

Dedomenon presentation

Raphaël Bauduin

8 february 2009 - FOSDEM

Introduction

Origins

Features

Foundations

Data Structure

Concepts

Customisation

Data types

Localisation

Rails engines

Openness

REST API

Export facilities

Open Source

Future

Development

Project

Enf of presentation

Questions?

Origins

- ▶ experiment started march in 2005, after FOSDEM, with the goals to have a web administered database system with
 - ▶ extreme flexibility

- ▶ first available in march 2006 (always after FOSDEM)
- ▶ joined forces with Zeropoint.IT in november 2007
- ▶ publication of source code under an OSI license in july 2008

Origins

- ▶ experiment started march in 2005, after FOSDEM, with the goals to have a web administered database system with
 - ▶ extreme flexibility
 - ▶ **easy extensibility**

- ▶ first available in march 2006 (always after FOSDEM)
- ▶ joined forces with Zeropoint.IT in november 2007
- ▶ publication of source code under an OSI license in july 2008

Origins

- ▶ experiment started march in 2005, after FOSDEM, with the goals to have a web administered database system with
 - ▶ extreme flexibility
 - ▶ easy extensibility
 - ▶ **complex data types**
- ▶ first available in march 2006 (always after FOSDEM)
- ▶ joined forces with Zeropoint.IT in november 2007
- ▶ publication of source code under an OSI license in july 2008

Origins

- ▶ experiment started march in 2005, after FOSDEM, with the goals to have a web administered database system with
 - ▶ extreme flexibility
 - ▶ easy extensibility
 - ▶ complex data types
 - ▶ **easy administration**
- ▶ first available in march 2006 (always after FOSDEM)
- ▶ joined forces with Zeropoint.IT in november 2007
- ▶ publication of source code under an OSI license in july 2008

Origins

- ▶ experiment started march in 2005, after FOSDEM, with the goals to have a web administered database system with
 - ▶ extreme flexibility
 - ▶ easy extensibility
 - ▶ complex data types
 - ▶ easy administration
 - ▶ **localised user interface**
- ▶ first available in march 2006 (always after FOSDEM)
- ▶ joined forces with Zeropoint.IT in november 2007
- ▶ publication of source code under an OSI license in july 2008

So, *what* is Dedomenon

Web database providing

- ▶ web-based management and end-users tools
- ▶ multi-lingual
- ▶ UTF-8 support at all levels
- ▶ easy information management: no need to create primary keys or tables to store Many to Many relations.
- ▶ advanced data types such as file attachments, URLs, email addresses, . . .
- ▶ export to a standard RDBMS
- ▶ a complete API to integrate the data store in your applications

Built on Free and Open Source

Postgresql is the RDBMS storing all the data

Ruby on Rails is the web application framework used

YUI is the javascript framework we chose for the web interface

Underlying database

Definition zone

entities ↔ table

details ↔ column

relations ↔ constraint or
many to many
relation table

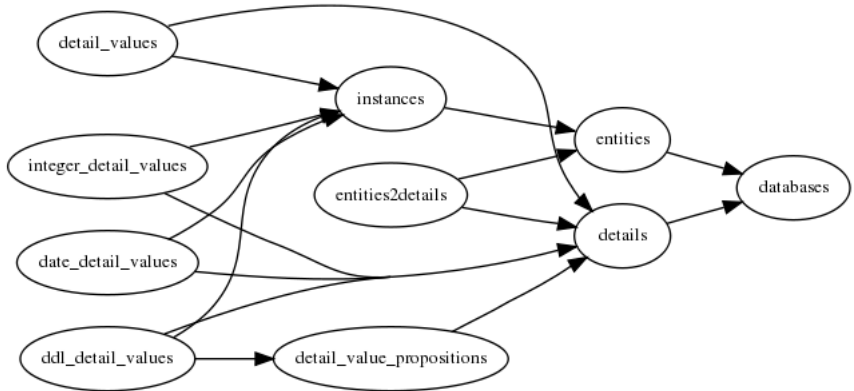
Instanciation zone

instances ↔ record

detail_values ↔ column's value

links ↔ record in a many
to many relation
table

Tables relations



Example

id	name	first_name	age	country_id
1	Torvalds	Linus	38	34

Example

id	name	first_name	age	country_id
1	Torvalds	Linus	38	34

The definition of the table is stored in tables:

- ▶ entities
- ▶ details
- ▶ data types
- ▶ relations

Example

id	name	first_name	age	country_id
1	Torvalds	Linus	38	34

The definition of the table is stored in tables:

- ▶ entities
- ▶ details
- ▶ data types
- ▶ relations

And entries are stored in other tables:

- ▶ instances
- ▶ detail values (1 per type: date, integer, ...)
- ▶ detail value propositions
- ▶ links

Postgresql to the rescue

tablefunc : contrib module reassembling virtual tables split over multiple tables

Postgresql to the rescue

tablefunc : contrib module reassembling virtual tables split over multiple tables

- ▶ Used to display tables in the web interface
- ▶ Unusable in the code for the complete REST API

Advantages

- ▶ Consistency: same approach, whatever the datatype
- ▶ Flexibility: easy to get at each data element individually
- ▶ Extensibility

Data types

Defining a new data type is writing a Ruby class inheriting
DetailValue and implementing

- ▶ `to_form_row`
- ▶ `self.format_detail`
- ▶ `self.valid?`

Data types

Defining a new data type is writing a Ruby class inheriting `DetailValue` and implementing

- ▶ `to_form_row`
- ▶ `self.format_detail`
- ▶ `self.valid?`

Data types can be very simple: `date_detail_value`

Data types

Defining a new data type is writing a Ruby class inheriting `DetailValue` and implementing

- ▶ `to_form_row`
- ▶ `self.format_detail`
- ▶ `self.valid?`

Data types can be very simple: `date_detail_value` or more complex: store a file on S3

You can also overwrite all methods provided by `ActiveRecord`, as is done in `ddl_detail_value` to implement the “choice in a list” data type

LongTextDetailValue



```
class LongTextDetailValue < DetailValue
  def to_form_row(i=0,o={})
    %Q{
      <input type="hidden"
        id="#{o[:entity].name}_#{detail.name}[#{i.to_s}]_id"
        name="#{detail.name}[#{i.to_s}][id]" value="#{id}">
      <tr><td>#{detail.name }:</td>
        <td>
          <textarea cols="30" rows="10"
            name="#{detail.name+"["+i.to_s+"]"}[value]">
            #{html_escape(value)}
          </textarea>
        </td>
      </tr>}
    end
  def self.format_detail(options)
    return html_escape(options[:value])
  end
end
```

EmailDetailValue








```
def self.format_detail(options)
  return "" if options[:value].nil?
  options[:format]=:html if options[:format].nil?
  case options[:format]
  when :html
    return %Q{<a href="mailto:#{options[:value]}">
      #{html_escape(options[:value])}
    </a> }
  else
    return options[:value]
  end
end

def self.valid?(value, o)
  if value.nil? or value==" " or
    value.match(/^[_\w-]+(\. [_\w-]+)*@[_\w-]+(\. \w+)*(\. [a-z]{2,3})$/)
    return true
  end
end
```

List formatting shot

Filter on field with value corresponding to  

Page: (100 items)

Name	Website	Email	View
Dryades	http://www.ylu.net	info@ylu.net	
Orthrus	http://www.Gourmetstores.Com	info@Gourmetstores.Com	
Zetes	http://www.Basketsbypatty.Com	info@Basketsbypatty.Com	
Nereids	http://www.vtj.net	info@vtj.net	
Meliads	http://www.Diamondengagmentring.Com	info@Diamondengagmentring.Com	
Lakhesis	http://www.Prosecutes.Com	info@Prosecutes.Com	
Oreades	http://www.shoppingjewelry.com	info@shoppingjewelry.com	

Form row formatting shot

The image shows two examples of form row formatting. The first example is a form with three rows: 'website:' with a text input, 'pays:' with a dropdown menu showing 'belgique', and 'email:' with a text input. Below these rows are 'Submit' and 'Cancel' buttons. The second example is a single row with 'file:' followed by a text input and a 'Browse...' button.

DropDownList

```
def to_form_row(i=0, o={})
  propositions = detail.detail_value_propositions
  %Q{
    <input type="hidden" id="#{o[:entity].name}_#{detail.name}#{i.to_s}
      name="#{detail.name}#{i.to_s}[id]" value="#{self.id}">
    <tr><td>#{detail.name }:</td>
      <td>
        <select name="#{detail.name+"["+i.to_s+"]"}[value]">
          #{ propositions.inject("") do |r,p|
r<< %Q{<option #{(detail_value_proposition_id==p.id)?'selected="selected"
          value="#{p.id}">#{p.value}</option>}
          end
        } </select></td></tr>}
  end
end
```

Dedomenon habla tu lengua

We use the Rails I18n

Translating the UI is done in one file.

To make your own, take the english to start from.

Beyond customisation

- ▶ Rails engine go further than normal plugins.
- ▶ You can add completely new functionalities to Dedomenon
- ▶ We build a proof of concept complete REST API as an engine!

Future

Rails 2.3 seems very promising as it will integrate parts of engines capabilities.

Get to your data

Work in progress nearing completion!

- ▶ Access your data from other applications
- ▶ We start with JSON but XML, yaml are possible
- ▶ built as an engine

Get to your data

Work in progress nearing completion!

- ▶ Access your data from other applications
- ▶ We start with JSON but XML, yaml are possible
- ▶ built as an engine

TheDataTank.com

This is the base of a service we will provide to ease the use of commonly useful data. Our first published database will be of belgian postal codes.

Freedom to move

In validation phase

- ▶ Export your data to standard RDBMS
- ▶ Export all data types and data schema
- ▶ Initial support of Postgresql
- ▶ Other export added easily: postgres specific code 80 lines long!

Freedom to code

- ▶ Project hosted at <http://www.dedomenon.org>
- ▶ Code managed with Git
- ▶ Mailing list at <http://lists.dedomenon.org>
- ▶ Platform with bug tracking, source code browsing, documentation, etc. . .

Build on the foundation

- ▶ New data types
- ▶ UI revamp
- ▶ Right-to-left interface
- ▶ Advanced reporting
- ▶ Import functionality
- ▶ Notifications
- ▶ Authentication schemes

Build a community

- ▶ We want to build a community
- ▶ We're listening to feedback
- ▶ Contributions are welcome

Let's start the discussion

Raphaël Bauduin rb@dedomenon.org
<http://www.dedomenon.org>