Liferay DEVCON

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.

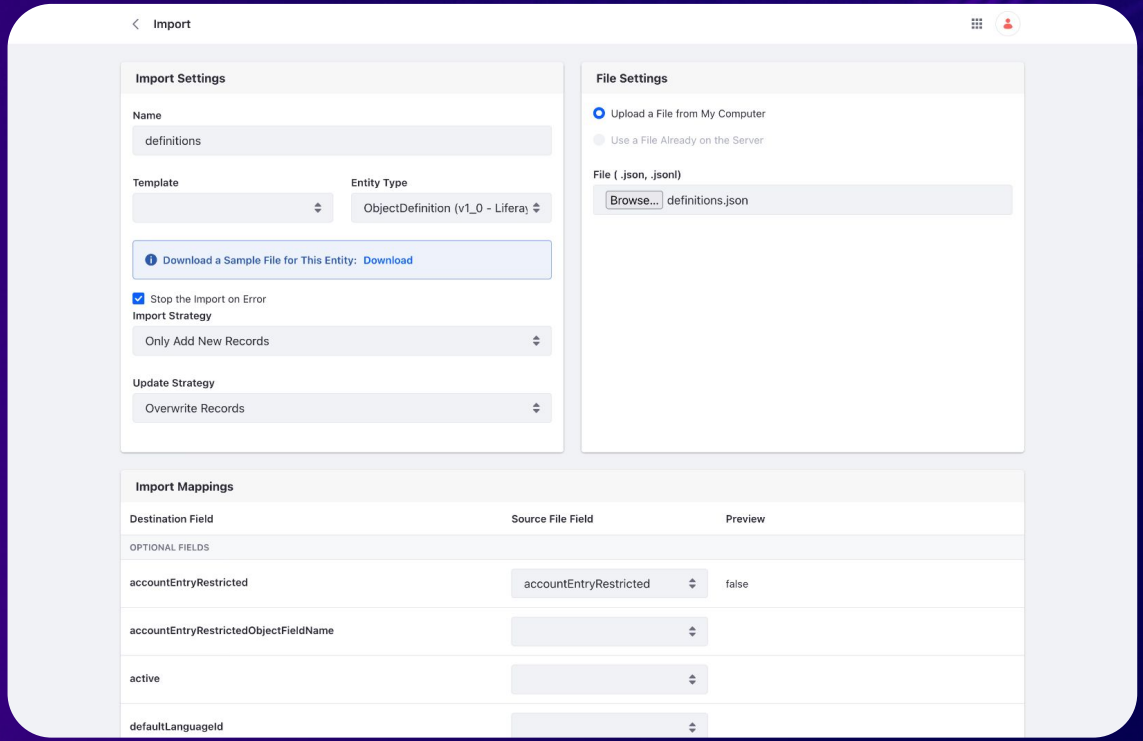
Importing the data

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*.



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",
    },
    ...
]'
```

Read documentation

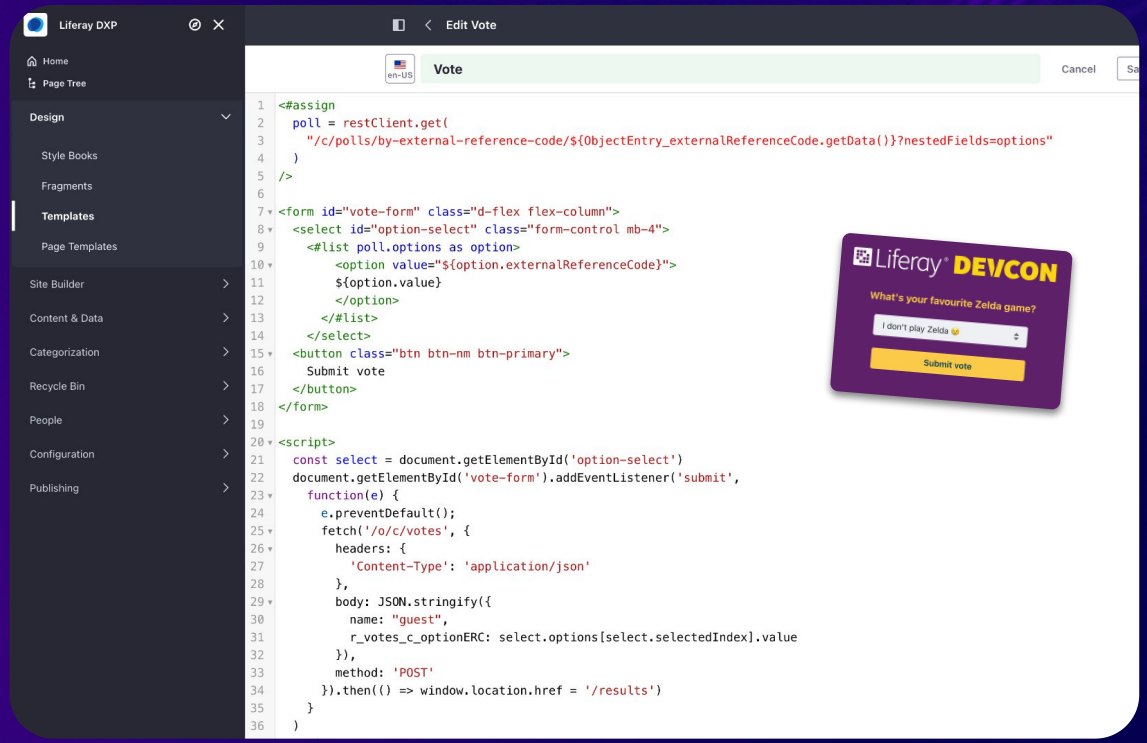
Rendering vote options

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.



restClient template variable

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

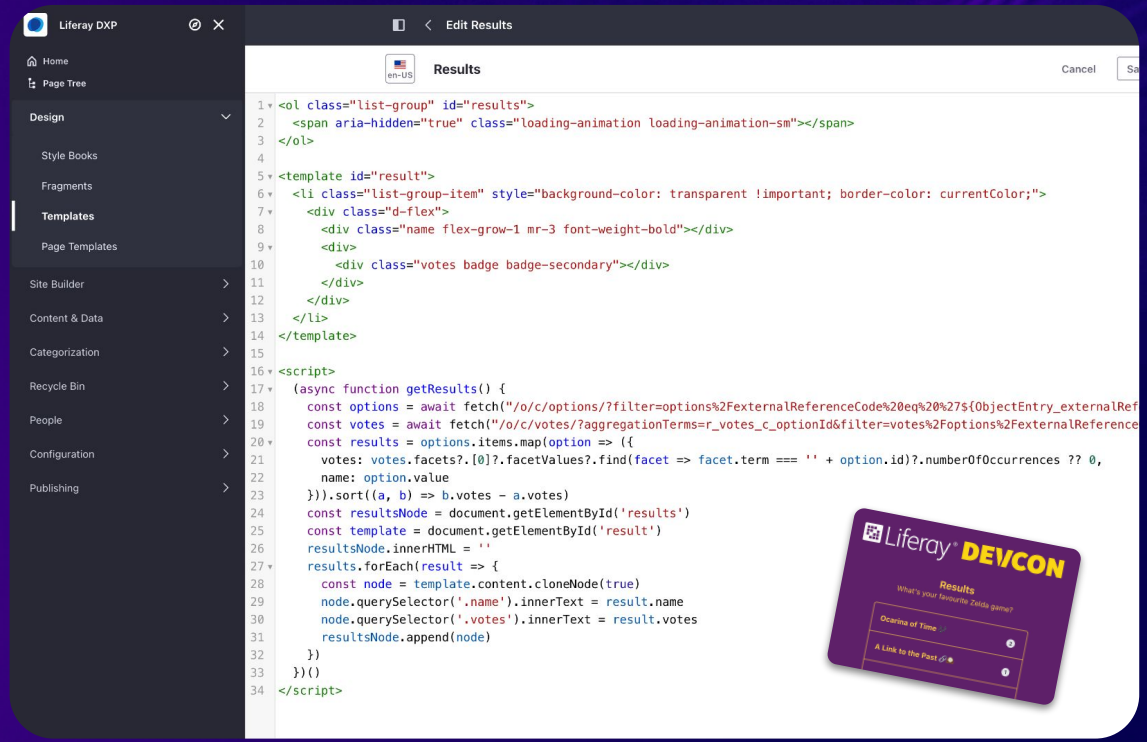
```
<#assign
  poll = restClient.get(
    "/c/polls/by-external-reference-code/zelda?nestedFields=options"
  )
/>
```

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.



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

Liferay® DEVCON

And now what? 

The future of Liferay
Headless APIs

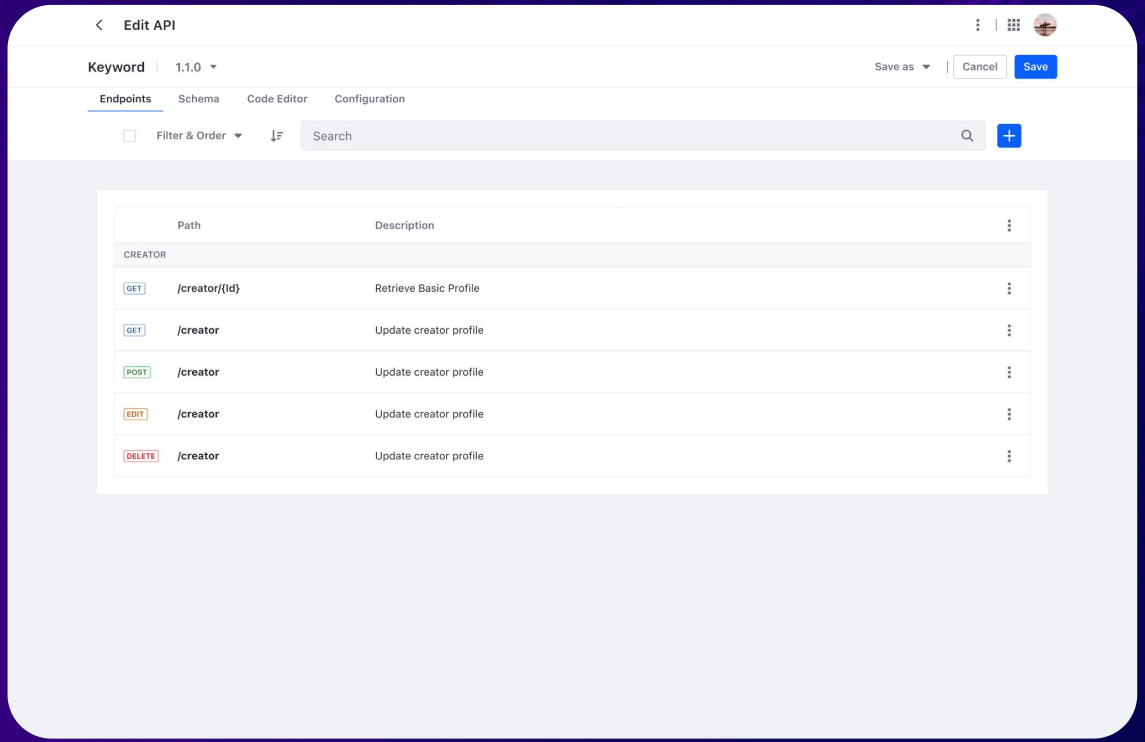
API Builder!

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.

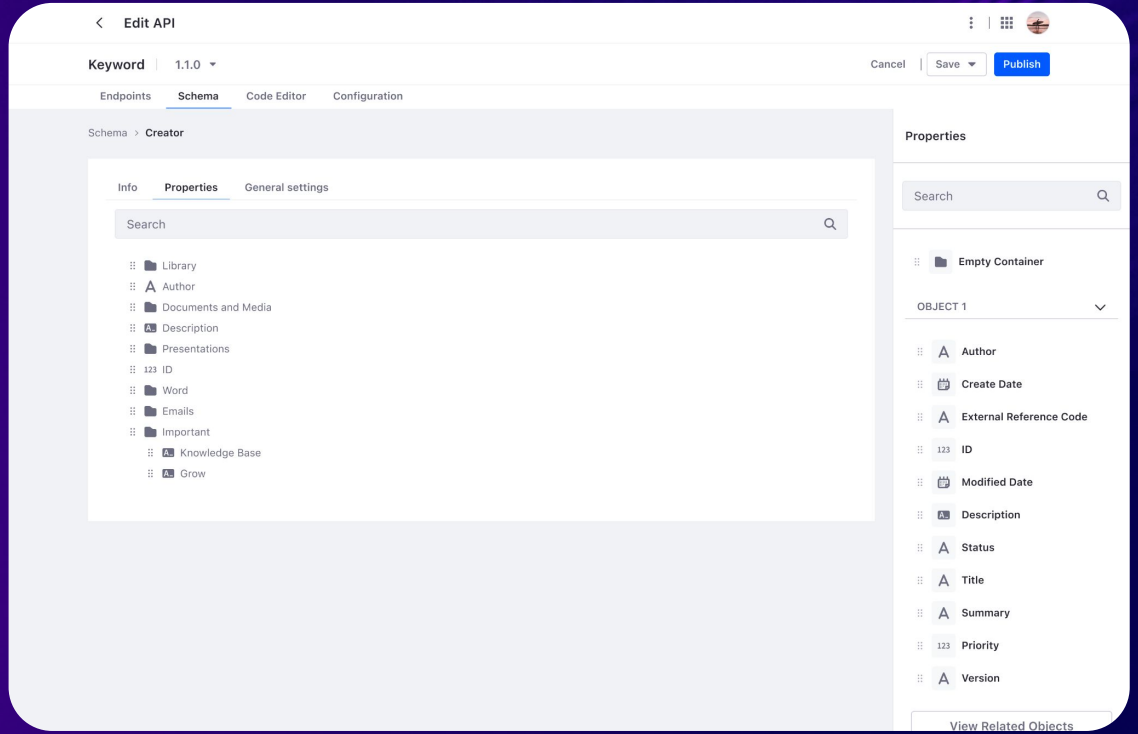


API Builder

Define your schemas

Model your API schemas through drag & drop.

Choose fields from your object and related elements.

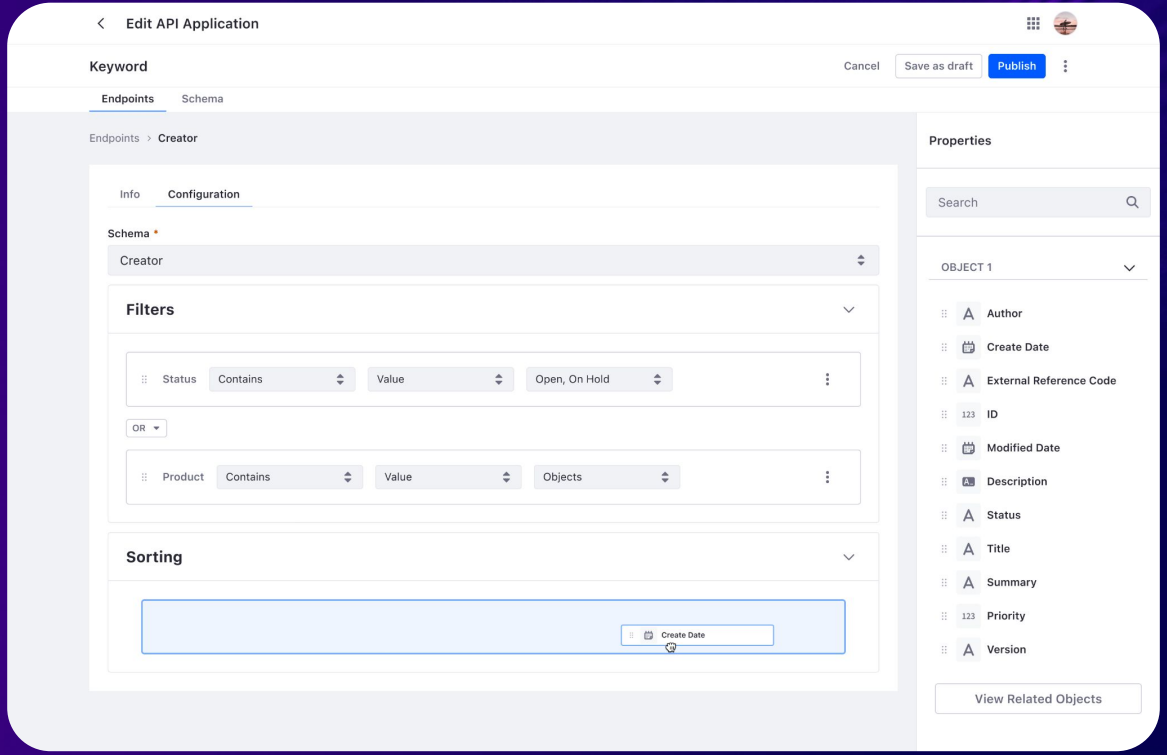


API Builder

Select your data

Define the behavior of your endpoints:

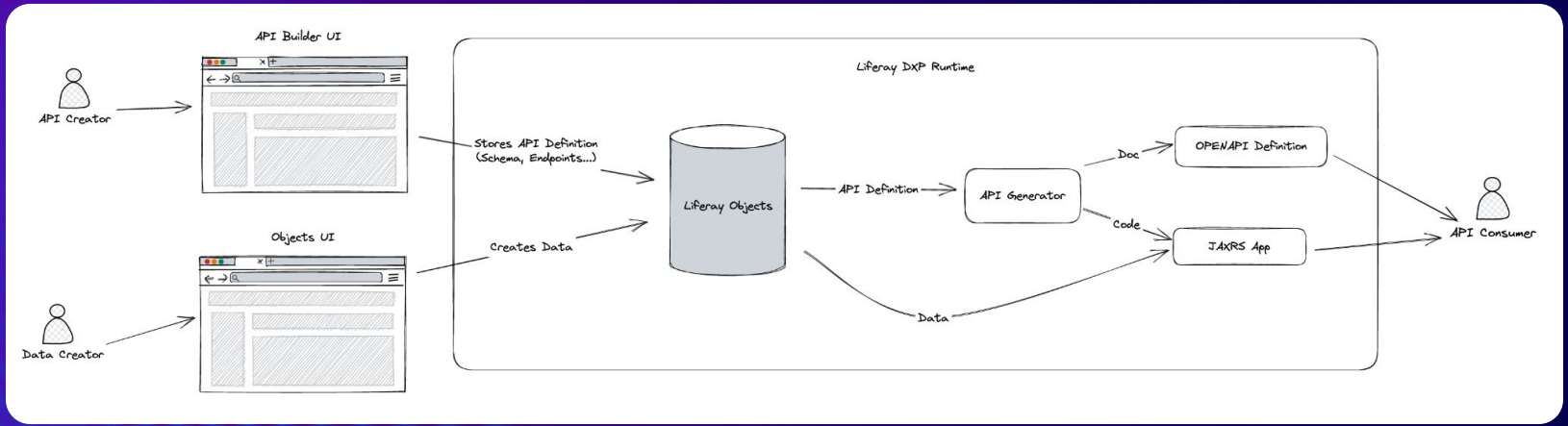
- Pre filter your data.
- Apply predefined sorting.



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!