Ian5 1 July 13, 2025, 9:13pm

I am using Javascript on a Knack page with a grid view to compute the value of two columns of the grid and then store the results back into the database in those columns. My Javascript uses a knack-view-render handler to do the computation and poke the values into the grid's HTML for display purposes. The knack-view-render handler also hooks the OnClick of a button on the same page, and that handler issues view-based PUTs to set the values in those columns back into the corresponding columns in the database table. These PUTs are not working. I have tried to follow the API documentation carefully, and I can see no problems in my code, but the API returns status code 400, and the body of the response says only this:

```
{"errors":[{"message":"Invalid request"}]}
```

Is there a way to get more detailed feedback about why the request is invalid?

Additional details: To invoke the view-based API, I am calling fetch with this URL:

```
https://api.knack.com/v1/pages/scene_563/views/view_1380/records/6791ac809ce49e02ce343388
```

and with this options object:

```
{
   method: "PUT",
   mode: "cors",
   cache: "no-store",
   headers: {
      'Content-Type': 'application/json',
      'X-Knack-Application-Id': Knack.application_id,
      'X-Knack-REST-API-KEY': 'knack',
      'Authorization': Knack.getUserToken(),
   },
   body: "{\"field_2027\":4,\"field_1737\":4}"
}
```

I've checked the scene, view, and field IDs, and I've also checked that the particular record ID exists in the grid. Any hints on what I'm doing wrong would be great, but even better would be a detailed error message in the HTTP response so I could diagnose the issue myself.

StephenChapman 2 July 14, 2025, 9:35pm

Hi @Ian5 , it's a bit tricky to troubleshoot without seeing the full code, but I'd be happy to jump on a call if you'd like to DM me. My theory is that your `body` isn't structured correctly.

I'd also recommend using @Callum Boase's **Knack API Helper library**, which I use in just about every app to make much simpler API calls. I'd be happy to show you how to use this too, otherwise you can reference Callum's video **here**.

Ian5 3 July 15, 2025, 3:59am

@StephenChapman , thanks so much for helping me out, here! I should have posted my code straight away — I guess I was hoping there was a magic way to get more detailed feedback from Knack.

My code is a bit complex, with a retry, but you can ignore most of that, I think. It executes straight through to the first fetch call and fails with status 400, so none of the retry logic is coming into play. If you need a simplified version I can provide that, but I'm out of time tonight, so I'll just post as-is for now:

```
    this.createOptionsBody = function(newRank, newAdjRank) {
        if (this.rankFieldId && this.adjRankFieldId) {
            return {
                [this.rankFieldId]: newRank,
                [this.adjRankFieldId]: newAdjRank,
            };
        } else if (this.rankFieldId) {
            return {
                [this.rankFieldId]: newRank,
            };
        } else if (this.adjRankFieldId) {
            return {
                [this.adjRankFieldId]: newAdjRank,
            };
        } else {
            throw new Error('At least one of this.rankFieldId and this.adjRankFie
        }
    }.bind(this);

    // Returns the number of milliseconds to wait before retrying, or zero if suc
    this.putRankToDatabase = async function(url, options, retryNumber) {
        const response = await fetch(url, options); // returns response
        console.log(`Fetch returned`);
        const hdrs = response.headers;
        console.log(hdrs);
        const planLimitRemaining = Number(hdrs.get("X-PlanLimit-Remaining"));
        console.log(`planLimitRemaining = ${planLimitRemaining}`);
        if (response.ok) {
```

The top-level function is the one at the bottom, `putRankToDatabaseWithRetry` . I hope it's not too hard to follow, but the first thing that function does is to build the URL and the options object for the call to fetch. The body is supplied by the first function, `createOptionsBody` . You can see that I've experimented with mode, cache directive, and accept header in the options, but those don't seem to make a difference.

Any ideas you have for things I should try are welcome. Thanks in advance!

StephenChapman 4 July 15, 2025, 4:34am

@Ian5 , I'm not going to lie, that was hard to follow.
Below is a starting point for utilising Knack API Helper, which has built-in auto-retries.
It's not the complete solution, but will hopefully give you an idea of how much simpler it could look:

- Ensure the `loadExternalFiles` function and call are added in, as this imports Knack API Helper into your code.
- Replace `view_xx` with your rendered view
- Replace `.button-class` with the button's class on your view, assuming this is custom-added?
- Adjust all your constant values to your needs
- Ensure your `PUT` scene/view is accessible to the logged-in user

```
// -----------------------------------------------------------
// PAGE LOAD FUNCTIONS
// -----------------------------------------------------------
loadExternalFiles ([
  { type: 'script', module: false, url: 'https://cdn.jsdelivr.net/npm/knack-api-h
]);
```

```
// ------------------------------------------------------------
// VIEW RENDER
// ------------------------------------------------------------
$(document).on('knack-view-render.view_any', async function(event, view, record)

  if (view.key === 'view_xx') {
    const recordId = record.id;
    const rankFieldId = 'field_2027';
    const adjRankFieldId = 'field_1737';
    const rank = 4;
    const adjRank = 4;
    const $btn = $(`#${view.key} .button-class`); // Some button on the view
    $btn.on('click', function() {
      const body = {
        [rankFieldId]: rank,
        [adjRankFieldId]: adjRank
      }
      await KnackAPI.makeRequest('put', {scene: 'scene_563', view: 'view_1380', r
    });
  }
```

Ian5 5 July 20, 2025, 4:16pm

It took me a few days to get back to this task, but @StephenChapman , again, thanks for your help.

I tried Knack API Helper. It didn't work at first. The Java console contained an error about KnackInitAsync not existing, so instead of loading the helper code that way I simply pasted it into my Javascript (which I load into Knack's Settings/API & Code/Javascript tab). That also enabled me to add a few console.log statements to capture the parameters of the fetch call so I could compare them to mine.

Bottom line: The fetch parameters were exactly the same (url and options object containing method, body, and headers), and Knack API Helper fails with exactly the same error.

So that makes me wonder whether the error is in my scene/view. I checked the scene, view, and field IDs one more time, and they are correct. The view itself is hidden on this page (display: none), but I can see it in my browser's "inspect element" view. Any ideas on what I am missing?

StephenChapman 6 July 20, 2025, 9:16pm

@Ian5 I sent you a message to book a call.

StephenChapman 7 July 22, 2025, 12:21am

For everyone watching at home, the main reason this wasn't working is because the `PUT` request was being performed on a grid view that did not have inline editing enabled.

@Ian5 I hope that helped! Are you able to please mark this post as solved?

1 Like

Related topics

Topic	Replies	Views	Activity
Having trouble doing a Put request using Jquery in Knack	3	719	March 17, 2023
No records returned from GET	3	446	January 7, 2023
Bad request 400 when calling up details view	0	391	December 31, 2020
New API user, help with view-based POST	4	421	May 17, 2018
CORS violations on API calls?	10	572	January 19, 2024