

# Introduction au langage Ruby et au framework Rails

Laurent Julliard <laurent chez nuxos point fr >  
[www.nuxos.fr](http://www.nuxos.fr)

Présentation Association GUILDE  
19 Décembre 2007



Version 1.5 16/12/2007





# Petit sondage...

Qui programme en...

- C/C++
  - Java/JSP...
  - C#/.Net
  - PHP
  - Perl
  - Python
- et...

## Ruby



# Ruby: origine et principes

- Langage né au Japon en 1993. Le papa: Yukihiro Matsumoto dit “Matz”.
  - Première release en 1995.
- Un langage de scripting de haut niveau orienté objet
  - Tout n'est qu'objet...
- Mariage réussi de plusieurs courants
  - Smalltalk, Lisp, Perl,...
- Suit le principe “POLS” ou principe de moindre surprise
  - Une syntaxe simple et lisible (littéralement)
- Fonctionne sur toutes les plateformes du marché



# Un langage ça se lit...

- Les langages humains se lisent, Ruby aussi...

```
5.times { print "Quick !" }
```

```
return unless books.include? les_miserables
```

```
['toast', 'cheese', 'tea'].each { |food| print  
  food.capitalize }
```



# Installer Ruby (et Rails)

- Sur Windows

- Ruby One-click Installer (Curt Hibbs) sur <http://rubyforge.org>
- Instant Rails = Ruby, Rails, Apache, MySQL (même site, même auteur)

- Sur Linux

- Ruby livré sous forme de packages avec la plupart des distributions (Fedora, Mandriva, Suse, Ubuntu, Debian...)
- Installation aussi très simple à partir des sources:

```
Pour Ruby: ./configure; make; make install
```

```
Puis Rails: gem install rails
```

- Sur Mac OS X

- Ruby: maintenant livré en standard par Apple sur 10.3 (version 1.8.2), dans Leopard tout sera inclus (Ruby 1.8.4, Rails, MySQL, Mongrel...)
- Ruby/Rails tout-en-un avec Locomotive (<http://locomotive.raaum.org/>)



# Ruby: tour d'horizon



# Partout des objets et des messages

- En Ruby tout est objet
- Un appel de méthode = envoi d'un message à un objet

Notation:

objet.message(...)

- Tout objet: oui mais...
  - des conventions de nommage et des notations implicites pour pouvoir faire court si on le souhaite !

```
"Ruby".size  
[1, 3, 5, 7].reverse  
5.succ  
-10.abs
```

```
# L'incontournable "Hello World"  
puts "Salut !"
```

```
#... qui signifie en fait  
STDOUT.puts("Salut !")
```

```
# ou encore sous forme de message  
STDOUT.send(:puts, "Salut !")
```

```
#... et des objets partout  
STDOUT.class  
STDOUT.methods  
STDOUT.methods.grep(/print/)  
STDOUT.methods.grep(Regexp.new('print'))
```



# Quelques classes de base

- **String**

- Chaîne de caractères

```
"Ruby", 'Ruby', String.new('Ruby')
```

- **Array**

- Tableau d'objets

```
[1, "deux", 5**20]
```

- **Hash**

- Tableau associatif d'objets

```
{ 'I' => 1, 'II' => 2 }  
Hash.new(...)
```

- **Integer (Fixnum, Bignum)**

```
5, 5**20
```

- **Float**

```
3.5, 23.2E-6
```

- **Ranges**

```
10..100, -3...5
```

- **FalseClass, TrueClass, NilClass**

```
true, false, nil
```

```
nil, false, 0, ""
```

ne signifient pas la même chose !!





# Conventions de nommage

- **NomDeClasse** (ex: LivrePoche)
- `nom_methode` et `nom_variable` (ex: `un_livre`)
- `methode_qui_questionne?` (ex: `existe?`)
- `methode_dangereuse!` (ex: `destruire!`)
- `@variable_d_instance` (ex: `@titre`)
- `@@variable_de_classe` (ex: `@@nb_de_livres`)
- `$variable_globale` ou `$VARIABLE_GLOBALE` (ex: `$LOAD_PATH`)
- **CONSTANTE** ou `AutreConstante` (ex: `PLATFORM`)
- `:symbole` (ex: `:option`)



# Exemple: classe, méthodes...

Une déclaration de classe comprend:

- un nom de classe ■
- des déclarations de:
  - méthodes d'instance ou de classe ■
  - des variables d'instances (attributs) ou de classe ■
  - des accesseurs ■
  - constantes ■
  - classes imbriquées (rares)
- Autre: variable globale ■

```
$SAFE = 1
```

```
class Avion
```

```
  @@avion_fabriques = 0
  attr_reader :moteurs
  attr_accessor :altitude, :speed
```

```
  def initialize(moteurs)
    @moteurs = moteurs
    @@avion_fabriques += 1
  end
```

```
  def self.quantite
    @@avion_fabriques
  end
```

```
end
```

```
zingue = Avion.new(2)
puts zingue.moteurs
zingue.altitude = 33000
puts Avion.quantite
```



# Les classes sont des objets

Les classes sont des objets  
comme les autres

- Elles sont toutes du type Class
- Ruby utilise une constante pour les nommer

Constante: Avion

Object pointer

Objet Class

```
puts Avion.class # Class

def factory(a_class, *args)
  a_class.new(*arg)
end

factory(Array, [1,2,3])
factory(Avion, 4)

[String, Array, Hash][rand(3)].new

# Une classe peut-être créée
var = Class.new
puts var.name #... chaine vide

# et nommer plus tard
AvionChasse = var
puts var.name # "AvionChasse"
```



# Methodes et paramètres

- Tous les paramètres sont passés par référence sur les objets.
- Une valeur par défaut peut être spécifiée pour les paramètres
- Parenthèses facultatives lors de l'appel d'une méthode
- Paramètres nommés (utilisation d'un Hash)
- La valeur retournée est la dernière valeur évaluée (souvent absence de return)
- Les méthodes sont public, protected ou private

```
def edit(chaine)
  chaine[3..6] = "soir"
end

ma_chaine = "Bonjour"
edit(ma_chaine)
puts ma_chaine # "Bonsoir"

# valeur par défaut
def save(flush=true)
  ...
end
save          # comme save(true)
save false   # parenthèses facultatives

# paramètres nommés
def modifier(args)
  puts args[:nb_pages]
  puts args[:auteur]
end
@livre.modifier(:nb_pages => 300,
                :auteur => 'DHH')
```



# Héritage

- Modèle d'héritage simple
- La Classe 'Object' est l'ancêtre de toutes les classes

```
class AvionCargo < Avion
```

```
  attr_accessor :load
```

```
  def initialize(moteurs, load)
```

```
    @load = load
```

```
    super(moteurs)
```

```
  end
```

```
end
```

```
ac = AvionCargo.new(2, 16000)
```

```
# Inspecter la chaine d'héritage...
```

```
puts ac.class.superclass
```

```
puts ac.class.ancestors
```

```
# ...[AvionCargo, Avion, Object, Kernel]
```

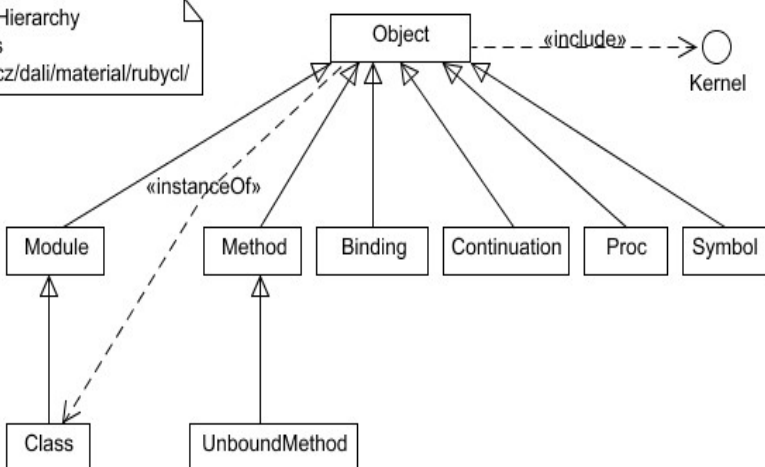
```
# Un peu d'introspection...
```

```
ac.respond_to? :altitude
```

```
ac.kind_of? Avion # true
```

```
ac.instance_of? Avion # false
```

Ruby Class Hierarchy  
Core classes  
[www.insula.cz/dali/material/rubycl/](http://www.insula.cz/dali/material/rubycl/)





# Les modules

- Les modules sont des classes non instantiables
- Elles ont deux usages
  - Structurer l'espace de nommage sur un modèle hiérarchique
  - Permettre le 'mixin' c'est à dire étendre les compétences d'une classe



# Modules: Espace de nommage

- Ruby permet d'imbriquer les déclarations de classe ou de modules
- Permet d'organiser l'espace de noms en structure arborescente
- Le nommage se fait avec la notation `::` comme dans `A::B::C`

```
# une classe déclarée dans un module  
Module ActiveRecord  
  class Base  
    ...  
  end  
end  
  
# et comment la nommer  
ar = ActiveRecord::Base.new
```



# Modules: Mixin

- Pas d'héritage multiple en Ruby mais utilisation des 'Mixin'

- utilisation du mot clé `include`
- un Mixin étend les compétences d'une classe en lui ajoutant une série de méthodes
- Exemple: `Enumerable`, `Comparable`, ...

```
class CollectionBouquin
  include Enumerable
  # comment itérer sur la collection
  def each
    ...
  end
  # relation d'ordre entre 2 éléments
  def <=> (autre_elt)
    ...
  end
end

c = CollectionBouquin.new(...)

# En incluant Enumerable on a gratuitement:
# min, max, find, sort, between?, etc...
c.find { |bouquin| bouquin.title =~ /Ruby/ }
c.min
c.sort
```





# Le 'typage à la canard' (*duck typing*)

- Paramètres et variables ne sont pas typés statiquement MAIS à l'exécution tous les objets sont l'instance d'une classe.
  - En pratique les erreurs de typage n'arrivent jamais
  - Sauf sur 'nil' et c'est une bonne chose (= valeur indéfinie)
- Ruby juge un objet sur ce qu'il est capable de faire pas sur un type défini statiquement.
  - Si un objet marche comme un canard et fait coin-coin alors Ruby le voit comme un canard
- **Conséquence: le code Ruby peut et doit être très générique**
  - On repousse le plus loin possible le moment où le code est spécifique d'un "type"



# Blocs et itérateurs



# Blocs

- Les blocs de code sont **incontournables** en Ruby
  - { ..... }
  - do  
.....  
end
- Bloc = fragment de code avec un contexte d'exécution
- C'est un objet comme les autres (classe Proc)
- Peut-être passé en paramètre et accepte lui aussi des paramètres

*# Anatomie générale d'un bloc*

```
{ |p1, p2, ...| code... }
```

Paramètres

Code exécutable

```
b2 = Proc.new { |x,y| x+y }  
b2.call(55,45)
```



# Itérateurs, transaction, closures...

- Ruby offre les habituelles boucles `for`, `while`, ... mais on leur préfère les itérateurs
- Le bloc peut être utilisé en mode “transaction”
- On peut aussi l'utiliser comme une “closure”
  - i.e. un bloc de code avec tout son contexte préservé

```
5.times { print "Coco!!"}  
  
arr = [23, 34, "33", 4.5]  
  
arr.sort_by { |elt| elt.to_i }  
# [4.5, 23, "33", 34 ]  
  
arr.collect { |elt| puts elt.class }  
# [Fixnum, Fixnum, String, Fixnum]  
  
# Transaction avec close implicite  
File.open("/tmp/bigfile.txt") do |fh|  
  fh.each_line { |l| puts l if l.size > 40 }  
end  
  
# Closure  
transfer = FileTransfer.new  
b1 = Button.new("Start") { transfer.start }
```



# Itérateurs (définition)

- Un bloc est passé comme un paramètre lors d'un appel de méthode
- Le bloc est sollicité par le mot réservé `yield` suivi de paramètres

```
def bis_repetita(&action)
  yield
  yield
end
```

```
bis_repetita { puts "Coco!" }
```

```
1.upto(5) { |step| puts "Step #{step}" }
```

*# une implémentation possible*

```
class Fixnum
  def upto(max, &bloc)
    for i in self..max do
      yield i
      i = i.succ
    end
  end
end
```



# Les symboles

- Une autre “étrange bestiole” ...  
    :nom\_du\_symbole
- Un symbole est une étiquette
- Ce n'est ni une variable, ni une chaîne de caractères...

```
Product.find( :all, :limit => 10, :order => 'prix', :readonly => true)
```



# Bibliothèques et Packages



# Les classes intégrées

- Types de base

- Object
- String, Array, Hash, Range, Regexp
- Integer, Fixnum, Bignum, Float, Numeric

- Fichiers

- IO, File, FileTest, Dir, Errno

- Math

- trigo, log, exp....

- Autres

- Kernel
- Exception
- Process, Signal
- Thread, ThreadGroup
- Time
- GC
- Marshal
- TrueClass, FalseClass, NilClass





# Les classes standards

- Database
  - DBM, GDBM, SDBM
- Protocoles réseau
  - Net (http, ftp, imap, pop, smtp, telnet), openssl
- Web/XML
  - CGI, URI, REXML, RSS, SOAP, XMLRPC, Yaml, Webrick
- Concurrency/Distribution
  - dRuby, Rinda
  - Thread, Mutex, Monitor,
- Math
  - Complex, Rational, mathn, Matrix,
- Lib. externes
  - DL
- Et beaucoup d'autres...



# L'entourage de Ruby



# Les Gems

- Gems: Outil de gestion de packages logiciels de Ruby
- Gère les dépendances, les mises à jour, installe les librairies, exécutables et documentation

class ActionMailer::Part  
In: lib/action\_mailer/part.rb  
Parent: Object

Represents a subpart of an email message. It shares many similar attributes of ActionMailer::Base. Although you can create parts manually and add them to the parts list of the mailer, it is easier to use the helper methods in ActionMailer::PartContainer.

Methods  
new to\_mail

Included Modules  
ActionMailer::AdvAttrAccessor ActionMailer::PartContainer

Public Class methods  
new(params)  
Create a new part from the given params hash. The valid params keys correspond to the accessors.

Public Instance methods  
to\_mail(defaults)  
Convert the part to a mail object which can be included in the parts list of another mail object.

[Validate]

## \$ gem search active

\*\*\* LOCAL GEMS \*\*\*

activerecord (1.14.4)

*Implements the ActiveRecord pattern for ORM.*

activesupport (1.3.1)

*Support and utility classes used by the Rails framework.*

## \$ gem install pdf-writer

Bulk updating Gem source index for: <http://gems.rubyforge.org>

Successfully installed pdf-writer-1.1.3

Installing ri documentation for pdf-writer-1.1.3...

Installing RDoc documentation for pdf-writer-1.1.3...

## \$ gem\_server # Gem documentation server





# Les outils de développement

- Shell interactif

- Irb: Ruby en direct

- Debugger

- ruby -r debug...
- pas à pas, point d'arret, pile d'appel, etc...

- Profiler

- ruby -r profile

%self	cumulative	total	self	children	calls	self/call	total/call	name
23.64	0.13	0.13	0.13	0.00	10002	0.00	0.00	String#split
21.82	0.30	0.39	0.12	0.27	10002	0.00	0.00	CallRot#read_line
18.18	0.55	0.54	0.10	0.44	1	0.10	0.54	#foreach
12.73	0.42	0.07	0.07	0.00	10002	0.00	0.00	#local
5.45	0.33	0.05	0.03	0.02	10002	0.00	0.00	CallRot#line_is_future?
3.64	0.44	0.02	0.02	0.00	10003	0.00	0.00	#allocate
3.64	0.35	0.02	0.02	0.00	20004	0.00	0.00	Array#pop
3.64	0.17	0.02	0.02	0.00	10002	0.00	0.00	Comparable#>
3.64	0.15	0.02	0.02	0.00	10002	0.00	0.00	OnCall#initialize
1.82	0.45	0.01	0.01	0.00	1	0.01	0.01	#open

- Couverture de code

- rcov  
(<http://eigenclass.org/hiki.rb?rcov>)

C0 code coverage information  
Generated on Mon May 22 12:09:23 CEST 2006 with rcov 0.4.0  
Threshold: 80%

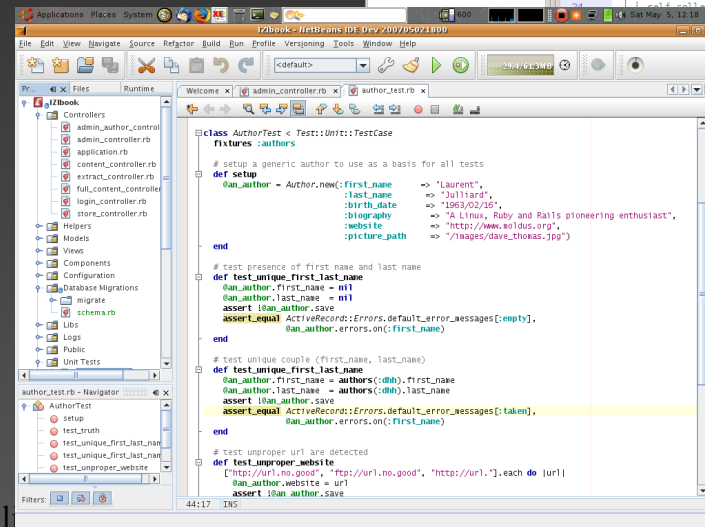
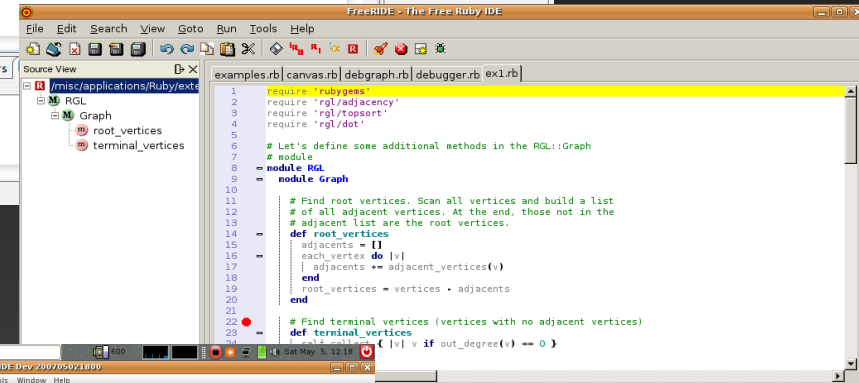
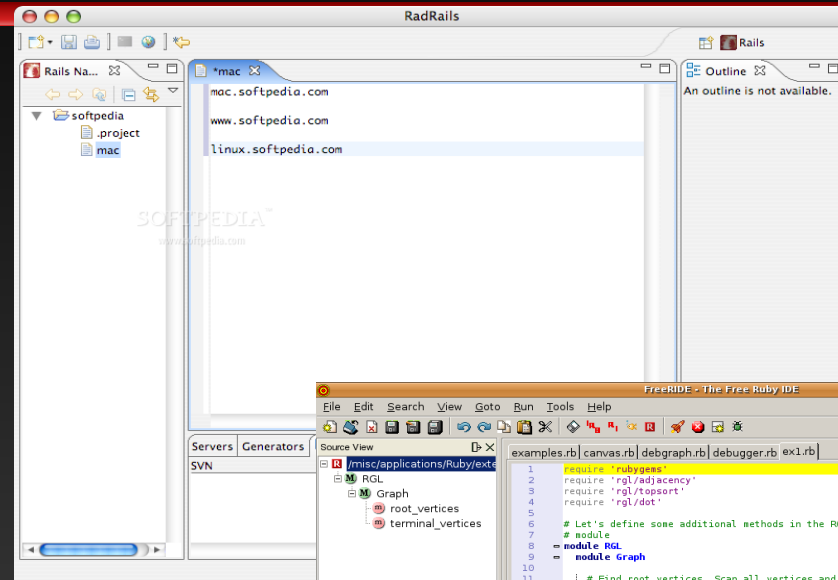
Name	Total lines	Lines of code	Total coverage	Code coverage
TOTAL	1754	1754	69.2%	60.1%
app/controllers/application.rb	39	39	46.2%	31.0%
app/helpers/application_helper.rb	147	147	30.1%	23.5%
app/models/aggregations/tada.rb	75	75	45.3%	31.1%
app/models/aggregations/upcoming.rb	78	78	48.7%	29.2%
app/models/article.rb	109	109	74.3%	67.9%
app/models/category.rb	30	30	66.7%	60.9%
app/models/sidebar.rb	36	36	55.6%	48.7%
components/plugins/sidebars/archives_controller.rb	35	35	34.3%	29.6%
components/plugins/sidebars/category_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/delicious_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/flickr_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/fortythree_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/fortythreeplaces_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/static_controller.rb	27	27	37.0%	29.2%
components/plugins/sidebars/tada_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/technorati_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/upcoming_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/xml_controller.rb	16	16	62.5%	53.8%
components/sidebars/sidebar_controller.rb	110	110	48.2%	32.5%
lib/html_engine.rb	29	29	82.8%	78.3%
lib/login_system.rb	85	85	60.0%	23.5%
lib/migrator.rb	28	28	53.6%	40.9%



# Les outils de développement

## Environnements de développement

- Editeurs: X/Emacs, vi, TextMate (Ruby mode)
- NetBeans 6.0 (Ruby/Rails mode)
- Komodo
- Aptana: basé sur Eclipse
- FreeRIDE (écrit en Ruby)
- Ruby In Steel (add-on Visual Studio 2005/2008)
- ArachnoRuby





# Le futur de Ruby

- Actuellement en version 1.8.6
  - Très stable, une release mineure tous les 6 à 8 mois.
- Ruby 1.9.x, version expérimentale avant la 2.0.
- 4 projets de machines virtuelles en cours !!!
  - Machine Virtuelle YARV (l'héritière)
  - JVM de Sun (JRuby)
  - CLR de Microsoft (IronRuby et Ruby.NET)
  - Machine Virtuelle Smalltalk (Rubinius)



# Le framework Ruby on Rails

Laurent Julliard <laurent chez nuxos point fr >

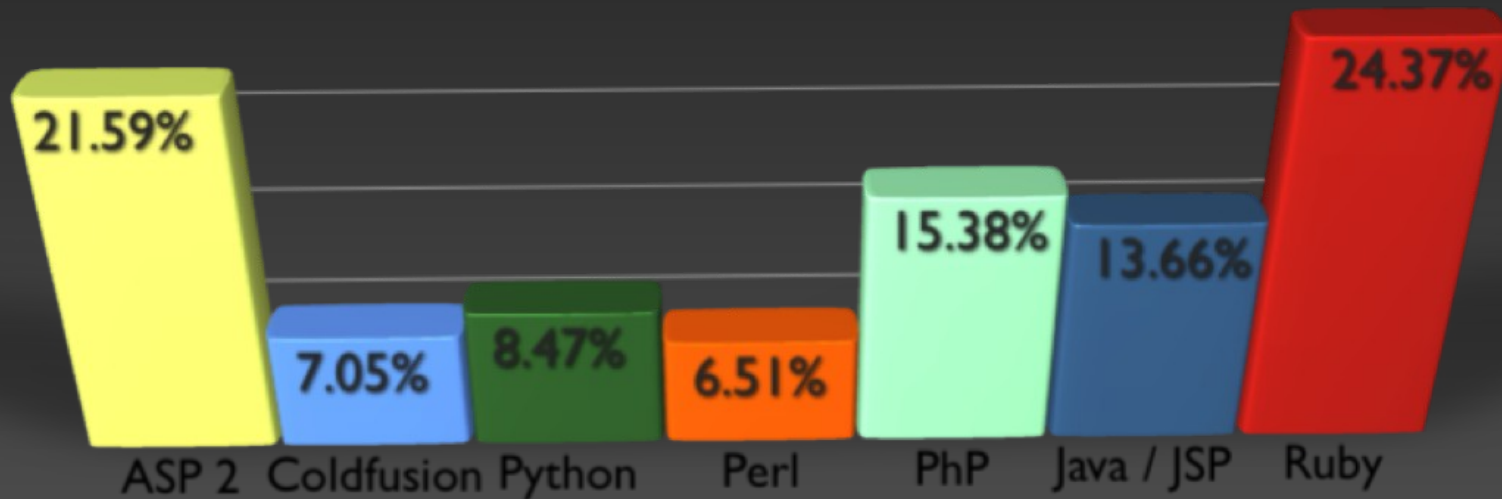
Paris On Rails 2007



# Adoption ?

Près de 6% des développeurs Web ont déjà **adopté** Ruby & Rails et presque **25%** s'y intéressent [1]

*What server platform(s) are you not currently developing for, but expect to in the next 12 months?[1]*



[1] <http://sitepoint.com/reports/reportwebsurvey2006/fr>





# Hot Stuff !

## The Top Five Technologies You Need to Know About in '07

From next-gen CPU architectures to high-powered personal-area networks, we name the five hottest trends in developing technology.

George Jones Today's Top Stories or Other Hardware Stories

Comments (7) Recommendations: 170

March 01, 2007 (Computerworld) – It seems like every month a change everything. Technology writers and analysts get hypere; back and hugging. And two years later, we're still talking about it the here and now.

That's why as we began to look at core technologies that may be over the next 12 months, we paid special attention to how soon users, either at the enterprise or the personal level. The result is with groundbreaking potential – this year as well as in the future.

### Five Hot Technologies for 2007

- 1. Ruby on Rails**  
Faster, easier Web development
- 2. NAND drives**  
Bye-bye, HDD?
- 3. Ultra-Wideband**  
200x personal-area networking
- 4. Hosted hardware**  
Supercomputing for the masses
- 5. Advanced CPU architectures**  
Penryn, Fusion and more

### 1. Ruby on Rails: Faster, easier Web development

Chances are you've heard the term *Ruby on Rails* – probably from someone on your Web development team lobbying heavily to use it for some or all of your company's Web development.

Ruby on Rails (also known as RoR and Rails) is a Web application framework written in Ruby, an object-oriented programming language known for its clean syntax. Released in 2004, RoR is an open-source project that originally served as the foundation of a project management tool designed by Web development company 37signals LLC. It is easily ported among Linux, Windows and Macintosh environments, and it can have a dramatic impact on the speed at which a Web development team is able to build and maintain enterprise Web sites and applications.

## Five Hot Technologies for 2007

### 1. Ruby on Rails

Faster, easier Web development

### 2. NAND drives

Bye-bye, HDD?

### 3. Ultra-Wideband

200x personal-area networking

### 4. Hosted hardware

Supercomputing for the masses

### 5. Advanced CPU architectures

Penryn, Fusion and more

#### WEBCAST

- Get the Requir
- Intern leverages leading-edge technology to keep up with rapid data-growth
- Optimize Your Business Applications with a Server-to-Storage Virtualized Infrastructure

#### WHITEPAPERS

- BlueArc Provides Powerful Platform for Storage Infrastructure Consolidation

#### WEBCAST

Optimize Your Business Applications with a Server-to-Storage Virtualized Infrastructure



# IBM

## De nombreux articles sur le site Developerworks [1]

IBM Country/region [select] All of dW

Home | Products | Services & industry solutions | Support & downloads

developerWorks > Open source | Java technology >

**developerWorks**  
In this article:  
Why Ruby?

### Using the Ruby Development Tools plug-in for Eclipse

IBM Country/region [select] All of dW

Home | Products | Services & industry solutions | Support & downloads

developerWorks > Open source >

**developerWorks**  
In this article:  
Downloading and installing

### Make Ruby on Rails easy with RadRails and Eclipse

IBM Country/region [select] All of dW

Home | Products | Services & industry solutions | Support & download

developerWorks > Open source | Linux >

**developerWorks**  
AIX and UNIX  
IBM Systems

### Test-first Ruby programming

IBM Country/region [select] Terms of use

Home | Products | Services & industry solutions | Support & downloads | My IBM

developerWorks > Information Management | Open source >

**developerWorks**  
More in this series:  
DB2 and Ruby on Rails

### DB2 and Ruby on Rails, Part 1: Getting started with DB2 and Ruby on Rails

Set up IBM\_DB adapter and driver for Rails application development

In this

IBM Country/region [select] All of dW

Home | Products | Services & industry solutions | Support & downloads

developerWorks > Java technology | Web development >

**developerWorks**  
More in this series:  
Crossing borders

### Crossing borders: What's the secret sauce in Ruby on Rails?

Lessons we can learn from the Rails framework

IBM Country/region [select] Terms

Home | Products | Services & industry solutions | Support & downloads | My IBM

developerWorks > Linux | Open source | Web development >

**developerWorks**  
In this article:  
Introducing Rails  
Generating code

### Fast-track your Web apps with Ruby on Rails

This Ruby-based framework for rapid development uses a model-view-controller pattern



# IBM



RAILSFRANCE.ORG ET LES ÉDITIONS EYROLLES PRÉSENTENT

## Paris on Rails,

la première conférence officielle Rails en France

17 novembre 2006  
Tour Descartes  
La Défense 5, Paris

[Accueil](#) [Conférenciers](#) [Planning](#) [Plan d'accès](#) [Inscription](#) [Partenaires](#)

### Développeurs, chefs de projets, décideurs

Venez découvrir Ruby on Rails, un framework open-source qui révolutionne le développement d'applications web. Rails maximise la productivité du développeur et le retour sur investissement de l'entreprise en permettant la création d'applications robustes et évolutives. Déjà adopté par de nombreuses entreprises outre-Atlantique le phénomène Ruby on Rails déferle désormais sur l'Europe.

Les meilleurs experts francophones seront présents à Paris pour vous faire découvrir tous les avantages de ce framework révolutionnaire et partager avec vous leurs retours d'expérience.

Réservation en ligne obligatoire - Tarif unique de 60 €

[S'inscrire](#)

### Pourquoi Rails révolutionne le développement Web ?

Basé sur le modèle MVC et sur l'utilisation de conventions, Ruby on Rails apporte au développeur d'applications Web pour l'entreprise un confort et une rapidité de développement jusque-là inégalés. De plus l'intégration des librairies AJAX Prototype et script.aculo.us fait de ce framework un des environnements les plus efficaces pour le développement d'applications "Web 2.0".

Merci à nos partenaires qui nous soutiennent dans l'organisation de cet événement :

EYROLLES



SALLE DE CONFÉRENCE IBM  
LA DÉFENSE 5 - PARIS

[Comment venir ?](#)

VENEZ RENCONTRER LES EXPERTS  
RAILS FRANCOPHONES



[Infos sur les conférenciers](#)

RESTEZ INFORMÉS : ABONNEZ VOUS À  
NOTRE LETTRE D'INFORMATION.

Votre e-mail

[S'abonner](#)



Principal sponsor de  
"Paris on Rails  
2006", la première  
conférence officielle  
française





# Sun Microsystems

- **Financement** du développement de JRuby
- **Charles Nutter** et **Thomas Enebo** sont désormais employés à plein temps par Sun..
- ... et travaillent sur le **portage** de l'interpréteur **Ruby** sur la **JVM**
- **Ruby** est le **premier langage dynamique** que **Sun** va porter sur la JVM

**InfoQ**  
Version 0.9

Tracking change and innovation in the enterprise software development community

News

Contribute News print

Welcome,  
Richard!  
[Sign out](#)  
[Preferences](#)  
[About us](#)  
[Personal feed](#)  
[Home](#)

## Sun Officially Backs Ruby, Brings JRuby In-House

Posted by [Peter Cooper](#) on Sep 07, 2006 12:16 PM

Community [Ruby](#), [Java](#) Topics [Change](#), [Open Source](#), [JCP Standards](#)

Charles Nutter [reports that he and JRuby co-developer Thomas Enebo are becoming employees](#) of [Sun Microsystems](#) later this month. JRuby is a pure Java implementation of the Ruby interpreter and Nutter reports that Java are backing a "Ruby-on-JVM" strategy:

Your





# Microsoft

- Intégration de Ruby dans Silverlight
- Financement du développement de IronRuby (Ruby CLR)
- John Lam est désormais employé à plein temps par Microsoft...
- ... et travaille sur le portage de l'interpréteur Ruby CLR

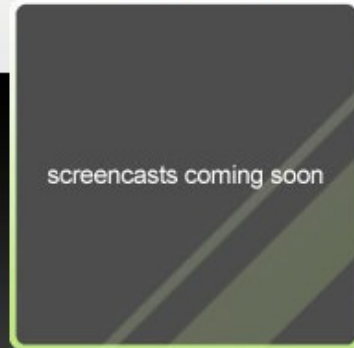


# Microsoft

rubyclr. **ruby** for the **.net framework**



welcome



screencasts | blog

start now!



download

community



wiki | mailing list

## People Talking RubyCLR

**eWeek** - [Microsoft Phasing in Support for Dynamic Languages on .Net](#)

**Port 25** - [John Lam and Sam Ramji discuss RubyCLR, Avalon Ruby Editor and Open Source Funding](#)



# Rails sur Ruby, pourquoi?

- Multi-plateforme
- Compact
  - 2 à 4 fois moins de code qu'en Java/C#/C/C++
- Expressif
  - Forte capacité d'auto-analyse (introspection et réflexion)

```
User.find_by_name_and_country("Hansson", "Danemark")
```

```
has_many :adresses
```



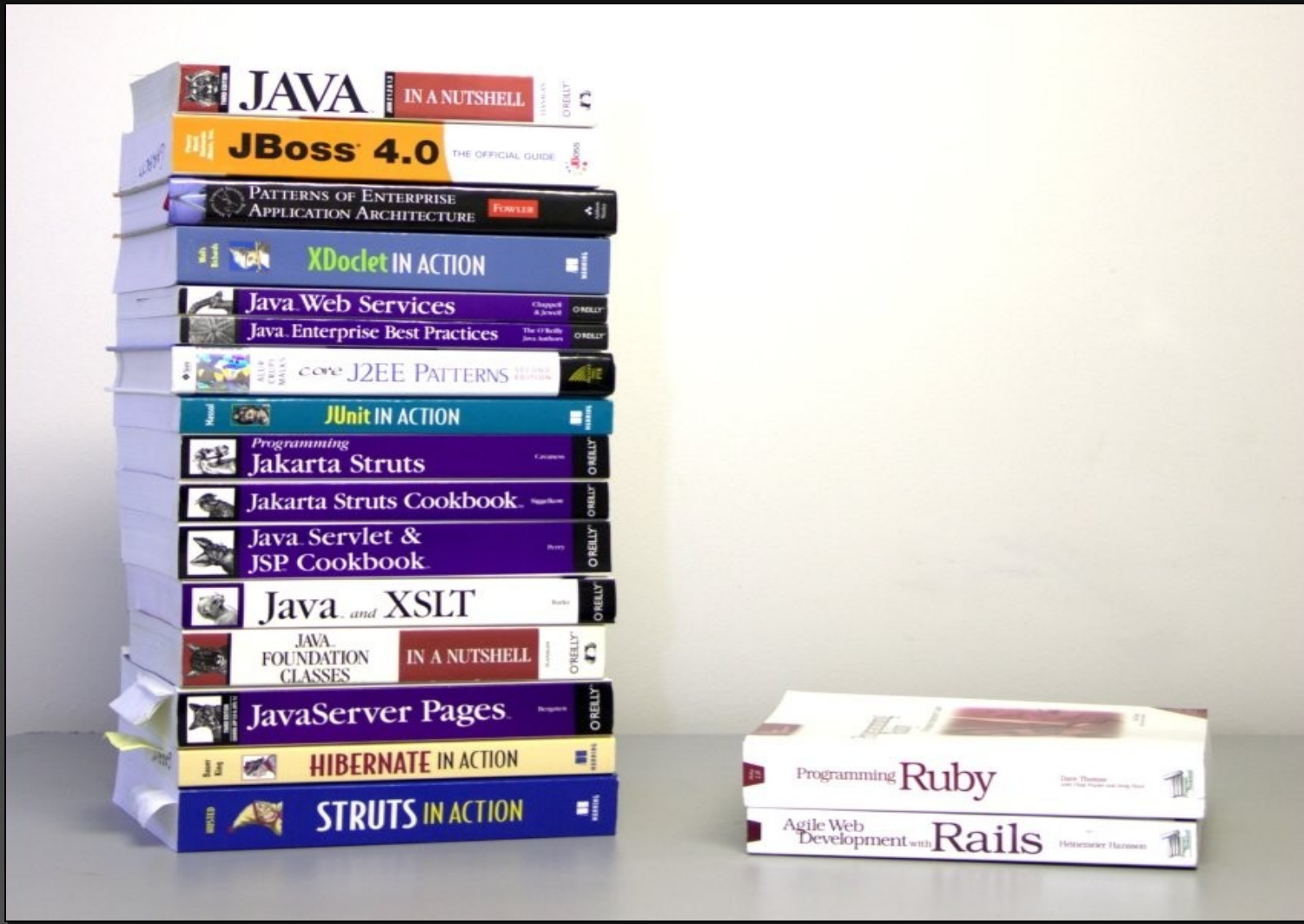
# La force de Rails: ses principes!

- Principe #1 : “*Don't Repeat Yourself*” (DRY)
  - Eviter les répétitions
- Principe #2 : “*Convention over configuration*”
  - Compacité et organisation du code
- Principe #3 : “*Say What You Do, Do What You Say*”
  - Expressivité et extensibilité du langage (DSL)
- Principe #4 : “*One language to rule them all...*”
  - Un seul langage pour tout faire + forte intégration





# Résultat...





# MVC... le retour

## ActionController



1

Contrôleur

1. Requête du navigateur
2. Le contrôleur interagit avec le modèle
3. Le contrôleur invoque la vue
4. La vue restitue le prochain écran du navigateur

4

3

2

Vue

Modèle

Base de données

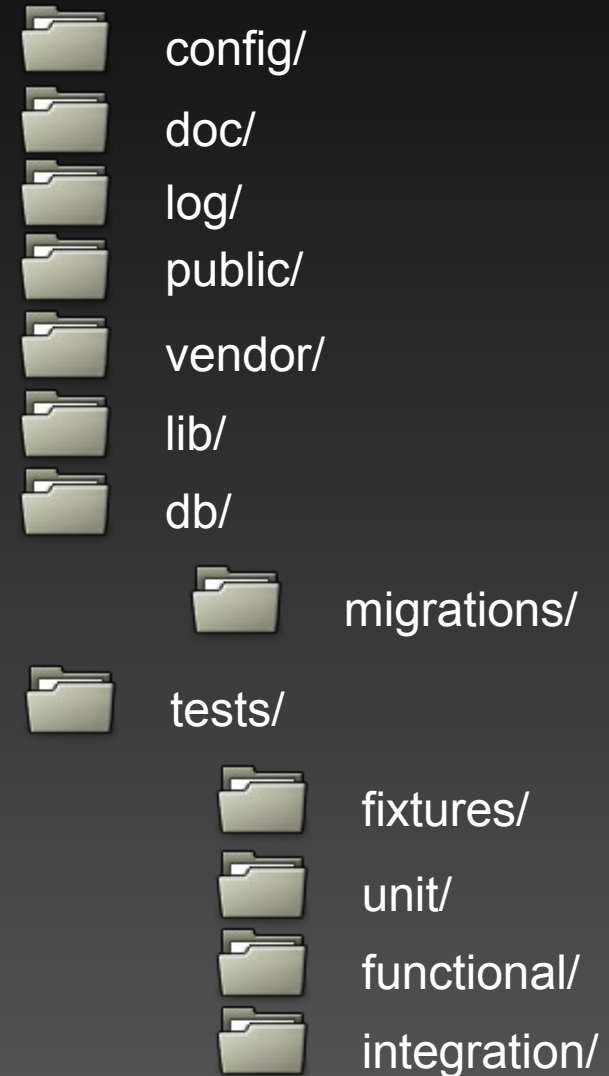
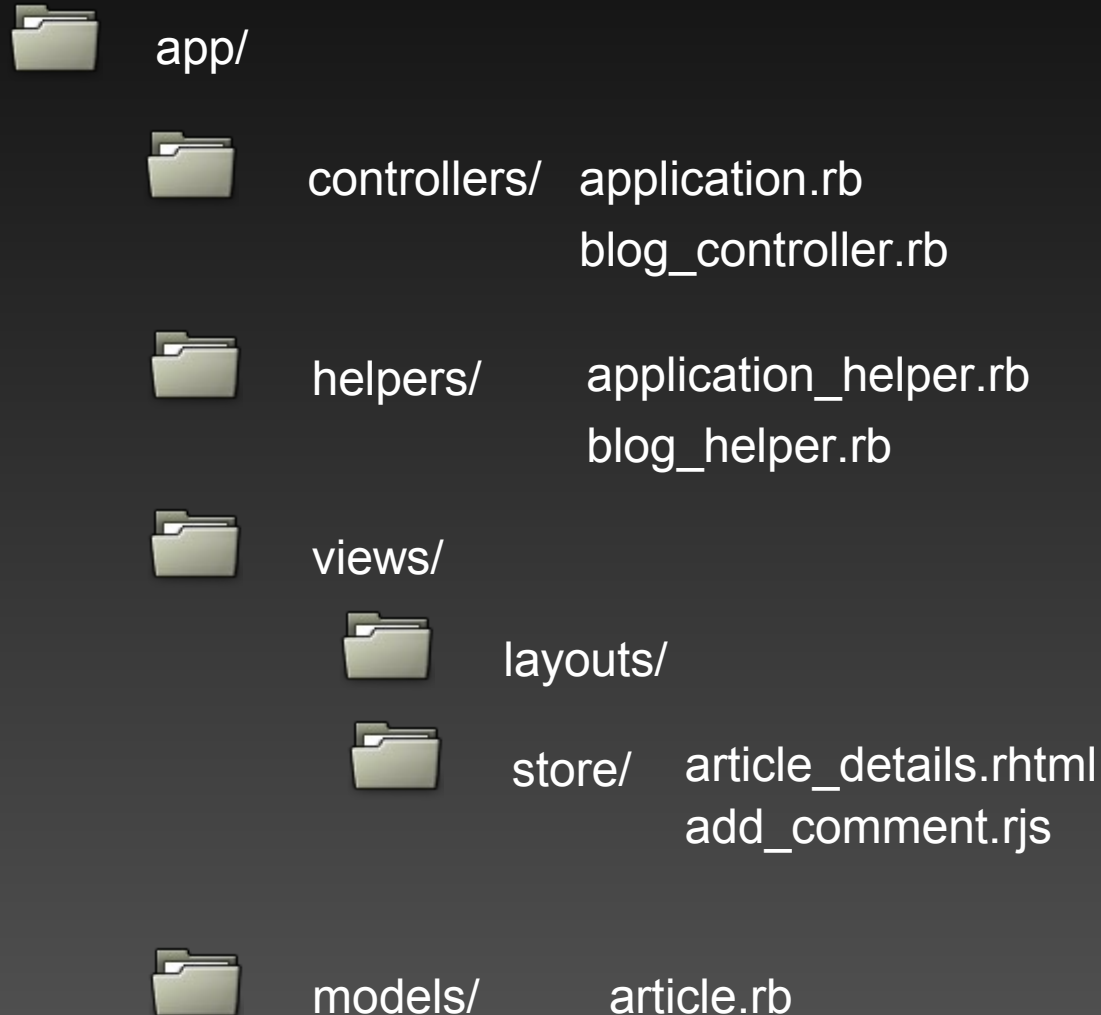
## ActionView

## ActiveRecord

SQLite, MySQL, DB2, Oracle, SQLServer...



# Chaque chose à sa place...





# Modèle



# ActiveRecord: d'abord un ORM

```
# app/models/produit.rb
```

```
class Produit < ActiveRecord::Base  
end
```

produits
id
titre
description
prix
created_at
updated_at

```
produit1 = Produit.new  
produit1.titre = "La métaphysique des tuyaux"  
produit1.save
```

```
produit2 = Produit.create( :titre => "...")  
produit2.destroy
```

```
tous_les_produits = Produit.find :all
```



# Les migrations

```
# Exemple pour le modèle produit
# db/migrate/001_create_products.rb

class CreateProduits < ActiveRecord::Migration

  def self.up
    create_table :produits do |t|
      t.string,      :titre, :limit => 100
      t.text,        :description
      t.decimal,     :prix,  :scale => 2
      t.timestamps  # crée created_at et updated_at (datetime)
    end
  end

  def self.down
    drop_table :produits
  end

end
```



# Les validations

## Les cerbères avant l'entrée des objets dans la base

```
class Produit < ActiveRecord::Base

  validates_presence_of :titre, :description
  validates_numericality_of :prix, :greater_than => 0.01
  validates_uniqueness_of :titre

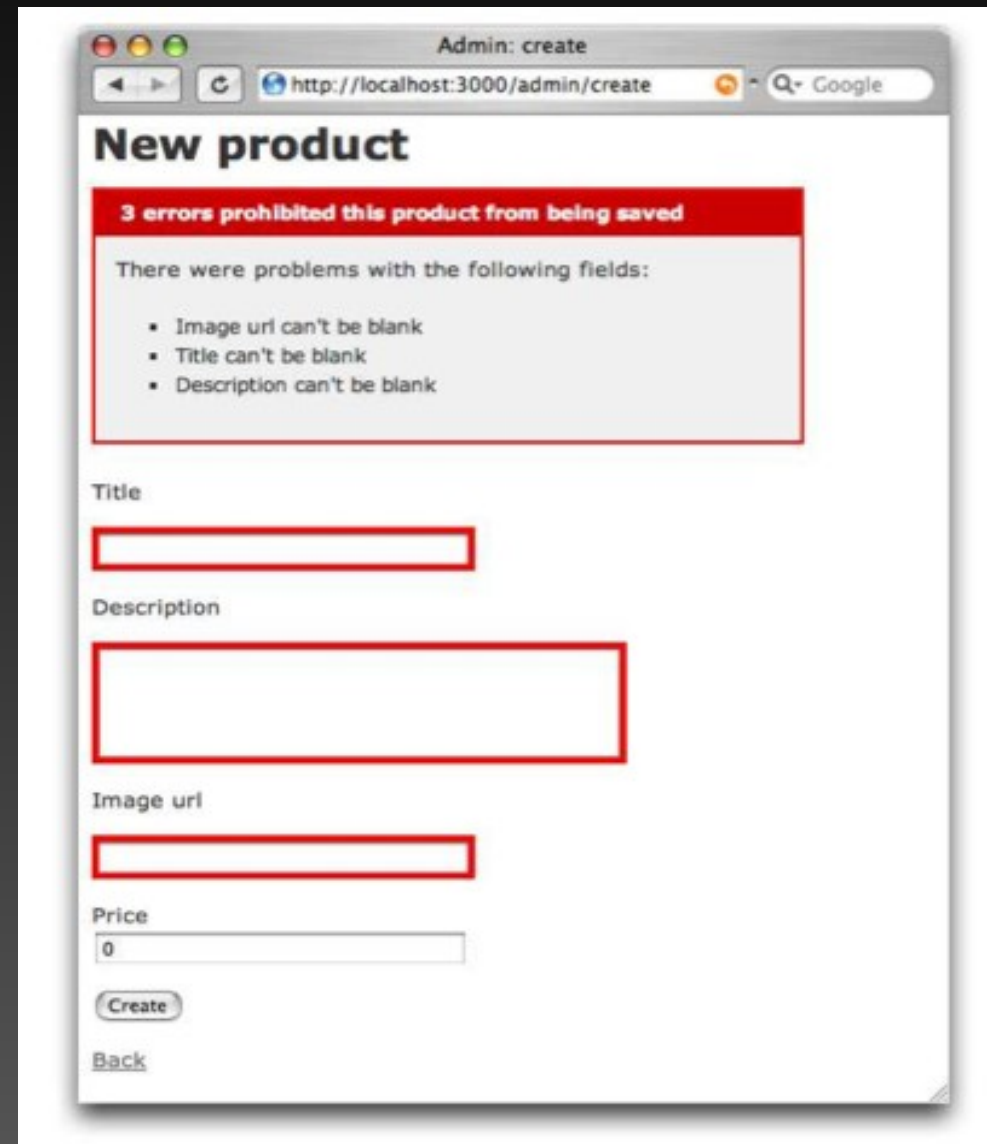
end
```



# ... et le retour utilisateur

## Affichage des erreurs en une ligne dans la vue

```
<%= error_messages_for :product %>
```







# Les associations: relation 1:N

**has\_many** et **belongs\_to**

articles
id
titre
texte



commentaires
id
article_id
note
created_at
updated_at

```
class Article < ActiveRecord::Base
  has_many :commentaires
end
```

```
class Commentaire < ActiveRecord::Base
  belongs_to :article
end
```



# Les associations: relation 1:N

## Les méthodes qui en découlent...

```
mon_article.commentaires
mon_article.commentaires.count

mon_article.commentaires.find( :all, :order => 'created_at' )

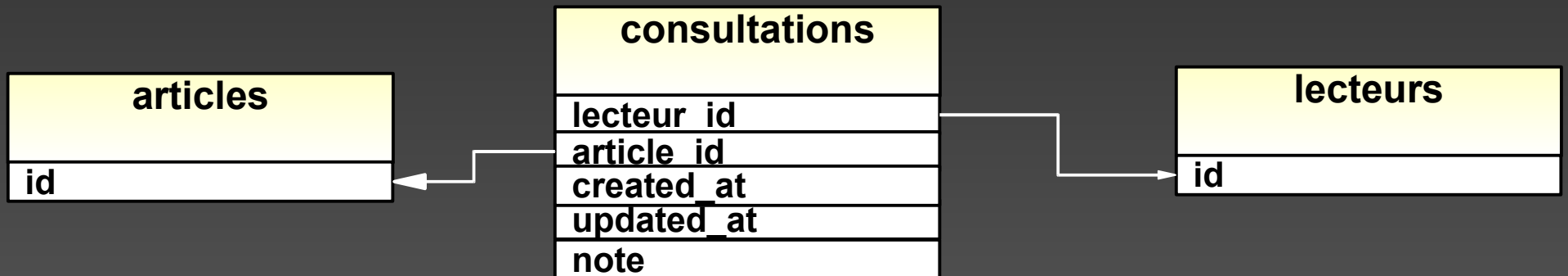
mon_article.commentaires <<
  Commentaire.new( :titre => 'Mon grain de sel' )

un_commentaire.article
```



# Autres associations

- Relation 1:1
  - has\_one, belongs\_to
- Relation N:N
  - has\_many\_and\_belongs\_to (habtm)
- Relation N:N (jointure enrichie)
  - has\_many ... :through => ...





# Contrôleur



# ActionController

Un mini controleur pour l'application Blog

```
class BlogController < ApplicationController

  # http://.../blog/list
  def list
    @articles = Article.find :all
  end

  # http://.../blog/show/6
  def show
    @article = Article.find(params[:id])
  end

  ...
end
```



# Vue



# ActionView

```
# app/views/blog/list.rhtml
<% for article in @articles %>
  <h1><%= article.titre %></h1>
  <p><%= truncate(article.texte, 80) %>&hellip;</p>
  <p><%= link_to "suite..", :action => 'show', :id => article %></p>
<% end %>
```

```
# app/views/blog/show.rhtml
<h1><%= @article.titre %></h1>
<p><%= @article.texte %></p>
<% for comment in @article.commentaires %>
  <%= render :partial => 'comment', :object => comment %></p>
<% end %>
<p><%= link_to "Retour", :action => 'list' %></p>
```



# ActionView

- Le partiel pour l'affichage du commentaire

```
# app/views/store/_comment.rhtml  
  
<p><%= h comment.note %></p>  
<p><%= comment.author %> - <%= comment.created_at %></p>
```





# RJS: Ajax intégré

- Intégration poussée de Prototype et Scriptaculous
  - Pas de Javascript à écrire
- Fonctions Javascript prédéfinies:
  - Champs à completion automatique (`auto_complete_field_for`)
  - Champs éditables (`in_place_edit_for`)
  - Tri et Drag-Drop des éléments DOM
  - Effet visuel (flash, fade in, hide/show, etc...)



# RJS: Ajax intégré

- Templates RJS
  - Génère du code Javascript à la volée au lieu de HTML et l'envoie vers le navigateur pour exécution
- Exemple: insérer du HTML et notifier l'utilisateur

```
# app/views/store/add_comment.rjs

page.insert_html :bottom, 'comment_list',
  "<li id=#{comment.id}>#{comment.title}</li>"
page.visual_effect :pulsate, "#{comment.id}"

page.replace_html 'flash_notice' , "Commentaire ajouté avec succès"

page.show 'flash_notice'
page.delay(3) do
  page.replace_html 'flash_notice' , ''
  page.visual_effect :fade, 'flash_notice'
end
```

end

nuxosgroup





# Les Tests



# Tests Unitaires

- Permet de tester le bon fonctionnement du modèle

```
# test/unit/product_test.rb

class ProductTest < Test::Unit::TestCase

  fixtures :products

  def test_should_refuse_product_with_no_title
    product = Product.new
    assert !product.valid?
    assert product.errors.invalid?(:title)
    assert !product.save
  end

end
```

```
ruby_in_a_nutshell:
  id: 1
  titre: Ruby in a nutshell
  description: Le langage...
  prix: 25.50
```

```
smalltalk_80:
  id: 2
  titre: Smalltalk 80
  description: Smalltalk.....
  prix: 32.00
```



# Tests Fonctionnels

- Permet de tester le bon fonctionnement des actions du contrôleur et le contenu de la vue

```
# test/unit/product_test.rb

class ProductControllerTest < Test::Unit::TestCase

  fixtures :products, :collections

  def test_should_add_an_article_to_the_cart
    product = products(:ror_in_a_nutshell)
    post :add_to_cart, :id => product

    assert assigns(:cart)

    assert_select 'td#prix_total',
                  :text => number_to_currency(product.prix)
  end
end
```

end



# Couverture...

File	Lines	LOC	COV
app/controllers/account_controller.rb	318	246	85.0%
app/controllers/admin_author_controller.rb	92	76	78.9%
app/controllers/admin_collection_controller.rb	70	56	80.4%
app/controllers/admin_controller.rb	19	13	100.0%
app/controllers/admin_parameter_controller.rb	62	48	83.3%
app/controllers/admin_product_controller.rb	296	234	53.8%
app/controllers/admin_publisher_controller.rb	68	54	
app/controllers/admin_subject_controller.rb	67	53	
app/controllers/application.rb	60	37	
app/controllers/billing_address_controller.rb	71	61	
app/controllers/content_controller.rb	102	76	
app/controllers/extract_controller.rb	45	33	
app/controllers/full_content_controller.rb	295	207	
app/controllers/order_controller.rb	357	267	
app/helpers/account_helper.rb	11	2	
app/helpers/admin_author_helper.rb	2	2	
app/helpers/admin_collection_helper.rb	2	2	
app/helpers/admin_helper.rb	2	2	
app/models/collection.rb	22	9	
app/models/content/content_file.rb	37	20	
app/models/content/layout.rb	15	4	
app/models/language.rb	16	6	
.....			
app/models/user.rb	205	140	
app/models/user_notifier.rb	92	75	
app/models/watermark/custom_watermark_parameters.rb	22	9	
...els/watermark/predefined_watermark_parameters.rb	22	8	100.0%
app/models/watermark/watermark_parameters.rb	23	8	100.0%
lib/authenticated_system.rb	120	62	64.5%
lib/authenticated_test_helper.rb	113	83	41.0%
lib/paybox.rb	122	69	78.3%
lib/pdf_ext/lib/pdf_ext.rb	295	214	58.4%
lib/string_extensions.rb	27	16	100.0%
Total	4956	3451	73.3%

**C0 code coverage information**  
Generated on Mon May 22 12:09:23 CEST 2006 with roov 0.4.0  
Threshold: 80%

Name	Total lines	Lines of code	Total coverage	Code coverage
TOTAL	1754	1754	89.2%	60.1%
app/controllers/application.rb	39	39	46.2%	31.0%
app/helpers/application_helper.rb	147	147	30.1%	23.5%
app/models/aggregations/tada.rb	75	75	45.3%	31.1%
app/models/aggregations/upcoming.rb	78	78	48.7%	29.2%
app/models/article.rb	109	109	74.3%	67.9%
app/models/category.rb	30	30	66.7%	60.9%
app/models/sidebar.rb	36	36	55.6%	48.7%
components/plugins/sidebars/archives_controller.rb	35	35	34.3%	29.6%
components/plugins/sidebars/category_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/delicious_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/flickr_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/fortythree_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/fortythreeplaces_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/static_controller.rb	27	27	37.0%	29.2%
components/plugins/sidebars/tada_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/technorati_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/upcoming_controller.rb	20	20	60.0%	50.0%
components/plugins/sidebars/xml_controller.rb	16	16	62.5%	53.8%
components/sidebars/sidebar_controller.rb	110	110	48.2%	32.5%
lib/html_engine.rb	29	29	82.8%	78.3%
lib/login_system.rb	85	85	60.0%	23.5%
lib/migrator.rb	28	28	53.6%	40.9%



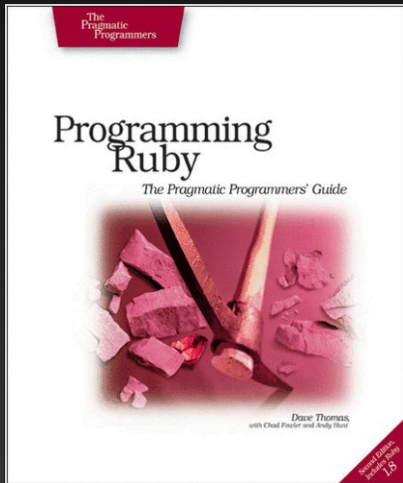


# Références – Les Sites

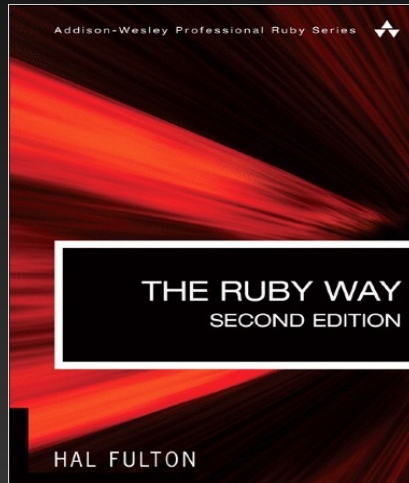
- Sites français
  - Le site Ruby: <http://www.rubyfrance.org>
  - Le site Rails: <http://www.railsfrance.org>
- Sites anglo-saxons
  - LE site Ruby: <http://www.ruby-lang.org>
  - Projets Ruby: <http://www.rubyforge.org>
  - LE site Rails: <http://www.rubyonrails.com/>
  - Pour les développeurs Java: 10 things every Java Programmer should know about Ruby de Jim Weirich (<http://onestepback.org/articles/10things>)



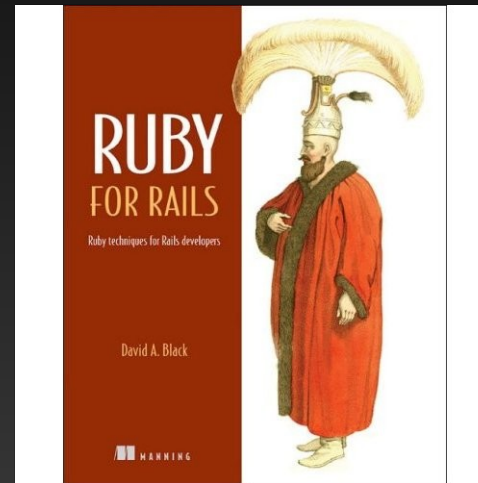
# Ouvrages: les indispensables



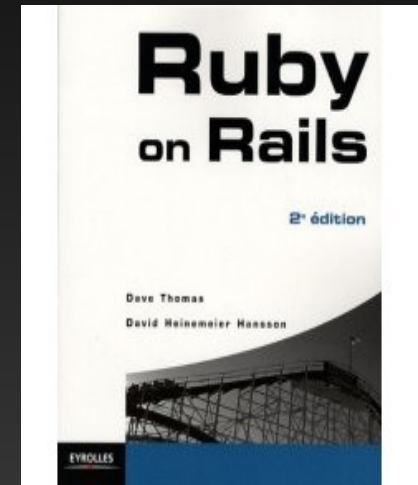
Programming Ruby



The Ruby Way



Ruby For Rails



Ruby on Rails – Eyrolles (en français)

De Ruby

...

A Rails

A ce jour plus de 19 titres en anglais et 8 en français sur Ruby on Rails !!!





# Ouvrages: les gratuits

- Ruby Programming (Pick Axe – 1ère édition)
  - <http://www.rubycentral.com/book/> (couvre Ruby 1.6)
- The Little Book of Ruby Book
  - <http://www.sapphiresteel.com/The-Little-Book-Of-Ruby> (une bonne petite introduction)
- Why's Poignant Guide
  - <http://poignantguide.net/ruby/> (l'informatique, l'humour, l'art et un brin de folie réunis...)



# Questions - Réponses