

Section 1: Debugging Techniques

Using `ipdb` for Debugging in Frappe

Step-by-Step Guide

Install `ipdb`: Make sure `ipdb` is installed in your development environment. You can install it using pip:

bash

Copy code

```
pip install ipdb
```

1. **Insert Breakpoint in Code:** Add the following line in your Python code where you want to start debugging:
python
Copy code

```
import ipdb; ipdb.set_trace()
```
2. This line will pause the execution and open the interactive debugging prompt when reached.
3. **Trigger the Code:** Run the part of your application that will hit the breakpoint. This could be a function call, an HTTP request, or any event that executes the code containing the breakpoint.

```
def customFunc(a,b):  
    print("Before breakpoint")  
    import ipdb; ipdb.set_trace()  
    result = a + b  
    print(f"Result: {result}")  
    return result
```

```
(env) amang@aman-Inspiron-15-3511:~/frappe-bench$ bench console  
Apps in this namespace:  
frappe, erpnext, custom_app  
  
In [1]: from custom_app.Customize_Events.Customize_Event_Customer import customFunc  
In [2]: customFunc(1,3)
```

```

> /home/amang/frappe-bench/apps/custom_app/custom_app/Customize_Events/Customize_Event_Customer.py(13)
tomFunc()
12     import ipdb; ipdb.set_trace()
--> 13     result = a + b
14     print(f"Result: {result}")

ipdb> n
> /home/amang/frappe-bench/apps/custom_app/custom_app/Customize_Events/Customize_Event_Customer.py(14)
tomFunc()
13     result = a + b
--> 14     print(f"Result: {result}")
15     return result

ipdb> n
Result: 4
/home/amang/frappe-bench/apps/custom_app/custom_app/Customize_Events/Customize_Event_Customer.py(15)

```

Debugging Client Scripts:

Debugging Client-Side Scripts

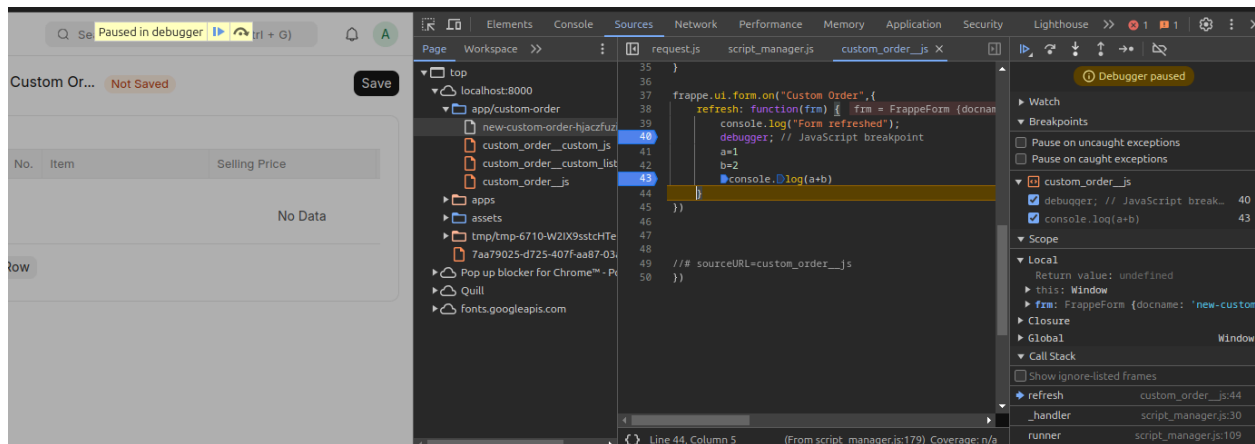
Using Browser Developer Tools

1. **Open Developer Tools:**
 - In most browsers, you can open the developer tools by pressing **F12** or by right-clicking on the page and selecting "Inspect".
2. **Inspect Elements:**
 - Navigate to the "Elements" tab to inspect the HTML structure of your Frappe app.
 - This allows you to see the DOM (Document Object Model) and understand how your elements are structured.
3. **View Console Output:**
 - Go to the "Console" tab to see the output of `console.log()` statements and any errors or warnings.
 - This is useful for understanding what is happening in your script at runtime.
4. **Set Breakpoints in JavaScript:**
 - Navigate to the "Sources" tab.
 - Locate your JavaScript file. In Frappe, this might be under the `webpack://` or an app-specific path.
 - Click on the line number to set a breakpoint.

Example

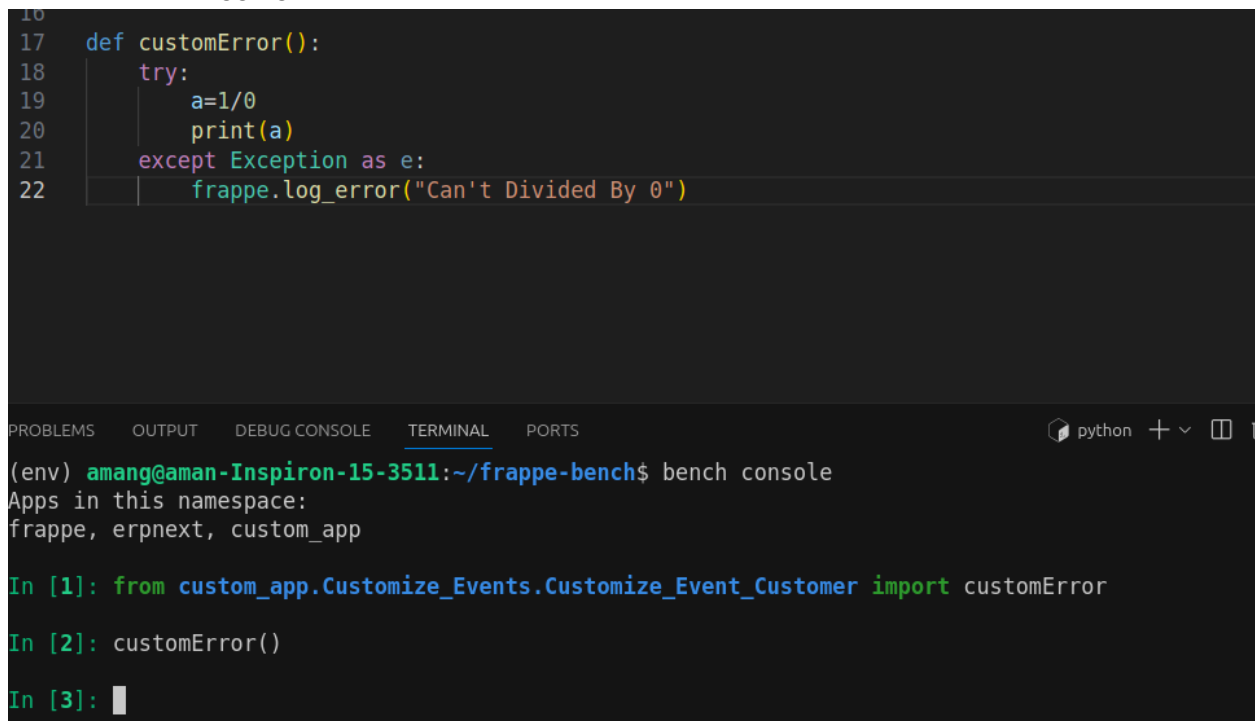
Consider you have a custom client script in Frappe:

```
frappe.ui.form.on('Custom Order', {
    refresh: function(frm) {
        console.log("Form refreshed");
        debugger; // JavaScript breakpoint
    }
});
```



Section 2: Error Logs

Custom Error Logging:



Accessing Error Logs:

Can't Divided By 0 Seen

Assigned To

Attachments

Tags

Share

0 · 0

FOLLOW

You last edited this · just now

You created this · just now

Title

Can't Divided By 0

Error

Traceback with variables (most recent call last):
File "apps/custom_app/custom_app/Customize_Events/Customize_Event_Customer.py", line 19, in customError
a=1/0
e = ZeroDivisionError('division by zero')
builtins.ZeroDivisionError: division by zero

Comments

A

Type a reply / comment

Activity

You created this · just now

You last edited this · just now

+ New Email

⚠

Connection lost. Some features may not work.

Section 3: Console Output

```
Console was cleared VM3076:1
< undefined
> console.log(cur_frm)
  ▶ FrappeForm {docname: 'u3iu0i5tig', doctype: 'Info', doctype_layout_name: undefined, in_form: true, hidden: false, ...} VM3164:1
< undefined
>
```