

**Programming Boot Camp**

# **Connecting Bubble and Vercel API**

**Tokyo Institute of Technology 2023/11/25**

**Naotake KYOGOKU**

In the learning phase up to this point

- Development of Web applications using Bubble
- Development of Web API using Vercel

Web API development using Vercel

With the knowledge up to this point, you can make the connection between the two!

So let's try it as an exercise!

Used by.

- Bubble application created in the 2nd session
- Vercel application created in the 4th session (already deployed)

Let's combine them to realize the following functions

Step1. Get pet list data from Vercel API

Step2. Get detailed pet data from Vercel API

Step3. Pet registration via Vercel API

Step4. Pet weight display & weight registration with Vercel API

⚠ To make it an exercise, only hints are given thereafter.

⚠ Those who have completed the development process will receive a copy of the Bubble application for their answers, which they can use to test their answers.

If you have finished that in plenty of time, too, you can proceed here in a more advanced way!

## Step5. Search the list of pets by name

- This one needs to address Prisma as well, so please try to pair up with someone to go ahead with it if possible!
- One of you will revamp the Prisma side 🧑🏻
- The other one is to revamp the Bubble side 🧑🏻.

**First, let me show you the finished application 🙄🙄**

In every step, there are two main things you need to do

- Set up a connection between Bubble and Vercel API.
- Connect Bubble to Vercel API and display the screen.

# Let's try it!



⚠️ If you have not attended some or all of the previous Learning Phases, please let us know if you need the following so we can share it with you! 🙋

- Bubble application created in the 2nd session
- Vercel application created in the 4th session (already deployed)

Here's a hint  .



# **Step1. Get pet list data from Vercel API**

# Bubble's API Connector configuration looks like this

move down collapse

Name

List

Use as

Data

Data type

JSON

GET

https://[REDACTED]/api/pets

(use [] for params)

Attempt to make the call from the browser

☐

Headers

Add header

Parameters

[REDACTED]

Add parameter

Include errors in response and allow workflow actions to continue

☐

Capture response headers

☐

This call has been modified since you last initialized; consider reinitializing.

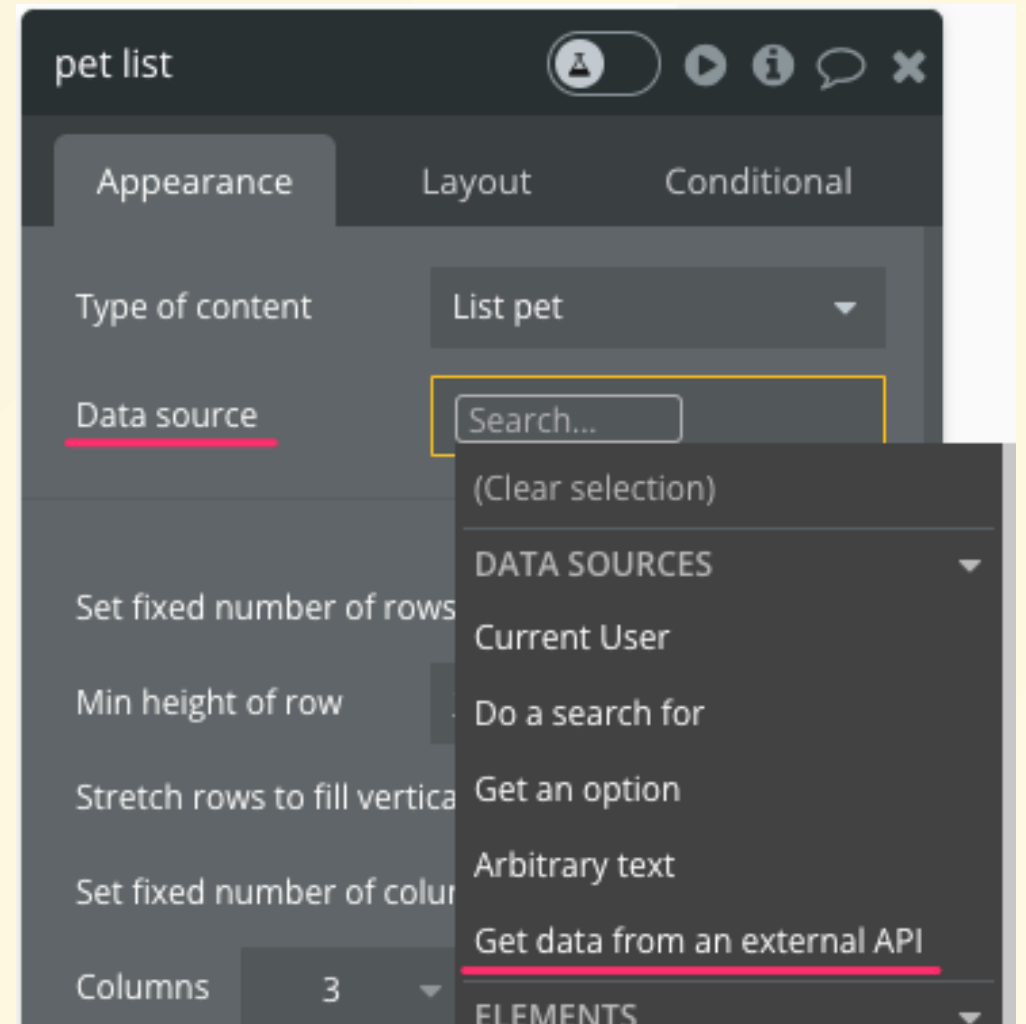
Reinitialize call

Manually enter API response

- When calling the API from the Data source for each data display item, it is

Get data from an external API

•





## **Step2. Get detailed pet data from Vercel API**



# Bubble's API Connector configuration looks like this

move up

move down

collapse

Name

Detail

Use as

Data

Data type

JSON

GET

https://[redacted]/api/pets/[pet\_id]

(use [] for params)

URL parameters

Key

pet\_id

Value

1

Private

☐

Allow blank

☐

Attempt to make the call from the browser

☐

Headers

Add header

Parameters

Add parameter

Include errors in response and allow workflow actions to continue

☐

Capture response headers

☐

Reinitialize call

Manually enter API response

- When moving from the list screen to the detail screen, only the ID of the pet is passed.
- Since `Data to send` is not available, use `Send more parameters to the page` instead.
- Then, on the detail screen, `When the screen is displayed`, call the API to get the pet's details using the pet's ID received by the URL parameter.
- The result is stored in the detail screen itself using Bubble's `Custom state` function.
  - This part may be a little difficult 🔥.
- Then, each item on the detail screen is displayed based on the value stored in the state.



## **Step3. Pet registration via Vercel API**

# Bubble's API Connector configuration looks like this

move up

move down

collapse

Name

Register

Use as

Action

Data type

JSON

POST

https://[redacted]/api/pets

(use [] for params)

Headers

Add header

Body type

JSON

Parameters

Add parameter

Body (JSON object, use <> for dynamic values)

```
1 {
2   "pet": {
3     "name": "<pet_name>",
4     "imageUrl": "<pet_image_url>",
5     "birthDate": "<pet_birthday>T00:00:00.000Z",
6     "gender": "<pet_gender>",
7     "ownerId": 1
8   }
9 }
```

Body parameters

Key

pet\_name

Value

tara

Private

☐

Allow blank

☐

Key

pet\_image\_url

Value

https://www.google.com/

Private

☐

Allow blank

☐

Key

pet\_birthday

Value

2023-11-23

Private

☐

Allow blank

☐

Key

pet\_gender

Value

Female

Private

☐

Allow blank

☐

Include errors in response and allow workflow actions to continue

☐

Capture response headers

☐

Reinitialize call

Manually enter API response



## **Step4. Pet weight display & weight registration with Vercel API**

# Bubble's API Connector configuration looks like this

move up

move down

collapse

Name

Weight Register

Use as

Action

Data type

JSON

POST

https://learning-phase-4-naotakke.vercel.app/api/pets/[pet\_id]/weights

(use [] for params)

URL parameters

Key

pet\_id

Value

1

Private

Allow blank

Headers

Add header

Body type

JSON

Parameters

Add parameter

Body (JSON object, use <> for dynamic values)

```
1 {
2   "weight": {
3     "weight": <weight>,
4     "petid": <pet_id>
5   }
6 }
```

Body parameters

Key

weight

Value

12

Private

Allow blank

Key

pet\_id

Value

1

Private

Allow blank

Include errors in response and allow workflow actions to continue

Capture response headers

Reinitialize call

Manually enter API response



- The pet weight display is included in the results of the API we prepared for the pet details screen, so we can use that!
- The concept of registering a pet's weight is similar to that of registering a pet, so we can use that as a reference.



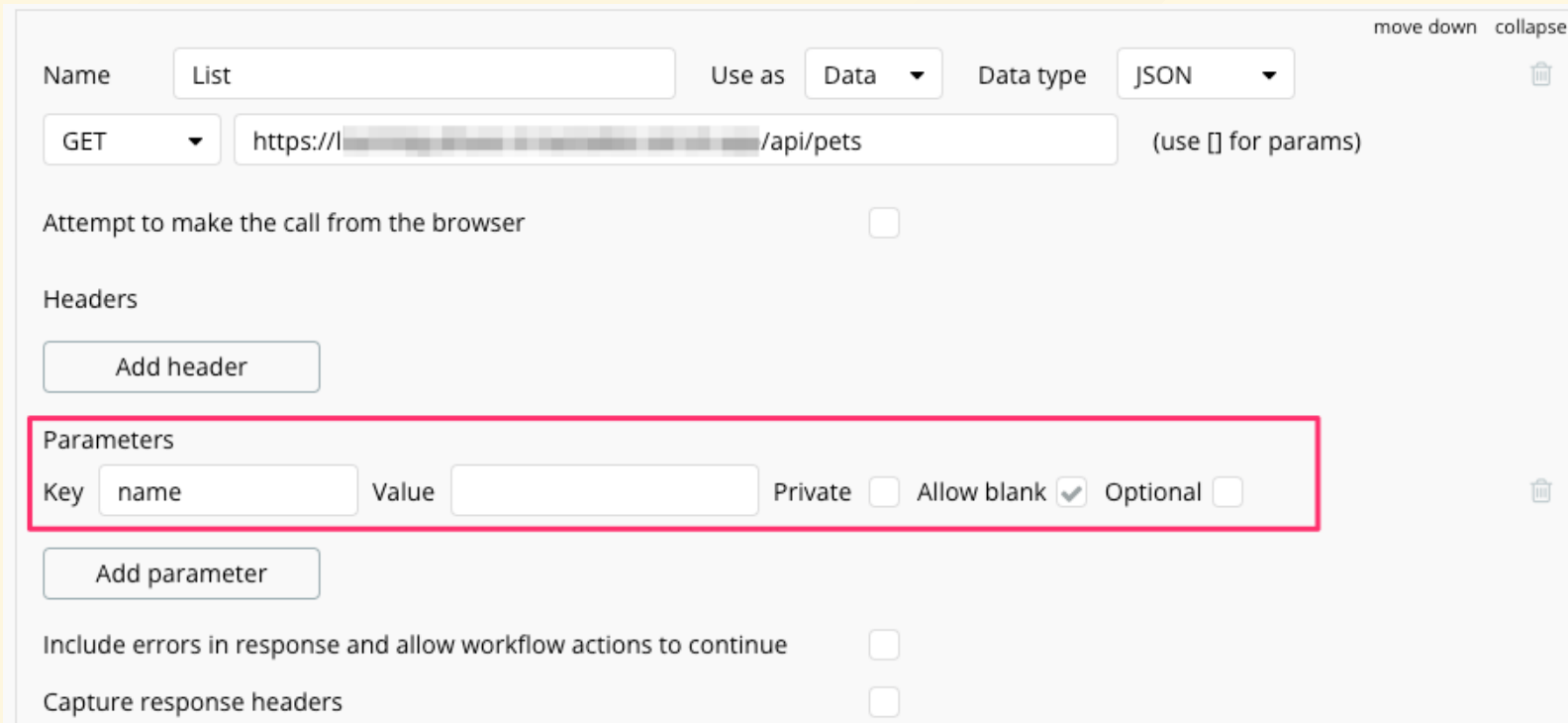
## **Step5. Search the list of pets by name**

# Here's what the Prisma side of the renovation looks like

[Pull request for additional search criteria](#)

# Here's what the Bubble side of the renovation looks like

- This is a form of changing the settings for acquiring a list prepared above.



The screenshot shows the 'List' configuration form in Bubble. The form includes fields for Name, Use as, Data type, Method, URL, and various checkboxes for browser calls, headers, parameters, and error handling. The 'Parameters' section is highlighted with a red box.

move down collapse

Name  Use as  Data type

(use [] for params)

Attempt to make the call from the browser ☐

Headers

Parameters

Key  Value  Private ☐ Allow blank ☒ Optional ☐

Include errors in response and allow workflow actions to continue ☐

Capture response headers ☐



**That's all!**

**Thank you for your hard work!**