# Tryton

## Développement d'application pour entreprise

# Qu'est-ce que Tryton ?

## Plate-forme applicative pour entreprise

## d'architecture trois tiers

# Qu'est-ce que Tryton ? (suite)

- Client PyGTK
- Serveur Python
- PostgreSQL (SQLite, MySQL)
- GPL-3
- Fork d'OpenERP

# Qu'est-ce que Tryton ? (suite)

- Achat / Vente
- Contacts
- Stock
- Comptabilité financière et analytique
- Projets
- Calendrier

# Pourquoi "forker" OpenERP

- Modifications non-acceptées
- Ajout vs redesign
- Qualité du code
- Mauvaise gestion des sources

# Objectifs de Tryton

- Consistances
- Migration
- Modularité
- Approche "Framework spécialisé"
- easy_install
- Code review

# Release process

- Trunk "releasable"
- Cycle de 6 mois
- 3 releases maintenues
- Serveur de tests

# Client

- PyGTK :
    - Léger (sans logique métier)
    - Multi-plateforme : X, Windows, MacOS X
- WebDAV
- CalDAV/CardDAV :
    - Thunderbird - Ligthning (SOGo)
    - iPhone
- XML-RPC
- JSON-RPC (futur GWT client)

# Serveur - Noyau

- ORM :
    - Création du schéma
    - Migration automatique
- Moteur de workflow
- Moteur de rapports (relatorio)
- Multi-langues

# Serveur - Noyau (suite)

- Sécurité :
    - SSL
    - Authentification
    - Règles d'accès :
        - par modèle
        - par enregistrement

# Serveur - Modules

- Modèles
- Wizard
- Vues
- Rapports
- Workflow
- Extension d'autres modules

# Neso

- Version standalone
- SQLite
- Process unique

# Exemple de développement

- Modèle
- Champs
- Valeurs par défaut
- Search-Create-Write-Delete
- Browse
- Vues
- États

# Modèles

```
1    from trytond.model import ModelSQL
2
3    class Opportunity(ModelSQL):
4        'Opportunity'
5        _description = __doc__
6        _name = 'training.opportunity'
7
8    Opportunity()
```

# Champs

```
 1    class Opportunity(ModelSQL, ModelView):
 2        'Opportunity'
 3        _description = __doc__
 4        _name = 'training.opportunity'
 5        _rec_name = 'description'
 6
 7        description = fields.Char('Description', required=True)
 8        start_date = fields.Date('Start Date', required=True)
 9        end_date = fields.Date('End Date')
10        party = fields.Many2One('party.party', 'Party', required=True)
11        comment = fields.Text('Comment')
12
13    Opportunity()
```

# Valeurs par défaut

```
1    class Opportunity(ModelSQL, ModelView):
2
3        start_date = fields.Date('Start Date', required=True)
4
5        def default_start_date(self):
6            return datetime.date.today()
7
8    Opportunity()
```

# Search-Create-Write-Delete

```
party_id = party_obj.search([
    ('name', '=', 'Bob'),
    ])

opportunity_id = opportunity_obj.create({
    'description': 'Bigdil',
    'party': party_id,
    })

opportunity_obj.write(opportunity_id, {
    'comment': 'Resto pas terrible',
    })

opportunity_obj.delete(opportunity_id)
```

# Browse

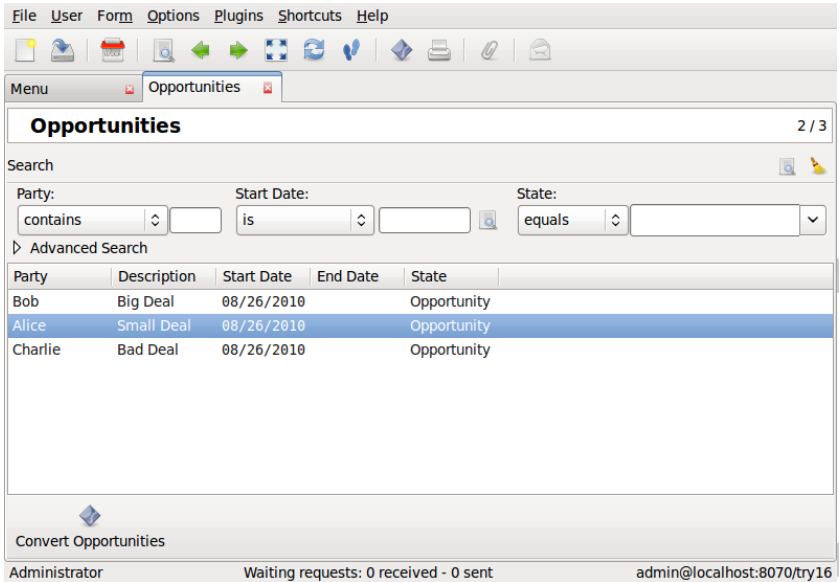
```
>>> opportunity = opportunity_obj.browse(opportunity_id)

>>> print opportunity.start_date
datetime.date(2010, 8, 27)

>>> for address in opportunity.party.addresses:
...     print address.street
Rue de Rotterdam
Chaussée de Namur
```

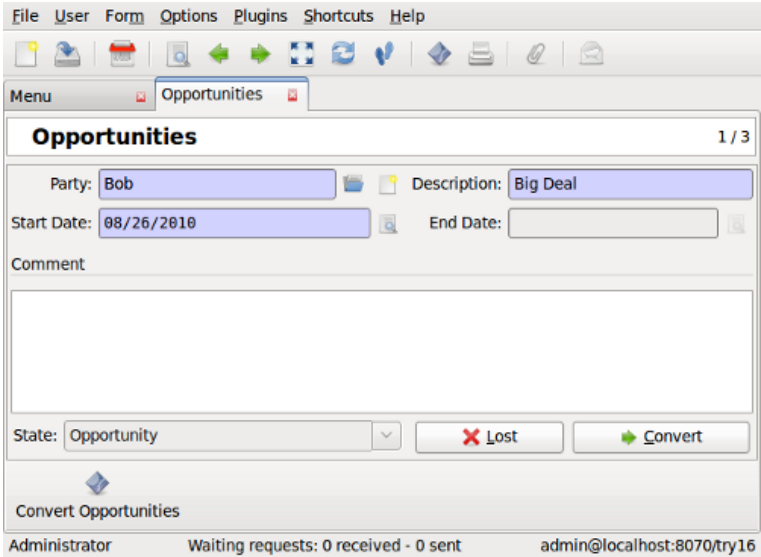
# Vue liste

```
 1    <record model="ir.ui.view" id="opportunity_view_tree">
 2        <field name="model">training.opportunity</field>
 3        <field name="type">tree</field>
 4        <field name="arch" type="xml">
 5            <![CDATA[
 6            <tree string="Opportunities">
 7                <field name="party" select="1"/>
 8                <field name="description" select="1"/>
 9                <field name="start_date" select="1"/>
10                <field name="end_date" select="2"/>
11            </tree>
12            ]]>
13        </field>
14    </record>
```



# Vue formulaire

```
 1    <record model="ir.ui.view" id="opportunity_view_form">
 2        <field name="model">training.opportunity</field>
 3        <field name="type">form</field>
 4        <field name="arch" type="xml">
 5            <![CDATA[
 6            <form string="Opportunity">
 7                <label name="party"/>
 8                <field name="party"/>
 9                <separator name="comment" colspan="4"/>
10                <field name="comment" colspan="4"/>
11            </form>
12            ]]>
13        </field>
14    </record>
```



# États

```
 1    from trytond.pyson import Not, Equal, Eval, In
 2
 3    class Opportunity(ModelSQL, ModelView):
 4
 5        state = fields.Selection([
 6                ('opportunity', 'Opportunity'),
 7                ('converted', 'Converted'),
 8                ('lost', 'Lost'),
 9            ], 'State', required=True, readonly=True, sort=False)
10
11        description = fields.Char('Description', required=True, states={
12            'readonly': Not(Equal(Eval('state'), 'opportunity')),
13            }, depends=['state'])
```

# Proteus

- CLI Python
- Active record
- Local et réseau

# Futur

- Process vs Thread
- namedtuple vs dict
- IMAP
- Gantt chart
- Widget de recherche

# Futur (suite)

```
 1    class Opportunity(ModelSQL, ModelView):
 2
 3        @classmethod
 4        def create(cls, values, **kwargs):
 5            pass
 6
 7        @classmethod
 8        def default_start_date(cls):
 9            return datetime.date.today()
10
11    opportunity = Opportunity(opportunity_id)
12    isinstance(opportunity, Opportunity)
```

# More

- Website: http://www.tryton.org/
- IRC: #tryton on irc.freenode.net

```
22:57 < zodman> dudes tryton development is easy !!!!
22:57 < zodman> xD
22:57 < zodman> good work!
```