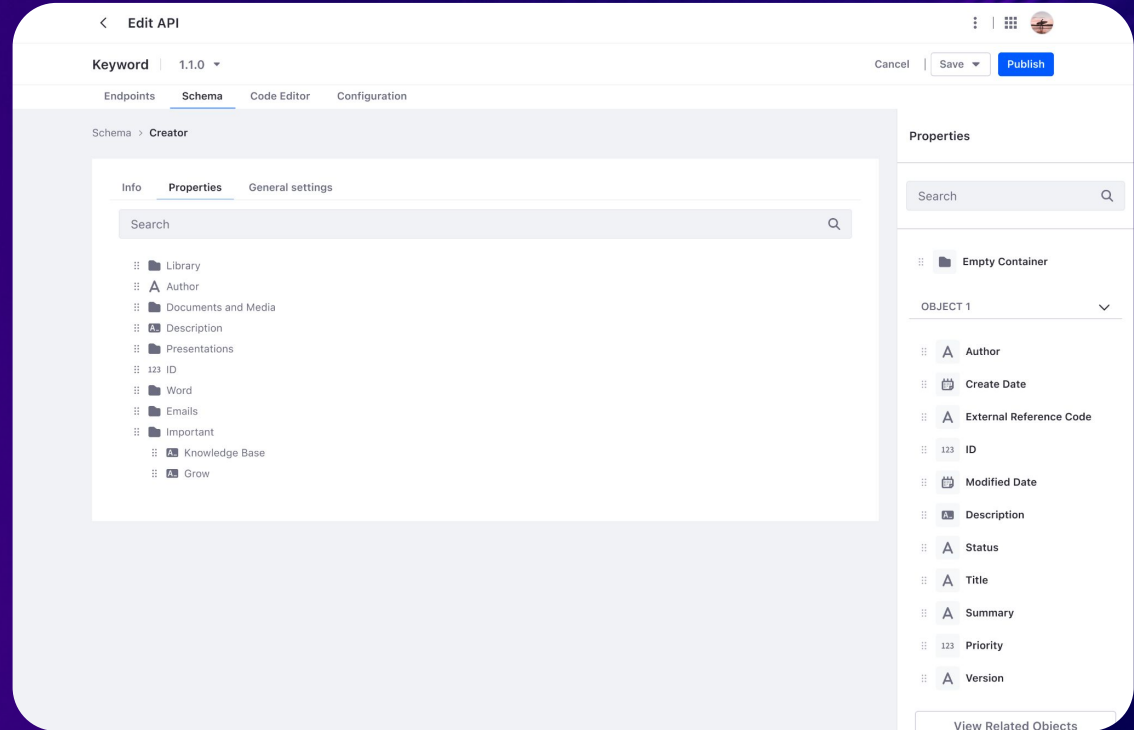


API Builder

Define your schemas

Model your API schemas through drag & drop.

Choose fields from your object and related elements.

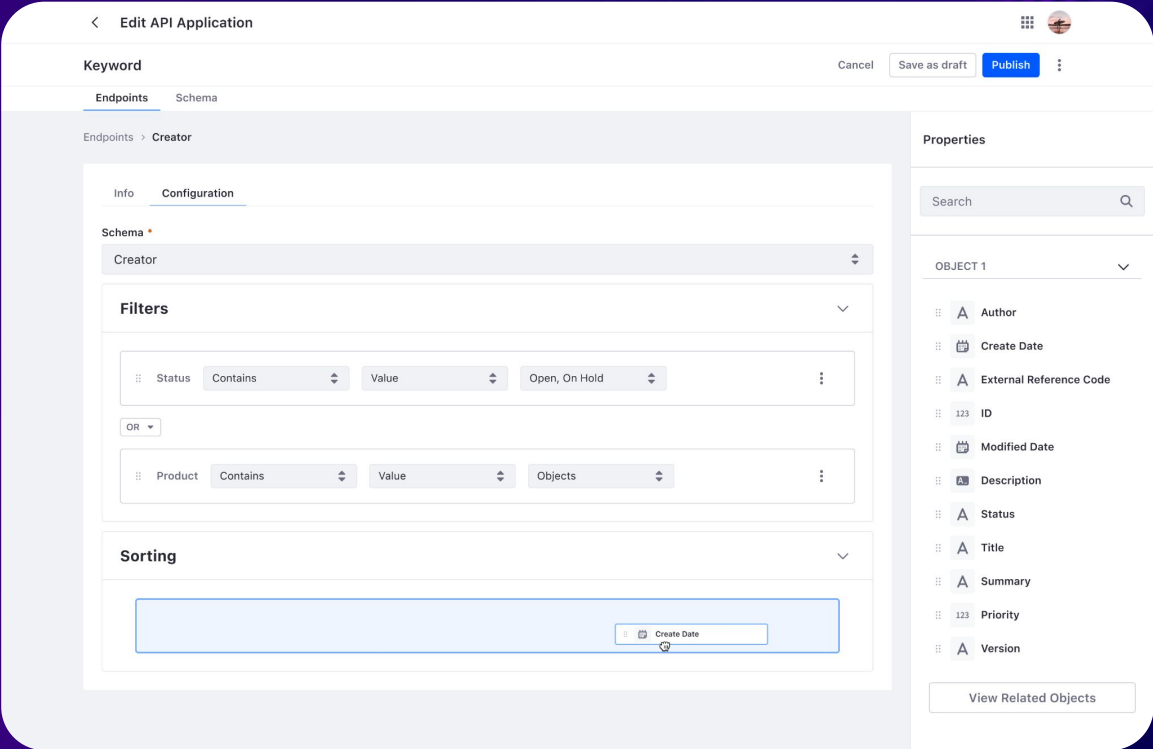


API Builder

Select your data

Define the behavior of your endpoints:

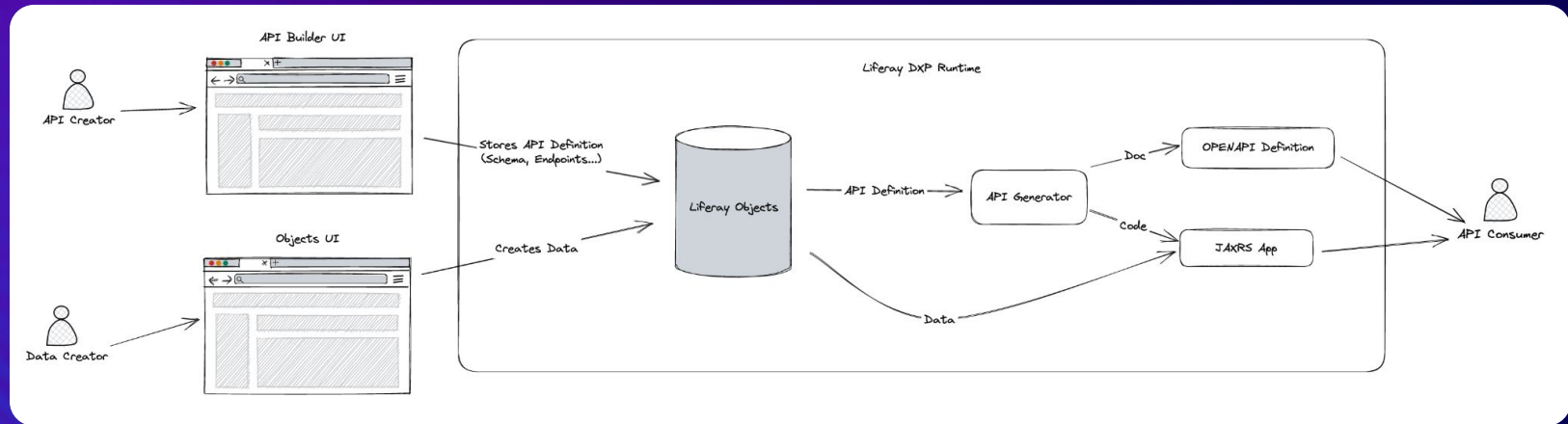
- Pre filter your data.
- Apply predefined sorting.



API Builder

Architecture

- API Definition is stored in Objects.
- The API generator transforms the API Definition into a JAXRS App.
- All at runtime, no more build steps.



Brought to you by the Headless Team





How was this session?

Please share your rating in
the event app. Thank you!