CS 683 Emerging Technologies
Fall Semester, 2006
Doc 20 Rails 4 Active Record
Nov 2, 2006

References

Agile Web Development with Rails 2nd Ed Bl.16 October 25, Thomas & Hanson, The Pragmatic Bookshelf, PDF

Rails API, http://api.rubyonrails.org/

Documentation
A Journey

What are the Methods & Parameters?

```
class CreateBooks < ActiveRecord::Migration
  def self.up
    create_table :books :options => '' do |t|
      t.column :title, :string
      t.column :author, :string
      t.column :date, :date
    end
  end

end
```

ActiveRecord::Migration Methods

announce
method_missing
migrate
say
say_with_time
suppress_messages
write

Migration Transformations

http://api.rubyonrails.org/classes/ActiveRecord/Migration.html

These are not methods in Migration Class

create_table(name, options)
drop_table(name): Drops the table called name.
rename_table(old_name, new_name)
add_column(table_name, column_name, type, options)
rename_column(table_name, column_name, new_column_name)
change_column(table_name, column_name, type, options):
remove_column(table_name, column_name)
add_index(table_name, column_names, index_type, index_name)
remove_index(table_name, index_name)

method_missing

Method in Object

Sent to an object when is it sent a message it does not have

Migrations forwards messages to an instance of

ActiveRecord::ConnectionAdapters::AbstractAdapter

ActiveRecord::ConnectionAdapters::AbstractAdapter

Includes Modules

Quoting
DatabaseStatements
SchemaStatements

SchemaStatements Methods

http://api.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/SchemaStatements.html

add_column
add_index
change_column
change_column_default
columns
create_table
drop_table
initialize_schema_information
native_database_types
remove_column
remove_index
rename_column
rename_table
structure_dump
table_alias_for
table_alias_length

TableDefinition

http://api.rubyonrails.org/classes/ActiveRecord/ConnectionAdapters/TableDefinition.html

SchemaStatements documentation sends us to TableDefinition

column() method documentation lists:

- types
- options

Type list is not the same as given in Migrations documentation

Active Record Details

Converting Types

SQL Type	Ruby Type
int, integer	Fixnum
decimal, numeric	BigDecimal
interval, date	Date
clob, blob, text	String
float, double	Float
char, varchar, string	String
datetime, time	Time
boolean	special handling needed

Where is this documented?
Source code does not seem to cover decimal

Binary Values

Not all databases support binary types

Ruby evaluates

- nil and false to false
- All other values to true

MySQL adapter maps boolean to tinyint

Naming Convention for Binary methods

```
rails = Book.find_by_name("Rails for Dummies")
if rails.isPublished
    puts "uses Ruby boolean conventions for true false"

if rails.isPublished?
    puts "false if isPublished field has value 0, "0", "f", "false", "", nil, false
```

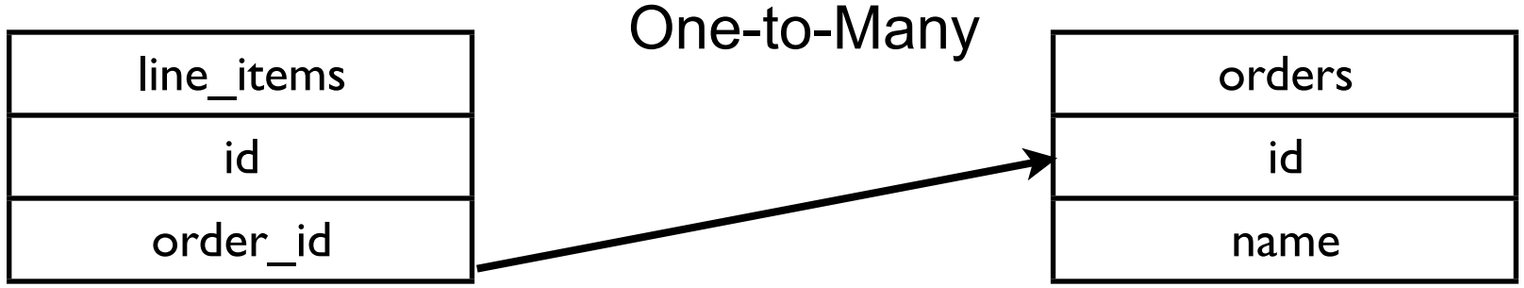
Table Relationships

One-to-One


```
class Invoice < ActiveRecord::Base
    belongs_to   :order
end
```

```
class Order < ActiveRecord::Base
    has_one      :invoice
end
```

One-to-Many




```
class LineItem < ActiveRecord::Base
    belongs_to   :order
end
```

```
class Order < ActiveRecord::Base
    has_many    :line_items
end
```

Table Relationships

Many-to-Many


```
class Category < ActiveRecord::Base
    has_and_belongs_to    :products
end
```

```
class Product < ActiveRecord::Base
    has_and_belongs_to    :categories
end
```

One-to-Many

books

id	title	author	date	publisher_id
1	Agile Web Development with Rails	Thomas, Dave	2005-11-17	1
2	Programming Ruby	Thomas, Dave	2005-01-02	1
3	Web Component Development with Zope 3	Weitershausen	2005-05-01	2

*

1

publishers

id	name
1	The Pragmatic Bookshelf
2	Springer

Publishers Table

```
class CreatePublishers < ActiveRecord::Migration
  def self.up
    create_table :publishers do |t|
      t.column :name, :string
    end
  end

  def self.down
    drop_table :publishers
  end
end
```

Books Table

```
class CreateBooks < ActiveRecord::Migration
  def self.up
    create_table :books do |t|
      t.column :title, :string
      t.column :author, :string
      t.column :date, :date
      t.column :publisher_id, :integer
    end
  end

  def self.down
    drop_table :books
  end
end
```

Models

```
class Book < ActiveRecord::Base
   belongs_to :publisher
end
```

Methods Added to Book

```
publisher(force_reload=false)
publisher=(object)
build_product(attributes={})
create_product(attribute={})
```

```
class Publisher < ActiveRecord::Base
   has_many :books
end
```

Methods Added to Publisher

```
books(force_reload=false)
books<<aBook
books.push(book1, ...)
books.delete(book1, book2,...)
books.clear
books.find(options...)
books.build(attributes={})
books.create(attributes={})
books.size
books.emply?
```

http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html

Sample Publisher Access

```
require File.dirname(__FILE__) + '/../test_helper'

class BookTest < Test::Unit::TestCase
  fixtures :books

  def test_Read
    ruby = Book.find_by_title('Programming Ruby')
    assert_equal(ruby.author, 'Thomas')
    assert_equal(ruby.id, 1)
    assert_equal(ruby.date, Date.new(2005,1,1))
    assert_equal(ruby.publisher.name, 'The Pragmatic Bookshelf')
  end
end
```

belongs_to

```
class Book < ActiveRecord::Base
    belongs_to :publisher
end
```

Defaults

Foreign Key	Target Class	Target Table
publisher_id	Publisher	publishers

Overriding Defaults

```
class Book < ActiveRecord::Base
    belongs_to :publisher
                                    :class_name => "BookPublisher"
                                    :foreign_key => "pub_id"
end
```

http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html#M000532

Automatic Saving

```
springer = Publisher.find_by_name('Springer')

networking = Book.new
networking.title = 'Fundametantal Networking in Java'
networking.date = Time.now
springer.books<<networking
```

Book networking is now saved in the database

Automatic Saving - One Way only

```
addison = Publisher.new
addison.name = 'Addison-Wesley'

ruby = Book.find_by_title('Programming Ruby')
ruby.publisher = addison
```

Publisher addison is not saved
ruby not updated

```
addison = Publisher.new
addison.name = 'Addison-Wesley'

ruby = Book.find_by_title('Programming Ruby')
ruby.publisher = addison
ruby.save
```

Now addison is saved in the database
ruby is updated