

*o-spreadsheet: develop custom  
functions to connect to your  
database or an external API*

Lucas Lefèvre, Software developer



@LucasLefevre



# Agenda


- 1 **Introduction**
- 2 Simple custom functions
- 3 Function reading external data
- 4 Performance pitfalls
- 5 Conclusion

# o-spreadsheet



<https://github.com/odoo/o-spreadsheet>



- Standalone library written in Typescript 
- Open source
- Integrated to Odoo
- Odoo is an external system from o-spreadsheet POV
- Extensible with plugins and custom functions

# What we are building today

=ODOO.SURVEY()

=ODOO.SURVEY(1)										
G	H	I	J	K	L	M	N	O	P	
	<b>Name</b>	<b>Birthday</b>	<b>Rating</b>	<b>Feedback</b>	<b>City</b>	<b>Country</b>	<b>How frequently do you buy products online?</b>			
	Martin Cohps	1979-07-04	4	Great	Wierde	Skipped	Once a year			
	Agnes Monica	1960-02-01	3	Good to know. I York	UK	Argentina	Once a year			
	Julia Ray	1999-06-14	4	See you next tin	Buenos Aires	Tunisie	Once a month			
	Karim West	2003-05-04	5	Reaaaaaally good	Tunis	France	Once a day			
	Loïc Rousseau	1992-08-15	4	I love sailing	Le Palais	New Zealand	Less than once a year			
	Clara Tegas	1960-10-17	5	Very welcoming	Wellington	Côte d'Ivoire	Once a year			
	Mariah Geury	1973-12-07	2	Not so pleased	Abidjan	Thailand	Once a week			
	Lucie Allard	1993-07-12	2	Thanks!	Bangkok	Mexico	Once a week			
	Marion Cousser	1998-01-04	4	Great! Congrats	Mexico	USA	Once a month			
	Clément Ouassi	2000-06-21	3	Not much to say	New York	Spain	Once a day			
	Hugo Domken	1990-02-12	5	Excellent. Best	Madrid	Skipped	Once a week			
	Antoine Stampe	1975-03-09	4	I liked it	Brussels		Once a month			

Code available <https://github.com/LucasLefevre/spreadsheet-functions-oxp2023>

# Declare a function

=MY.ADD(1, 2) 🌟

- Declare the **arguments**

```
/** @odoo-module */  
  
import { helpers, registries } from "@odoo/o-spreadsheet";  
import { _t } from "@web/core/l10n/translation";  
  
const { arg, toNumber } = helpers;  
  
const MY_ADD = {  
  description: _t("Adds two numbers together."),  
  args: [  
    arg("value1 (number)", _t("The first number")),  
    arg("value2 (number)", _t("The second number")),  
  ],  
  returns: [ "NUMBER" ],  
  // ...  
};
```

Documentation **Adding a new custom function**

[https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add\\_function.md](https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add_function.md)

# Declare a function

=MY.ADD(1, 2) 🌟

- Declare the **arguments**
- Implement **compute**
  - return a primitive value
  - return a 2d array of primitives

```
/** @odoo-module */  
  
import { helpers, registries } from "@odoo/o-spreadsheet";  
import { _t } from "@web/core/l10n/translation";  
  
const { arg, toNumber } = helpers;  
  
const MY_ADD = {  
  description: _t("Adds two numbers together."),  
  args: [  
    arg("value1 (number)", _t("The first number")),  
    arg("value2 (number)", _t("The second number")),  
  ],  
  returns: [ "NUMBER" ],  
  compute: function (value1, value2) {  
    // TODO  
  },  
};
```

Documentation **Adding a new custom function**

[https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add\\_function.md](https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add_function.md)

# Declare a function

=MY.ADD(1, 2) 

- Declare the **arguments**
- Implement **compute**
  - return a primitive value
  - return a 2d array of primitives
- **Cast** user inputs to required type
  - `toNumber("20%", locale) // 0.2`
- Add the function to the **registry**
- **References** work out of the box

Documentation **Adding a new custom function**

[https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add\\_function.md](https://github.com/odoo/o-spreadsheet/blob/17.0/doc/add_function.md)

```
/** @odoo-module */  
  
import { helpers, registries } from "@odoo/o-spreadsheet";  
import { _t } from "@web/core/l10n/translation";  
  
const { arg, toNumber } = helpers;  
  
const MY_ADD = {  
  description: _t("Adds two numbers together."),  
  args: [  
    arg("value1 (number)", _t("The first number")),  
    arg("value2 (number)", _t("The second number")),  
  ],  
  returns: [ "NUMBER" ],  
  compute: function (value1, value2) {  
    value1 = toNumber(value1, this.locale)  
    value2 = toNumber(value2, this.locale)  
    return value1 + value2;  
  },  
};  
  
registries.functionRegistry.add("MY.ADD", MY_ADD);
```



# Our function

=ODOO.SURVEY(1) 

- Returns a 2D matrix of values
- Each column is a question and answers
- How to fetch the data ?

```
/** @odoo-module */

import { _t } from "@web/core/l10n/translation";

import { helpers, registries } from "@odoo/o-spreadsheet";
const { arg, toNumber } = helpers;

registries.functionRegistry.add("ODOO.SURVEY", {
  description: _t("Return the results of a survey."),
  args: [arg("survey_id (number)", _t("The survey id"))],
  category: "Odoo",
  returns: ["RANGE"],
  compute: function (surveyId) {
    surveyId = toNumber(surveyId, this.locale);
    // TODO
  },
});
```

# Add a plugin

---

- Private **state**
- Introduce new **getters** to make parts of its state available
- Dispatch **commands**
- Some existing plugins:
  - **CellPlugin**
  - **ConditionalFormatPlugin**
  - **BordersPlugin**

## For our use case

- Acts as a proxy between the function and the data source
- Adds one getter **getSurveyResults**

```
/** @odoo-module */

import { registries, UIPlugin } from "@odoo/o-spreadsheet";

export class SurveyPlugin extends UIPlugin {
  static getters = ["getSurveyResults"];

  getSurveyResults(surveyId) {
    // TODO
    return 0;
  }
}

registries.featurePluginRegistry.add(
  "SurveyPlugin",
  SurveyPlugin,
);
```

# Our function

=ODOO.SURVEY(1) 

- How to fetch the data ?
  - With our new plugin **getter**  
**getSurveyResults**

```
/** @odoo-module */

registries.functionRegistry.add("ODOO.SURVEY", {
  // ...
  compute: function (surveyId) {
    surveyId = toNumber(surveyId);
    return this.getters.getSurveyResults(surveyId);
  },
});
```

# Fetch the data



- Function evaluation is **synchronous**
- Re-evaluate cells when data is fetched

```
/** @odoo-module */
```

```
import { registries, UIPlugin } from "@odoo/o-spreadsheet";  
import { ServerData } from  
"@spreadsheet/data_sources/server_data" ;
```

```
export class SurveyPlugin extends UIPlugin {  
    static getters = ["getSurveyResults"];  
  
    getSurveyResults (surveyId) {  
  
    }  
}
```

```
registries.featurePluginRegistry.add("SurveyPlugin",  
SurveyPlugin);
```

# Fetch the data



- Function evaluation is **synchronous**
- Re-evaluate cells when data is fetched
- Use **ServerData**
  - handles caching
  - notifies when data is fetched

```
/** @odoo-module */
```

```
import { registries, UIPlugin } from "@odoo/o-spreadsheet";
import { ServerData } from
"@spreadsheet/data_sources/server_data" ;

export class SurveyPlugin extends UIPlugin {
  static getters = ["getSurveyResults"];
  constructor(config) {
    super(config);
    const services = config.custom.env.services;
    this.serverData = new ServerData(services.orm, {
      whenDataIsFetched: () => this.dispatch("EVALUATE_CELLS"),
    });
  }

  getSurveyResults(surveyId) {
    const data = this.serverData.get(
      "survey.survey",
      "get_survey_results_for_spreadsheet",
      [surveyId]
    );
    return data;
  }
}
```

# Fetch the data



- Error cells

=ODOO.SURVEY(42)				
G	H	I	J	
	#ERROR	Error Survey 42 not available		

```
export class SurveyPlugin extends UIPlugin {  
  // ...  
  
  getSurveyResults(surveyId) {  
    const data = this.serverData.get(  
      "survey.survey",  
      "get_survey_results_for_spreadsheet",  
      [surveyId]  
    );  
    if (!data) {  
      throw new Error(  
        _t("Survey %s not available", surveyId)  
      );  
    }  
    return data;  
  }  
}
```

# Return the data



- can be
  - a single primitive value
  - a 2d matrix of primitive values
- Normal access rights apply

```
from odoo import models, _

class Survey(models.Model):
    _inherit = "survey.survey"

    def get_survey_results_for_spreadsheet(self):
        """Returns a 2d array of the results of the survey"""
        self.ensure_one()
        if not self.search([("id", "=", self.id)]):
            return False
        result = []
        all_answers = self.user_input_ids.user_input_line_ids
        data = self.question_ids._prepare_statistics(all_answers)
        for question_results in data:
            question = question_results["question"]
            answers = question_results["answer_line_ids"]
            column = (
                [question.title]
                + answers.mapped("display_name")
            )
            result.append(column)
        return result
```



*Demo*





# Performance pitfalls



## *Request at every evaluation*

- You don't control when cells are evaluated
- Cache http request results
  - Managed by `ServerData`

## *One request per cell*

- Batch requests



# Batch requests

```
this.serverData.batch.get(...);
```

## Input

- list of ids (*self*)
- list of arbitrary args (using `@api.model`)

## Output

- list of results
- Avoid `Exceptions`. The entire batch would fail

```
// client side
const data = this.serverData.get(
  "survey.survey",
  "get_survey_results_for_spreadsheet",
  [surveyId]
);

// BECOMES
const data = this.serverData.batch.get(
  "survey.survey",
  "get_survey_results_for_spreadsheet",
  surveyId
);

# server side
def get_survey_results_for_spreadsheet(self):
    return [
        survey._get_result_table()
        for survey in self
    ]

def _get_result_table(self):
    # same code we had before
```

# Other APIs

=WORD.EXIST("hello")

Exact same principles

- Any HTTP request
- Cache results
- Batch requests (if you can)
- Don't forget security concerns

```
/** @odoo-module */

import { _t } from "@web/core/l10n/translation";
import { registries, UIPlugin } from "@odoo/o-spreadsheet";

export class DictionaryPlugin extends UIPlugin {
    static getters = ["doesWordExist"];
    cache = {};

    doesWordExist(word) {
        if (this.cache[word] === undefined) {
            this.fetch(word);
        }
        switch (this.cache[word].status) {
            case "pending":
                throw new Error(_t("Loading..."));
            case "rejected":
                throw this.cache[word].error;
            case "fulfilled":
                return this.cache[word].found;
        }
    }
}

registries.featurePluginRegistry.add(
    "DictionaryPlugin", DictionaryPlugin
);
```

# Other APIs

=WORD.EXIST("hello")

Exact same principles

- Any HTTP request
- Cache results
- Batch requests (if you can)
- Don't forget security concerns

```
export class DictionaryPlugin extends UIPlugin {  
  // ...  
  fetch(word) {  
    // Mark the value as "pending" in the cache  
    this.cache[word] = { status: "pending" };  
    fetch(`https://api.toys/api/check_dictionary?text= ${word}`)  
      .then((response) => response.json())  
      .then((data) => {  
        // Update the cache with the result  
        this.cache[word] = {  
          status: "fulfilled",  
          found: data.found,  
        };  
      })  
      .catch((error) => {  
        // Update the cache with the error  
        this.cache[word] = {  
          status: "rejected",  
          error,  
        };  
      })  
      .finally(() => {  
        // Trigger a new evaluation when the data is loaded  
        this.dispatch("EVALUATE_CELLS");  
      });  
  }  
}
```

# Thank You!

Code available <https://github.com/LucasLefevre/spreadsheet-functions-oxp2023>

Documentation <https://github.com/odoo/o-spreadsheet>



#OdooExperience



/odoo



@odoo.official



@odoo