Gery Debongnie (GED) - RD Framework Team

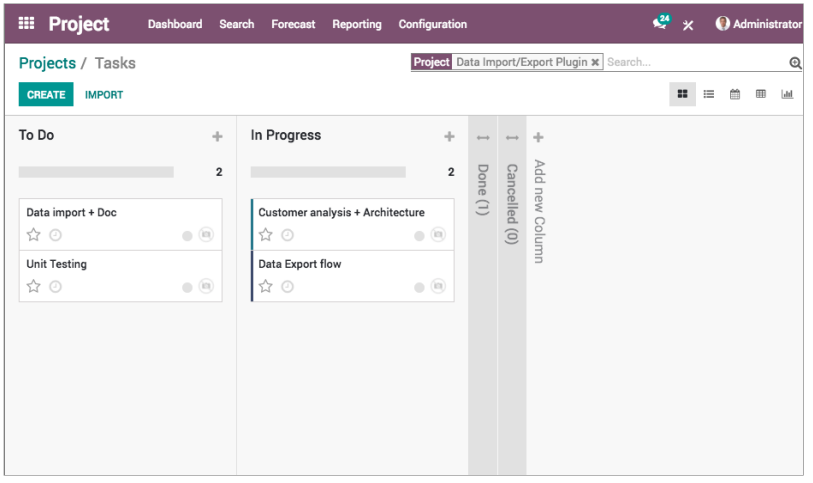
Introduction

What's new in v11?

Odoo JS in 2017:

Odoo JS in 2017:

We rewrote the javascript views...

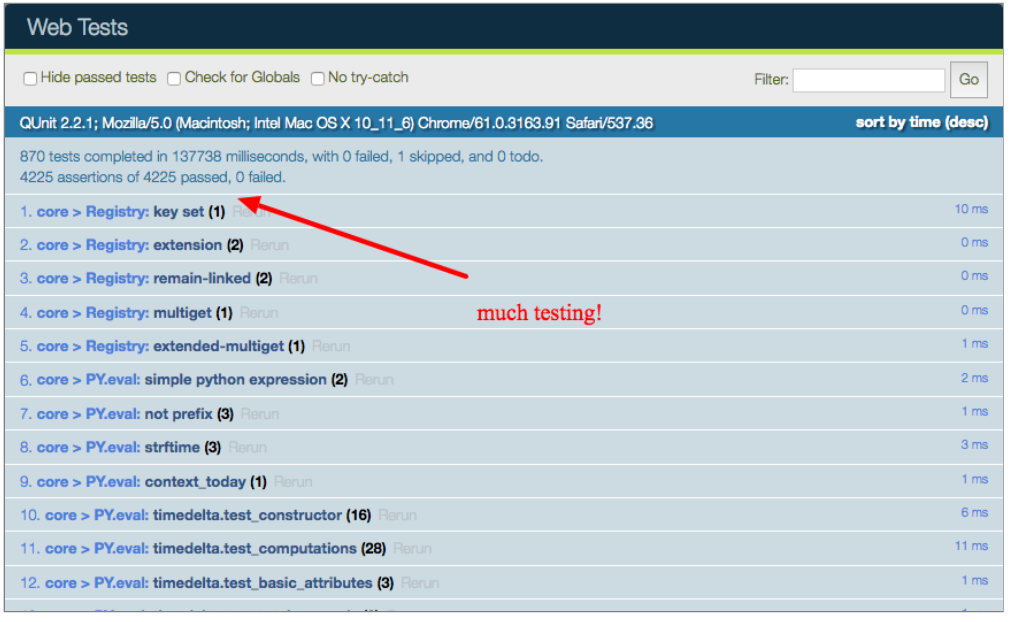


> Using SLOC to measure software progress is like using kg for measuring progress on aircraft manufacturing

Odoo v11 in a nutshell (JS)

	v10	v11
LOC	33.8k	32.8k
Comments	3.0k	12.5k
Tests LOC	3.9k	29.8k
Assets size (gzipped)	1042kb	909kb

Also New in v11: JS unit tests



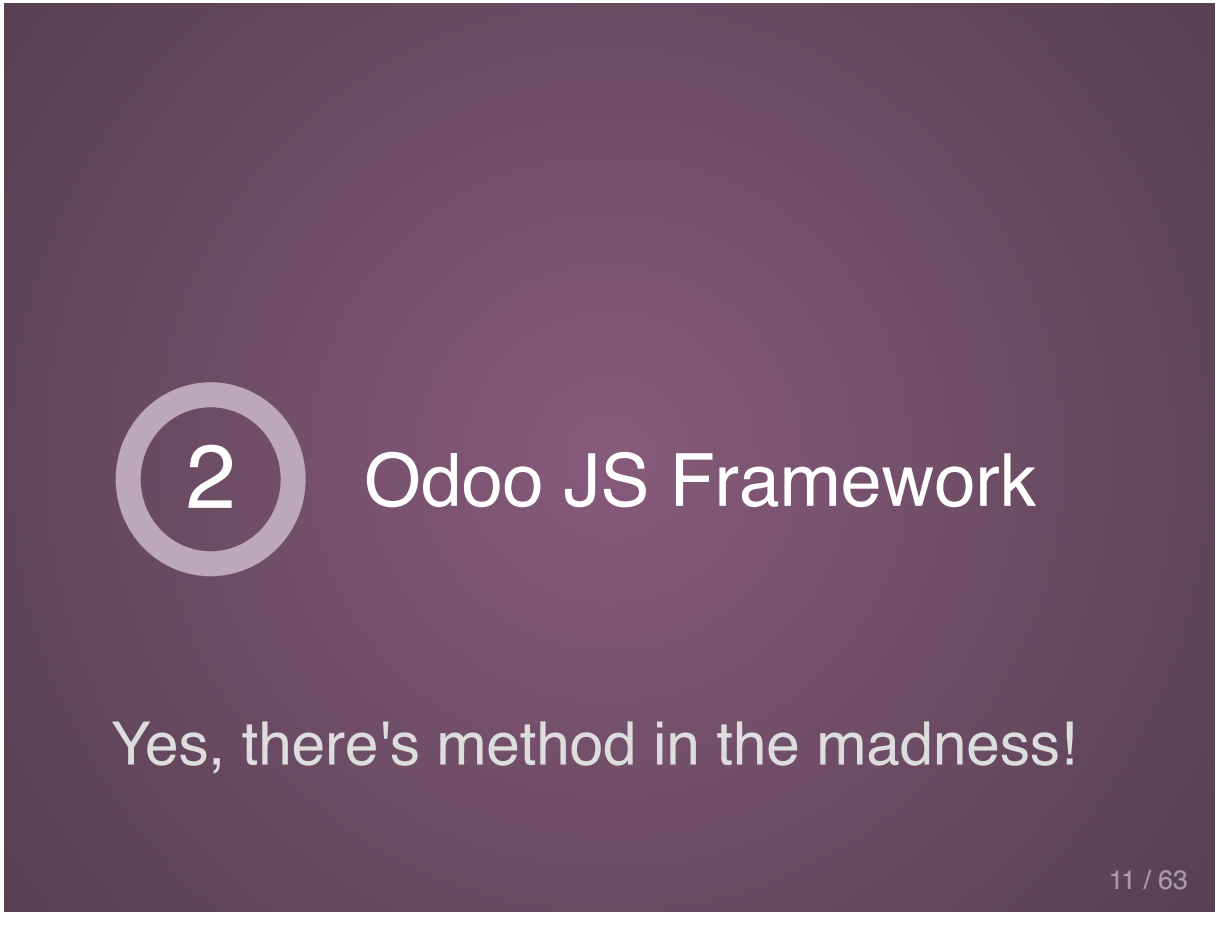
New in v11: various

- rpc: one method to rule them all
- lazy loading support
- class for manipulating domains/context
- better code organization, primitives for concurrency programming, ...
- depreciate dataset, Model

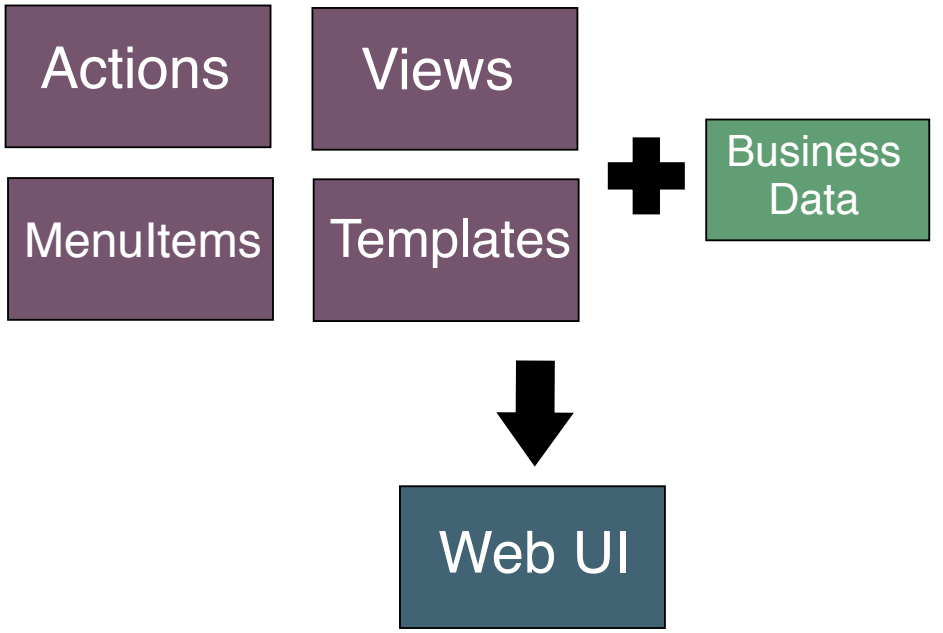
```
return this._rpc({
    model: 'res.partner',
    method: 'create',
    args: [data],
    context: context,
});
```

and also

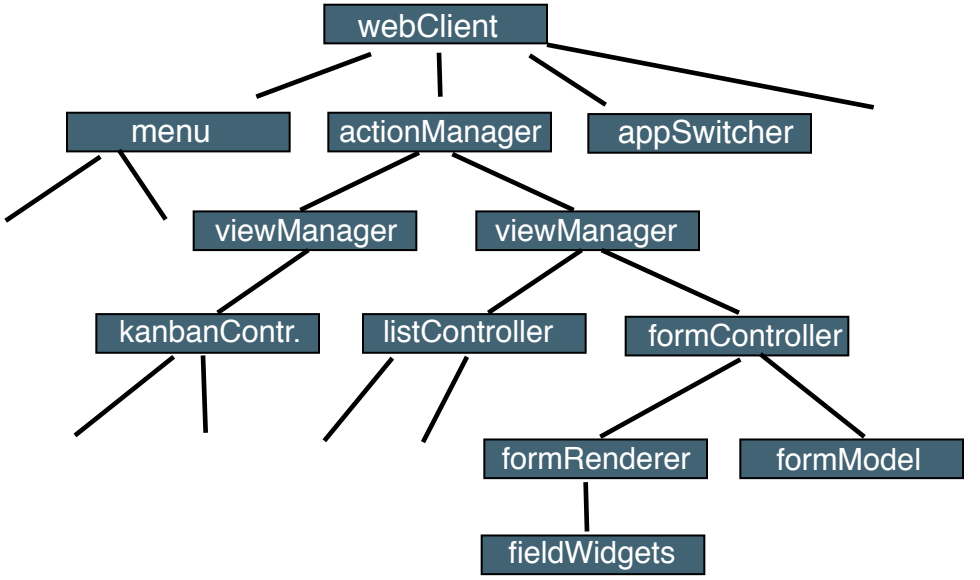
- more reusable widgets
- simpler code
- better communication between components
- more documentation
- coding guidelines
- ...


2
Odoo JS Framework
Yes, there's method in the madness!
11 / 63

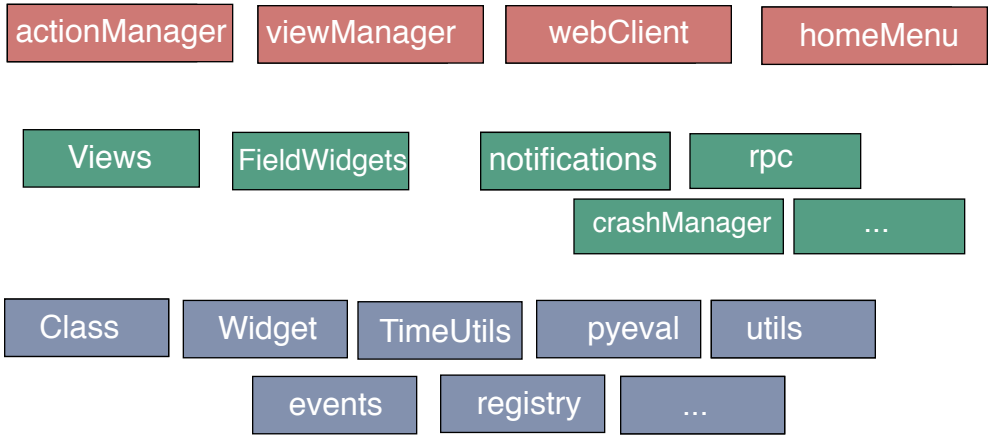
Web Client Architecture



Component Tree



Web Client Layers



Component Communication

Before:

main bus + reference hunting

```
var view_id = this.__parent.view.action_manager
                  .current_action.controller.view_id;
```

Component Communication

Before:

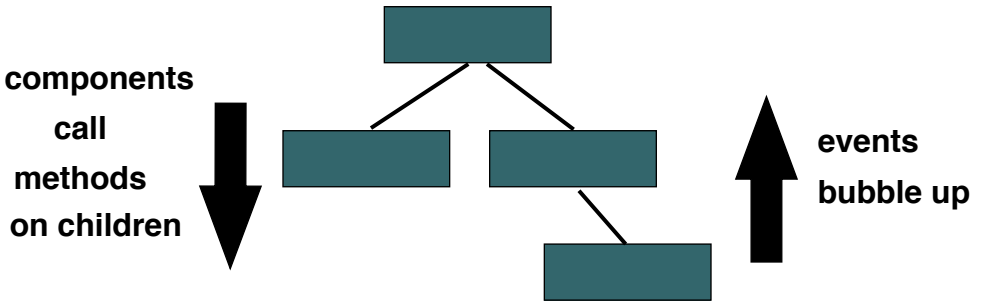
main bus + reference hunting

```
var view_id = this.__parent.view.action_manager
                  .current_action.controller.view_id;
```

Now and in the future:

- try to instantiate component with what it needs
- never fish for something in parent
- moving toward event bubbling up to communicate with environment

Communication




```
this.trigger_up('display_notification', {
    title: 'Move Zig',
    message: 'All your bases are belong to us',
});
```


3
New JS Views
18 / 63

> “No, I will not allow you to rewrite the web client from scratch just for fun...”

my manager
performance appraisal, 2016

New JS views

Full rewrite of JS views + 30k loc of unit tests

- Largest (framework) project in 2017

Thank you Team!!!

New JS views

Full rewrite of JS views + 30k loc of unit tests

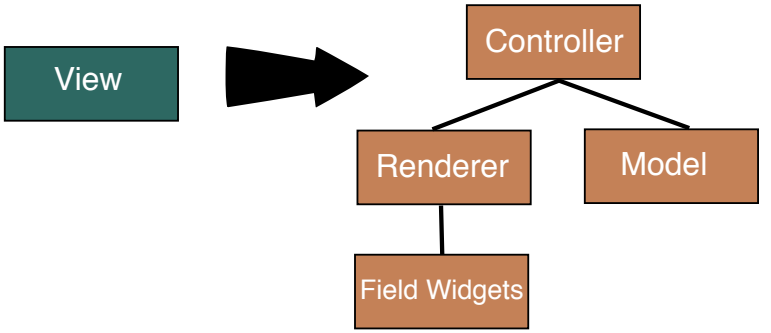
- Largest (framework) project in 2017

Thank you Team!!!

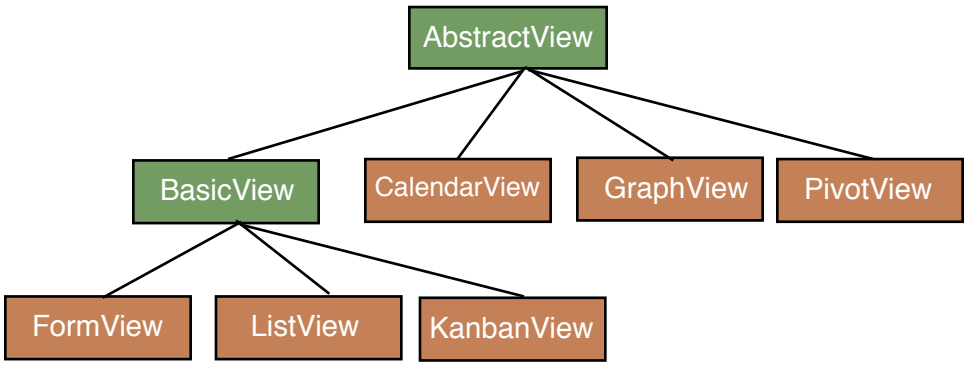
Motivations: reduce technical debt

- separate rendering from behaviour/model
- use views in other places
- use field widgets elsewhere
- unify behaviour across views
- ...

New Views: Overall Design



Class Hierarchy



Similar hierarchy for Controllers/Renderers/Models

BasicModel: Our Browser DB

- used by List, Form and Kanban
- about 3.7k LOC

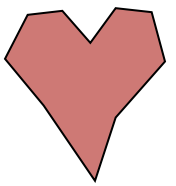
it handles data fetching, onchanges, groups of records, relational fields, sorting, safepoints, rollbacks, modifiers, x2many command generation, ...

BasicModel: Our Browser DB

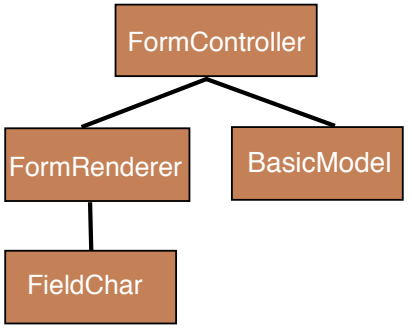
- used by List, Form and Kanban
- about 3.7k LOC

it handles data fetching, onchanges, groups of records, relational fields, sorting, safepoints, rollbacks, modifiers, x2many command generation, ...

Here is a demo

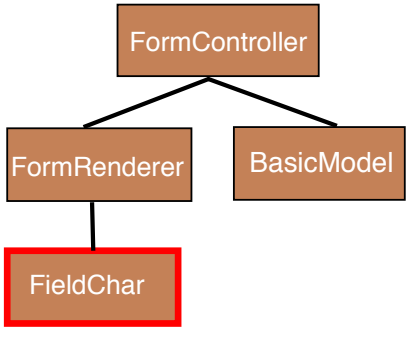
GED + BasicModel = 

Field Widgets: change example



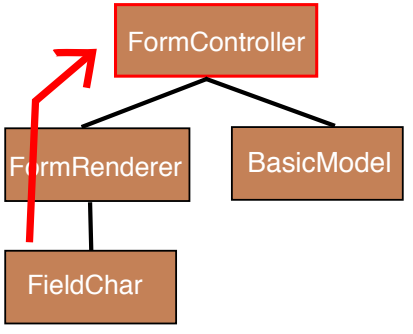
- Widget is rendered in edit mode

Field Widgets: change example



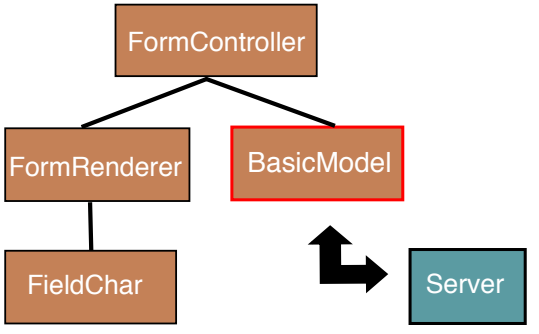
- Widget is rendered in edit mode
- user edit field value in input

Field Widgets: change example



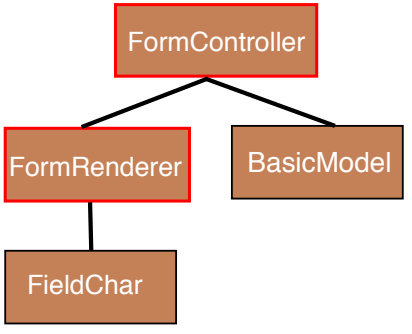
- Widget is rendered in edit mode
- user edit field value in input
- event is triggered, and bubble up to renderer

Field Widgets: change example



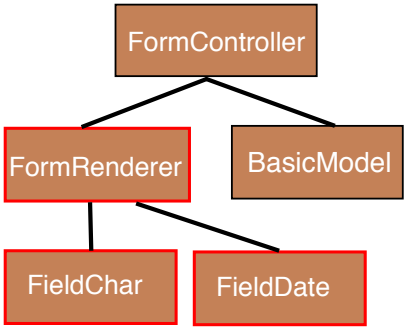
- Widget is rendered in edit mode
- user edit field value in input
- event is triggered, and bubble up to renderer
- controller notifies model, model goes to work

Field Widgets: change example



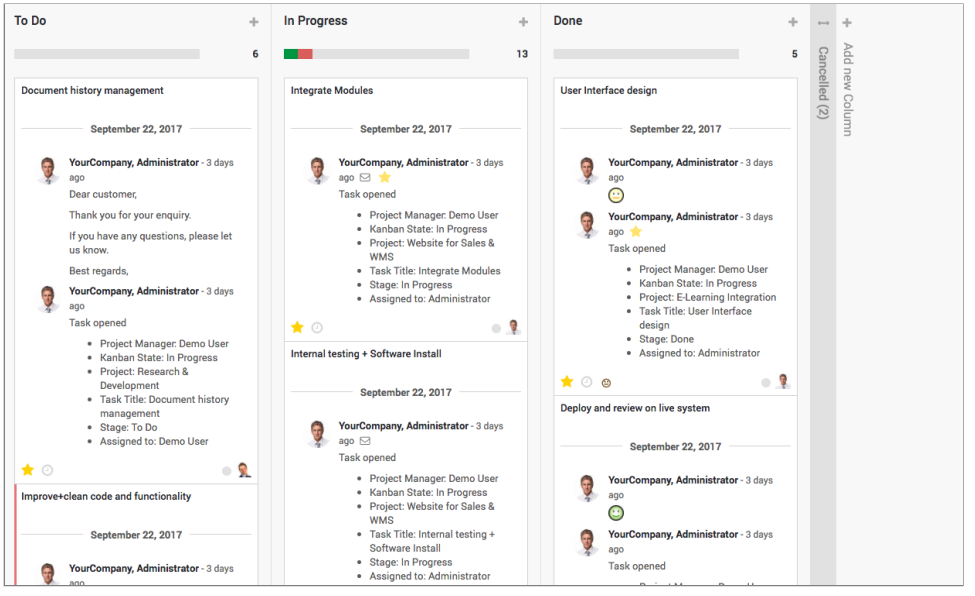
- Widget is rendered in edit mode
- user edit field value in input
- event is triggered, and bubble up to renderer
- controller notifies model, model goes to work
- controller confirms changes to renderer

Field Widgets: change example



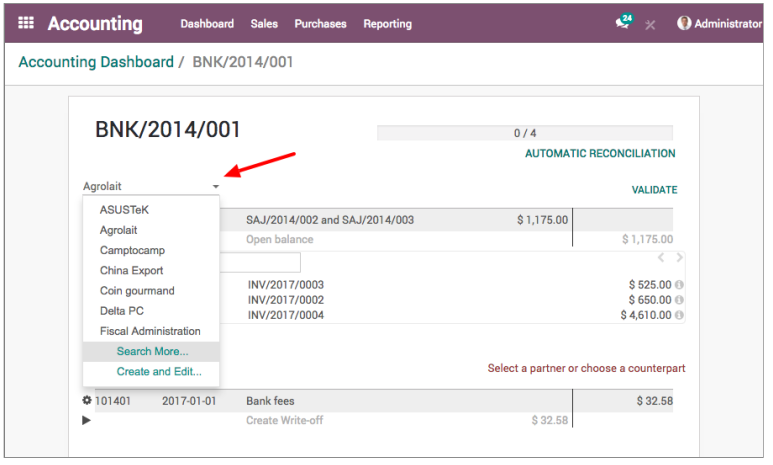
- Widget is rendered in edit mode
- user edit field value in input
- event is triggered, and bubble up to renderer
- controller notifies model, model goes to work
- controller confirm changes to renderer
- renderer updates (changed) fields

Just because we can doesn't mean we should

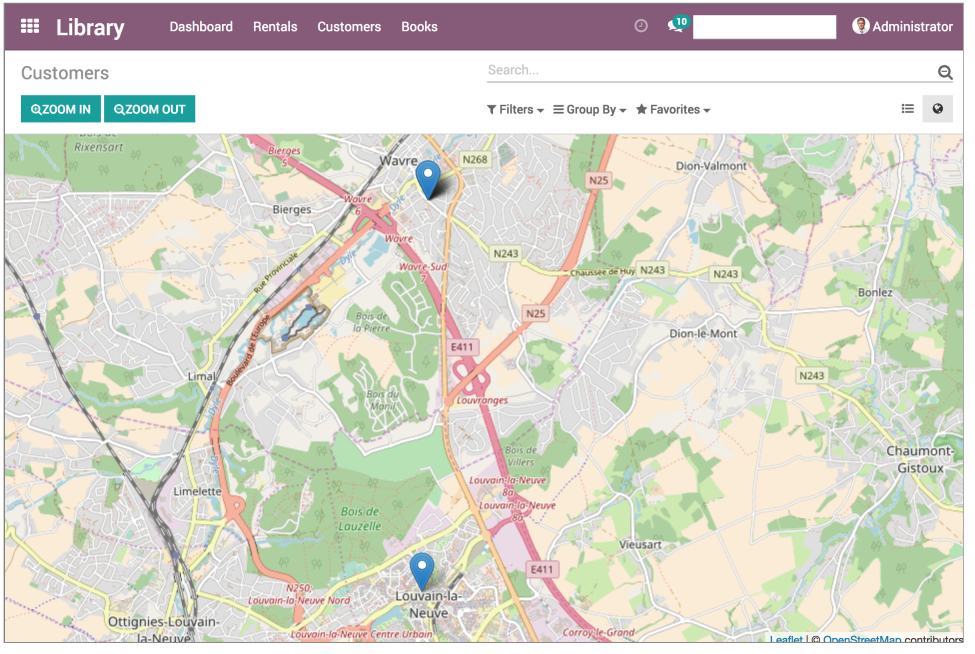


Views/Fields in the wild

- Views can now be instantiated anywhere
- same for fields



Proof of Concept: 128 LOC



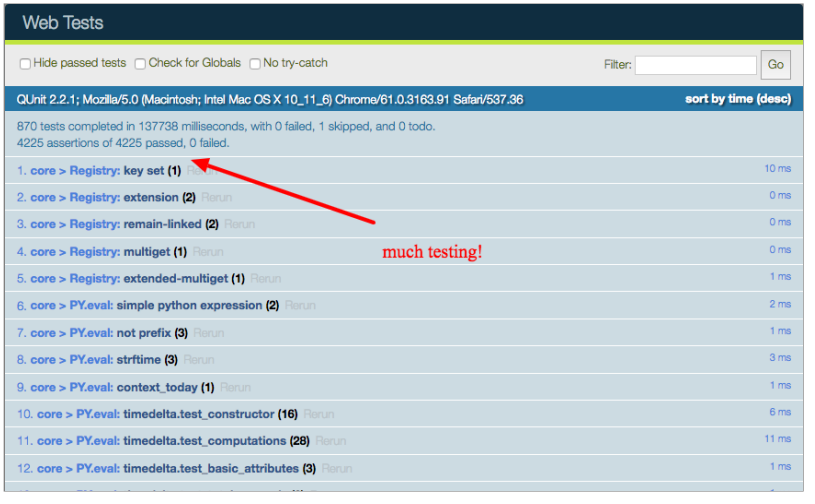
New Testing Framework

> “I don't like all these tests. That is way too much. It will slow you down.”

someone above me...

New Testing Framework

Route /web/tests



New Testing Framework

- new views were developped with TDD
- (almost) pure QUnitjs
- powerful test helpers to mock RPCs
- can test async code synchronously!

New Testing Framework

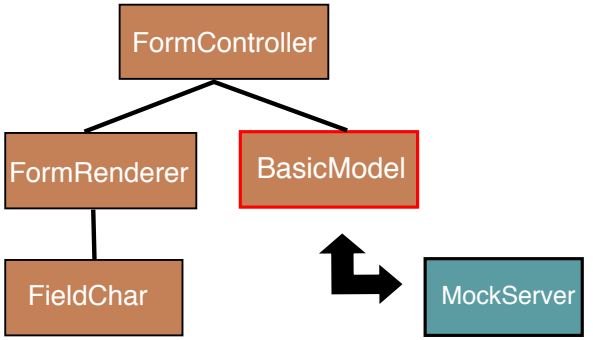
- new views were developped with TDD
- (almost) pure QUnitjs
- powerful test helpers to mock RPCs
- can test async code synchronously!

Benefits

- Executable specification
- much easier to accept PR
- much easier to refactor

Test Hero: MockServer

mock_server.js



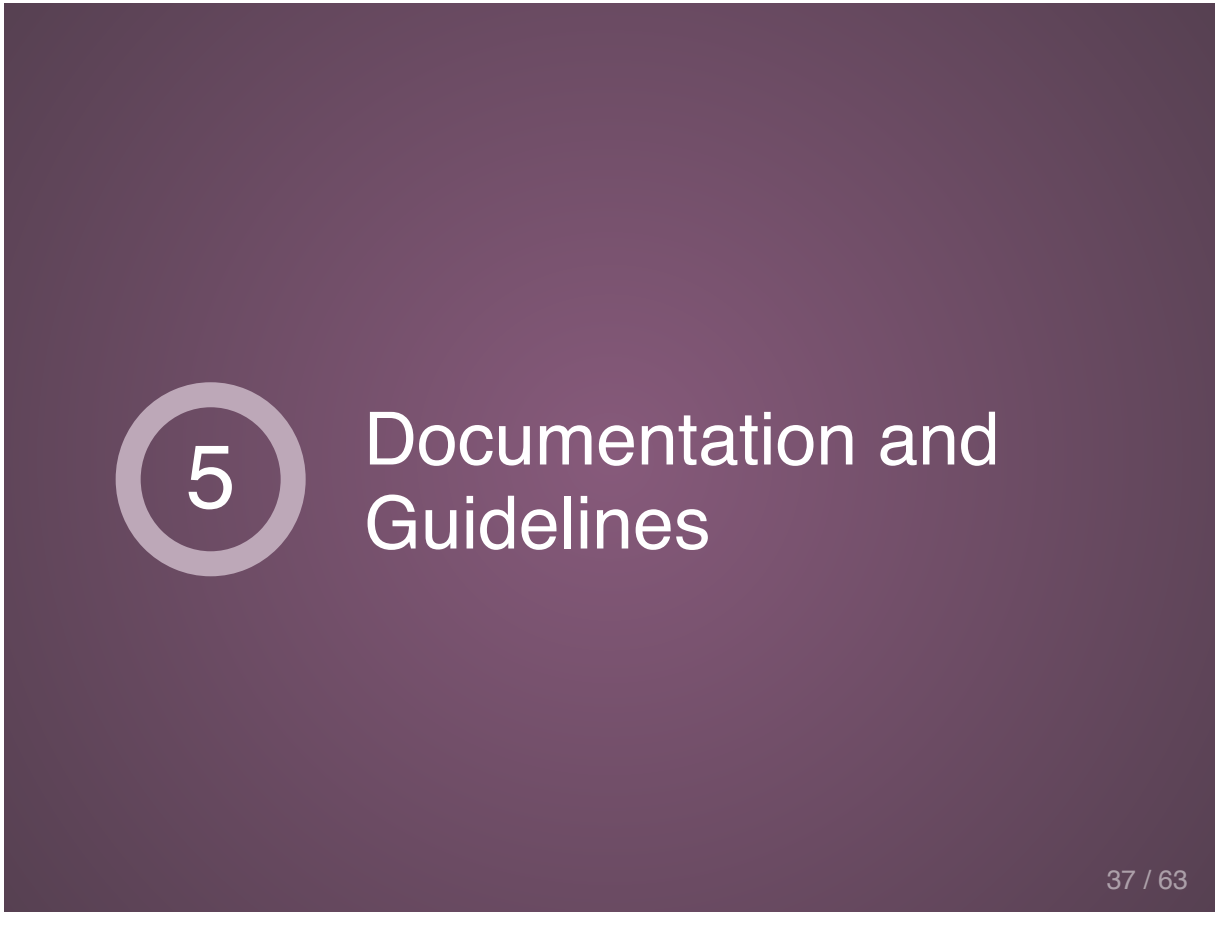
Can mock: copy, create, default_get, fields_get, name_get, name_create, name_search, onchange, read, read_group, search_count, search_read, unlink, write, ...

```
QUnit.test('no html injection in char field', function (assert) {
    assert.expect(1);

    var form = createView({
        View: FormView,
        model: 'partner',
        data: this.data,
        arch: '<form string="Partners">' +
                '<field name="foo"/>' +
            '</form>',
        res_id: 1,
        viewOptions: { mode: 'edit' },
    });

    form.$('input')
        .val('<script>throw Error();</script>')
        .trigger('input');
    form.$buttons.find('.o_form_button_save').click();
    assert.strictEqual(
        form.$('.o_field_widget').text(),
        '<script>throw Error();</script>',
        'the value should have been properly escaped'
    );

    form.destroy();
});
```


5
Documentation and Guidelines
37 / 63

Documentation and Guidelines

Bad news

Did not have time to update main documentation...

Documentation and Guidelines

Bad news

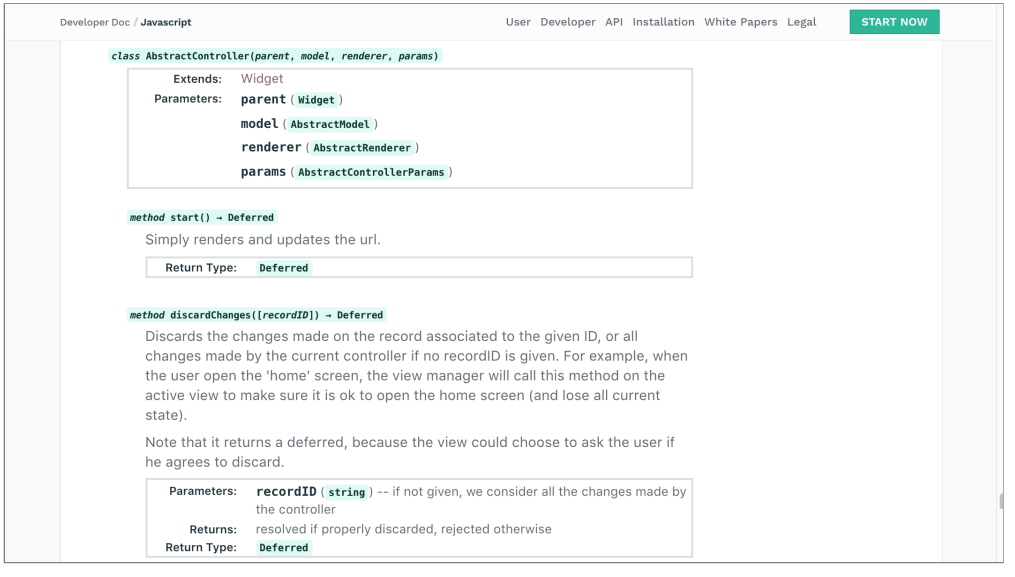
Did not have time to update main documentation...

Good news

- finally committed to document core classes
- coding guidelines
- uptodate doc online (extracted from docstring)

Uptodate html doc

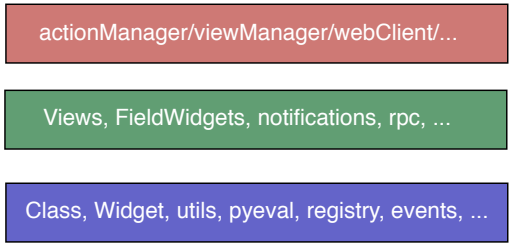
Starting in v11 => doc extracted from docstrings



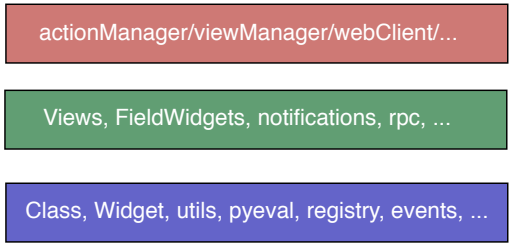
Coding Guidelines

6 The Future

Current State

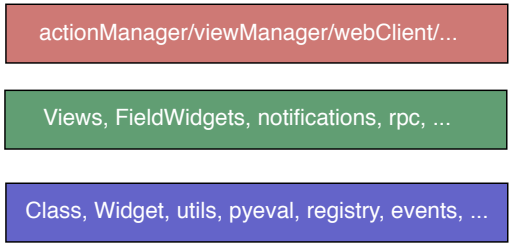


Current State



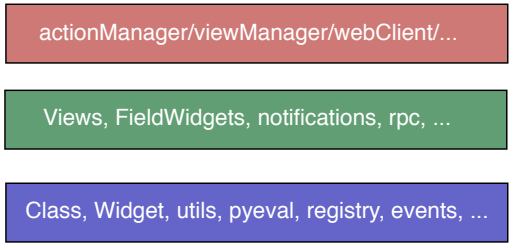
- bottom layer: mostly ok

Current State



- bottom layer: mostly ok
- medium layer: same

Current State



- bottom layer: mostly ok
- medium layer: same
- top layer: vastly too complex

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)
- solve the field widget model extension problem

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)
- solve the field widget model extension problem
- remove dataset and other old cruft

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)
- solve the field widget model extension problem
- remove dataset and other old cruft
- allow EcmaScript 2015

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)
- solve the field widget model extension problem
- remove dataset and other old cruft
- allow EcmaScript 2015
- improve error handling (logging/catching errors/recovery)

High priority

- massive simplification of top layer
- high quality developer documentation (JS)

Medium priority

- add didMount and willUnmount to Widget (remove do_show)
- simplify basicmodel (+ perf improvements)
- solve the field widget model extension problem
- remove dataset and other old cruft
- allow EcmaScript 2015
- improve error handling (logging/catching errors/recovery)
- ...

What makes a good framework?

What makes a good framework?

expressivity

What makes a good framework?

expressivity

easy to understand

What makes a good framework?

one obvious solutions for most problems

expressivity

easy to understand

API stability

What makes a good framework?

one obvious solutions for most problems

expressivity

easy to understand

API stability

What makes a good framework?

one obvious solutions for most problems

consistency

expressivity

easy to understand

API stability

What makes a good framework?

one obvious solutions for most problems

consistency

expressivity

easy to understand

Odoo v11 is much better!!!

Thank you for your attention...

Views: challenges/unsolved problems

- Field widget which needs to integrate with BasicModel
- Bad API when using field widgets standalone
- Optimizing RPCs/commands