OpenERP / odoo Connector

OpenDays 2014

Guewen Baconnier / Alexandre Fayolle

camptocamp

INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS

30 seconds to say...

- Documentation:

  http://www.openerp-connector.com

- Mailing list:

  https://launchpad.net/~openerp-connector-community

What is a connector?

- Connect odoo with external systems
- Exchange any data between them

What is not the connector?

- A middleware, it is a module inside odoo
- It does not listen external systems, it speaks to them

A framework

Does not do anything “out of the box”, rather:

- Suggests an implementation style
- Proposes bare “interface” classes
- Defines bricks
  - Jobs
  - Events
  - Backend / version mechanisms

Glossary

- Backend
- Event
- Job / jobs queue
- Worker
- Binding

Existing implementations

- Magento
- Prestashop
- Multicompany (odoo to odoo)
- Solr
- CMIS
- Connector E-Commerce (not a full implementation)

  See http://openerp-connector.com/ for links!

Find the doc

- http://www.openerp-connector.com
- Have a look at http://www.openerp-magento-connector.com (reference implementation, lots of great examples)

- Source code itself.

Mailing List

- Join the team on https://launchpad.net/~openerp-connector-community

- Shared for discussions about the connector and all the different implementations

How to install

- Just like any other add-on!
- Grab the source code

```
bzr branch lp:openerp-connector/7.0
```

- Add to `--addons-path`
- Install in odoo like any other add-on

Multiprocessing and Workers

- One odoo process: all is fine
- Several processes: at least one independent process must be launched for the processing of the jobs queue
  - Tip: use the provided `openerp-connector-worker` script

http://openerp-connector.com/guides/multiprocessing.html

Multiple databases and Workers

- Supported!

  Gotcha:

- Only runs on databases that are
  - in the option `database` in the command line
  - or `db_name` in the configuration file
  - if nothing is provided, will run on all databases

Design goals – Overview

- Do not impose a way of using the framework to the developer, only strong advices
- Propose bits of implementation, which can be used or not, and the developer can cherry pick
- The developer is in charge of choosing her path through the code, the code does not choose it for her
- Do not mix everything in `Model` objects

Design goals - Events

- Declare an event only once
- Consume it several times
  - → several actions can be triggered
- Different connectors can share events

Code example: Events

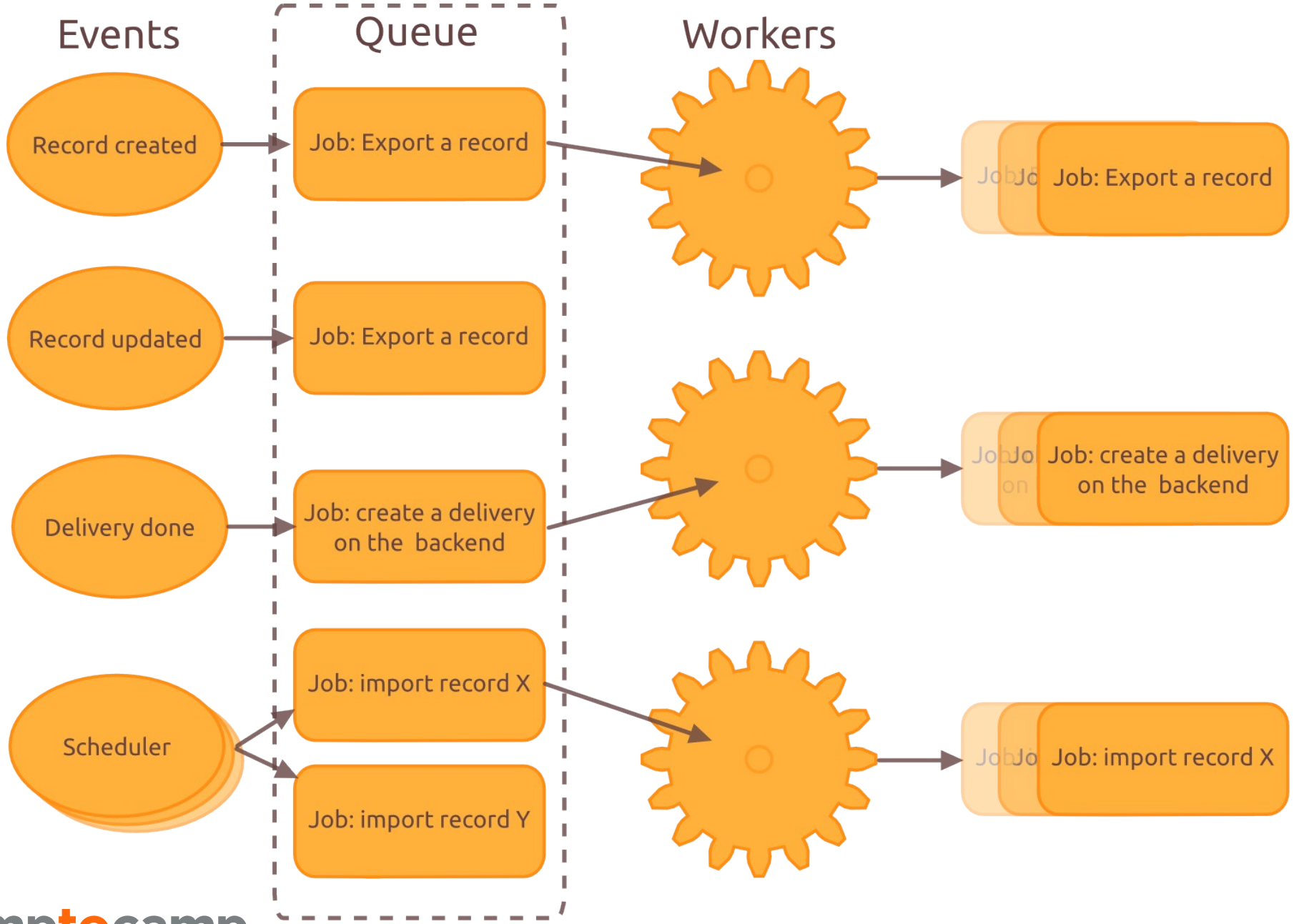
```
on_record_write = Event()
"""
``on_record_write`` is fired when one record has been updated.

Listeners should take the following arguments:

 * session: :py:class:`~connector.session.ConnectorSession` object
 * model_name: name of the model
 * record_id: id of the record
 * vals:  field values of the new record, e.g {'field_name': field_value, ...}

"""
```

```
@on_record_create(model_names='project.project')
@on_record_write(model_names='project.project')
def delay_export(session, model_name, record_id, vals):
    record = session.browse(model_name, record_id)
    if record.trello_sync:
        descr = _('Export project %s') % record.name
        consumer.delay_export(session, model_name, record_id, vals,
                              priority=4, description=descr)
```


Design goals - Jobs

- Reliable
- No batches!
- Run in the background in a separate process
- Overview of what is running, failing, waiting.

Overview - Jobs



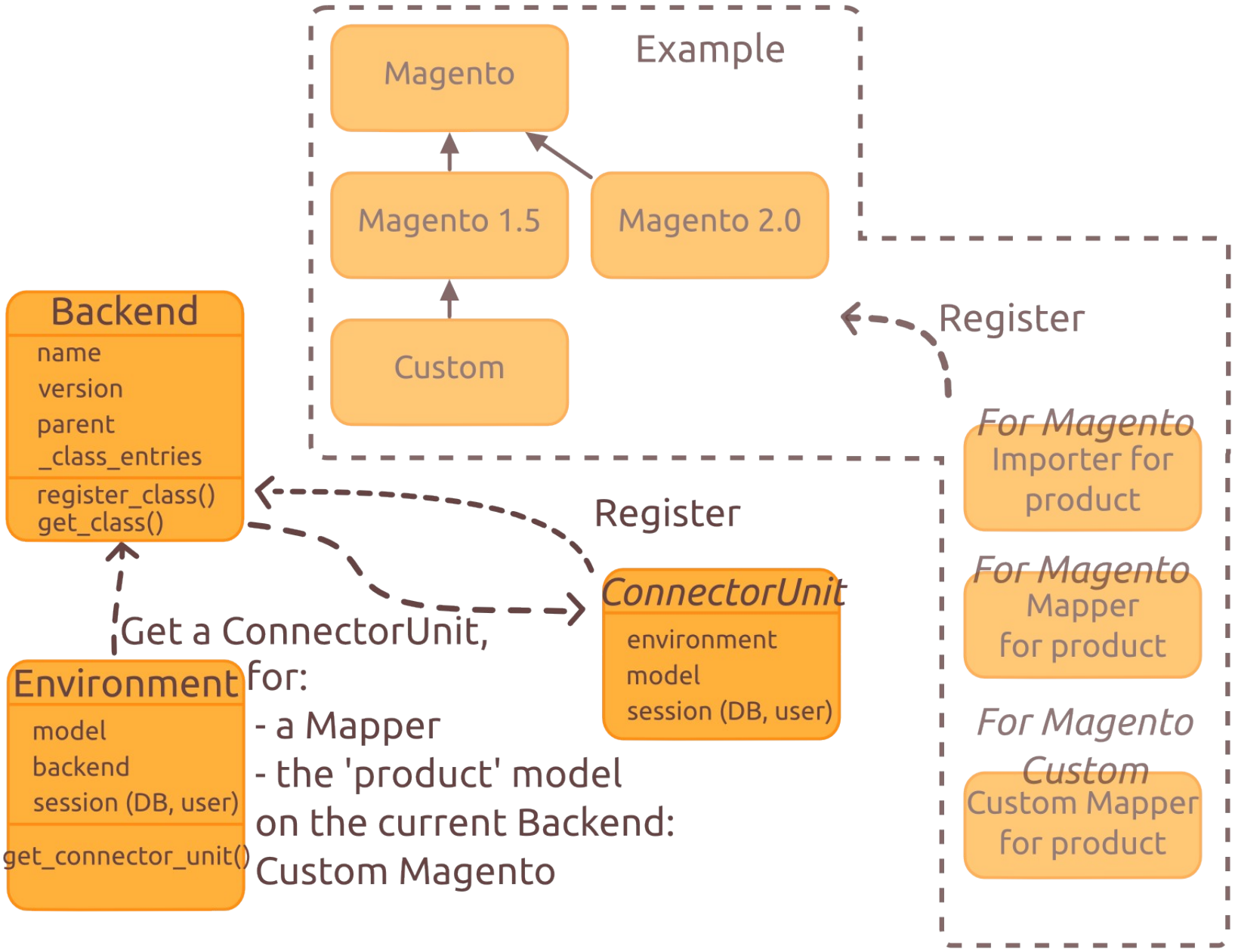
Code example: Jobs

```
@job
def export_record(session, model_name, binding_id, fields=None):
    """ Export a record on Trello """
    record = session.browse(model_name, binding_id)
    env = get_environment(session, model_name)
    exporter = env.get_connector_unit(TrelloExporter)
    return exporter.run(binding_id, fields=fields)


export_record(session, 'project.task', 4, ['name'])

export_record.delay(session, 'project.task', 4, ['name'])
```


Design goals - Backends

- Different behaviors across versions of a backend (Magento 1.7 or 2.0, or a customized Magento version...)
- Extensible

Overview - Backends



Code example: Backends

```
trello = Backend('trello', 'generic')
""" Generic Trello Backend """

trello_v2 = Backend(parent=trello, version="2.0")
""" Hypothetical V2 of the API Trello Backend """

trello_custom = Backend(parent=trello, version="custom")
""" Using custom ConnectorUnits for Trello Backend """
```


Code example: Backends

```
@trello
class ProjectAdapter(TrelloAdapter):
    _model_name = 'project.project'
    _prefix = '/boards/'


@trello_v2
class ProjectAdapter(TrelloAdapter):
    _model_name = 'project.project'
    _prefix = '/v2/boards/'
```


The glue: Environment and ConnectorSession

- `Environment` is the current working context:
  - `Model`
  - `Backend` (with version)
  - `ConnectorSession`
- `ConnectorSession` is a container for:
  - `Cursor`
  - `User`
  - `Context`

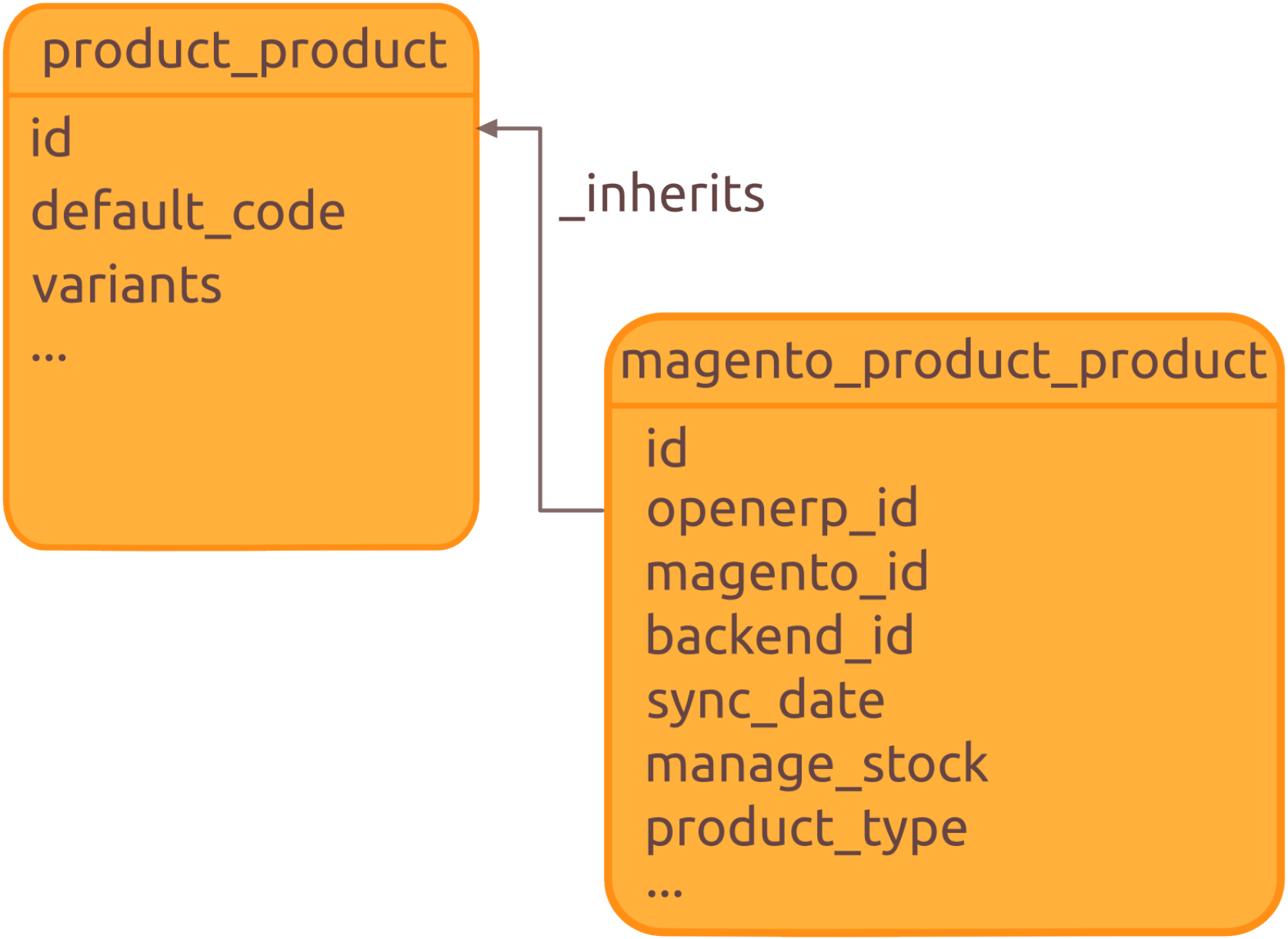
Design goals – Binding records

- Store the external / odoo couples of IDs
  - Plus synchronization date
  - And possibly extra data related only to **this** record and **this** backend
- If several backends are possible (e.g. Magento):

  → `_inherits` on the model
- If only one backend possible (e.g. Trello):

  → data may be added directly in the model

Example of binding record



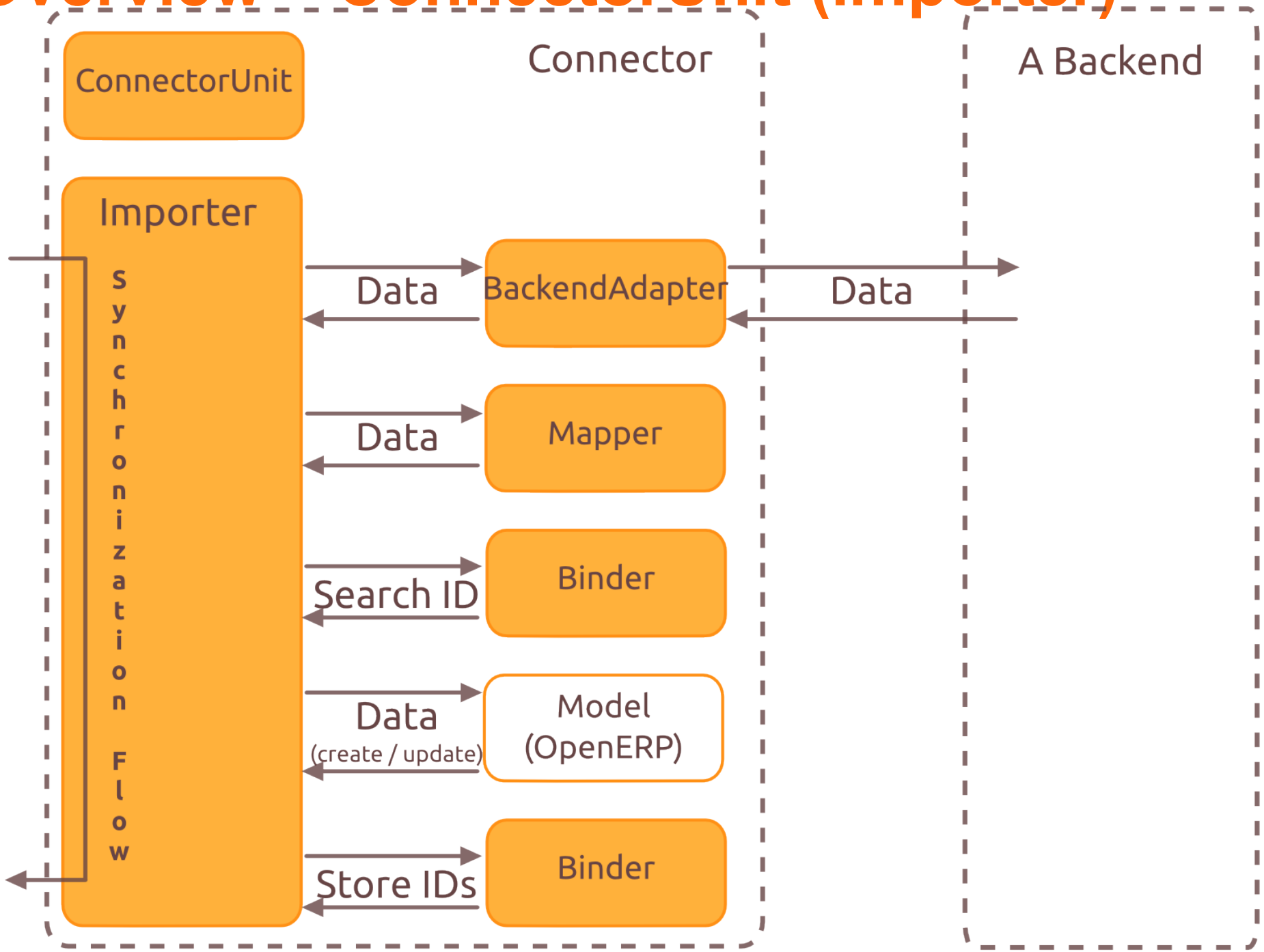
Design goals - ConnectorUnit

- Bare, decoupled, classes
- Registered with a backend
- 1 class: 1 responsibility
- Combined to do the synchronizations

ConnectorUnit – Synchronizer

- 2 kinds of synchronizers:
  - Importer
  - Exporter
- Manage the flow of the synchronization
  - Use all or part of the other `ConnectorUnit` classes to perform each synchronization step

Overview – ConnectorUnit (importer)



ConnectorUnit - Mapper

- Transform data
- Take data in, give data out

Code example: Mapper

```
@trello
class ProjectImportMapper(ImportMapper):
    _model_name = 'project.project'

    direct = [('name', 'name'),
              ('shortUrl', 'trello_short_url'),
              ('url', 'trello_url'),
              ]

@trello_custom
class CustomProjectImportMapper(ProjectImportMapper):
    _model_name = 'project.project'

    @mapping
    def another_mapping(self, record):
        color = record['color']
        return {'color': color.lower()}
```


Code example: using a Mapper in a Synchronizer

```
def _after_create(self, data, created):
    """ When a record is created on Trello, it returns the full json
    representation of the record, so we have the opportunity to save back
    some values.
    """
    try:
        import_mapper = self.get_connector_unit_for_model(ImportMapper)
    except NoConnectorUnitError:
        _logger.debug('No ConnectorUnit found for ImportMapper '
                      'for model %s', self.model._name)
    else:
        map_record = import_mapper.map_record(created)
        with self.session.change_context({'connector_no_export': True}):
            self.session.write(self.model._name,
                               self.binding_id,
                               map_record.values())
```


ConnectorUnit - Binder

- Know the external ID for an odoo ID (and conversely)
- Store the couple “external / odoo” IDs in binding records

Code example: Binder

```
def to_backend(self, binding_id):
    """ Give the external ID for an OpenERP ID

    :param binding_id: OpenERP ID for which we want the external id
    :return: backend identifier of the record
    """
    trello_record = self.session.read(self.model._name,
                                      binding_id,
                                      ['trello_id'])
    assert trello_record
    trello_id = trello_record['trello_id']
    if not trello_id:  # prefer None over False
        return None
    return trello_id
```


Code example: Binder bind method

```
def bind(self, external_id, binding_id):
    """ Create the link between an external ID and an OpenERP ID and
    update the last synchronization date.

    :param external_id: External ID to bind
    :param binding_id: OpenERP ID to bind
    :type binding_id: int
    """
    now_fmt = datetime.now().strftime(DEFAULT_SERVER_DATETIME_FORMAT)
    # avoid to trigger the export when we modify the `trello_id`
    with self.session.change_context({'connector_no_export': True}):
        self.session.write(self.model._name,
                           binding_id,
                           {'trello_id': str(external_id),
                            'trello_sync_date': now_fmt})
```


Code example: using a Binder in a Synchonizer

```
self.trello_id = self.binder.to_backend(self.binding_id)

result = self._run(*args, **kwargs)

if self.trello_id:
    self.binder.bind(self.trello_id, self.binding_id)
```


ConnectorUnit - BackendAdapter

- Communicate with the external system
- Speak the external language (REST, XML/RPC, …)
- But 1 interface within odoo
  - Typically CRUD
  - Often uses helper library (trollop, requests, magento, gdata...)

Code example: BackendAdapter

```
class TrelloAdapter(BackendAdapter):
    """ External Records Adapter for Trello """

    _prefix = None  # to define in subclasses

    def create(self, vals):
        record = self.trello.post(self._prefix, params=vals)
        return json.loads(record)
```


Code example: using a BackendAdapter in a Synchronizer

```
def _create(self, data):
    """ Create the Trello record """
    self._validate_data(data)
    return self.backend_adapter.create(data)
```


Tests

- Tests!
- External system should not be required to run the tests

  → **Mock** the external API

  - Examples in the Magento connector

Tips

- When developing, deactivate the cron tasks / worker process in charge of assigning and queueing the jobs
- Use `erppeek` to run jobs one by one:

```
model('queue.worker').assign_then_enqueue(1)
```

- Set a high priority on the job(s) you want to process first
  - high priority == low value

Demo: Trello connector

https://launchpad.net/openerp-trello-connector


tocamp
camptocamp
INNOVATIVE SOLUTIONS
BY OPEN SOURCE EXPERTS