Présentation de Ruby on Rails

Nicolas Cavigneaux

Juillet 2007

Qui suis-je ?

Nicolas Cavigneaux

Ingénieur d'étude et développement chez webpulser

Spécialisé en Ruby

Amoureux de Rails

Ruby on Rails

Ruby on Rails

- Un framework pour les applications Web

Ruby on Rails

- Un framework pour les applications Web
- Écrit à l'aide de Ruby

Ruby on Rails

- Un framework pour les applications Web
- Écrit à l'aide de Ruby
- Inventé par David Heinemeier Hansson

Ruby on Rails

- Un framework pour les applications Web
- Écrit à l'aide de Ruby
- Inventé par David Heinemeier Hansson
- Souple, élégant, dynamique, agile, productif, complet, …

Ruby !


RAILS


Rencontres Mondiales du Logiciel Libre


webpulser
interactive business engines

Ruby !

- À la base de Rails

Ruby !

- À la base de Rails
- Langage de script objet

Ruby !

- À la base de Rails
- Langage de script objet
- Dynamique

Ruby !

- À la base de Rails
- Langage de script objet
- Dynamique
- Souple

Ruby !

- À la base de Rails
- Langage de script objet
- Dynamique
- Souple
- Syntaxe élégante

Ruby !

- À la base de Rails
- Langage de script objet
- Dynamique
- Souple
- Syntaxe élégante
- Inspiré de Smalltalk et Lisp

Et la syntaxe ?

Et la syntaxe ?

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Et la syntaxe ?

Définition d'une classe

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Et la syntaxe ?

Définition d'une classe

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Appel de méthode avec deux paramètres

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Appel de méthode avec deux paramètres

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Appel de méthode avec deux paramètres

Appel à un itérateur

Et la syntaxe ?

Définition d'une classe

Définition d'une méthode

```
class Voiture < Vehicule

  def avancer(distance)
    deplacer(:avancer, distance)

    distance.times do |i|
      klaxonner
    end
  end

end
```

Appel de méthode avec deux paramètres

Appel à un itérateur

Et Rails ?


RAILS


Rencontres Mondiales du Logiciel Libre


webpulser
interactive business engines

Et Rails ?

- Vue d'ensemble

Et Rails ?

- Vue d'ensemble
- Le modèle MVC

Et Rails ?

- Vue d'ensemble
- Le modèle MVC
  - Le **Modèle** : ActiveRecord

Et Rails ?

- Vue d'ensemble
- Le modèle MVC
  - Le **Modèle** : ActiveRecord
  - La **Vue** : ActionView

Et Rails ?

- Vue d'ensemble
- Le modèle MVC
  - Le **Modèle** : ActiveRecord
  - La **Vue** : ActionView
  - Le **Contrôleur** : ActionController

Et Rails ?

- Vue d'ensemble
- Le modèle MVC
  - Le **Modèle** : ActiveRecord
  - La **Vue** : ActionView
  - Le **Contrôleur** : ActionController
- Développer avec Rails

Pourquoi Rails ?

Pourquoi Rails ?

- Pour prendre le meilleur de ce qui existe à l'heure actuelle !

Pourquoi Rails ?

- Pour prendre le meilleur de ce qui existe à l'heure actuelle !
- Interactivité, simplicité, souplesse

Pourquoi Rails ?

- Pour prendre le meilleur de ce qui existe à l'heure actuelle !
- Interactivité, simplicité, souplesse
- Structuration, infrastructure puissante, évolutivité

Pourquoi Rails ?

- Pour prendre le meilleur de ce qui existe à l'heure actuelle !
- Interactivité, simplicité, souplesse
- Structuration, infrastructure puissante, évolutivité
- À mi-chemin des mondes PHP / J2EE

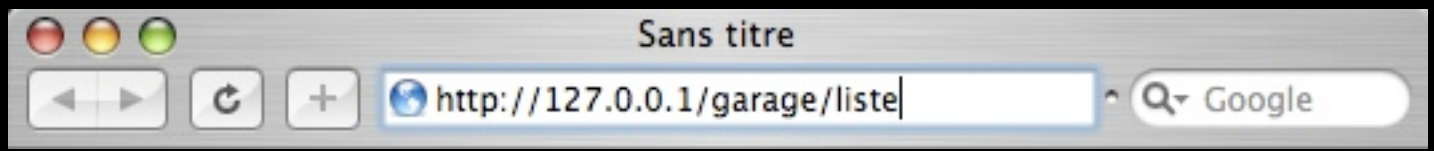
Un exemple


RAILS

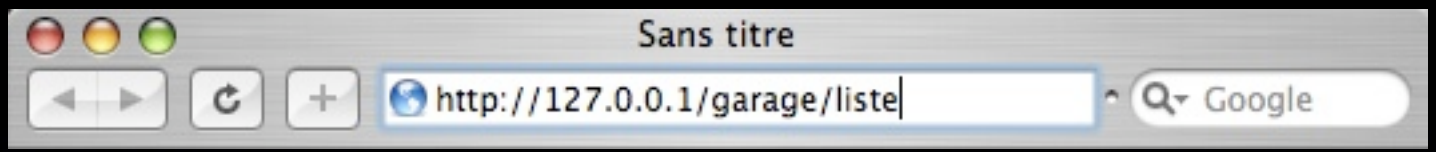

Rencontres Mondiales du Logiciel Libre

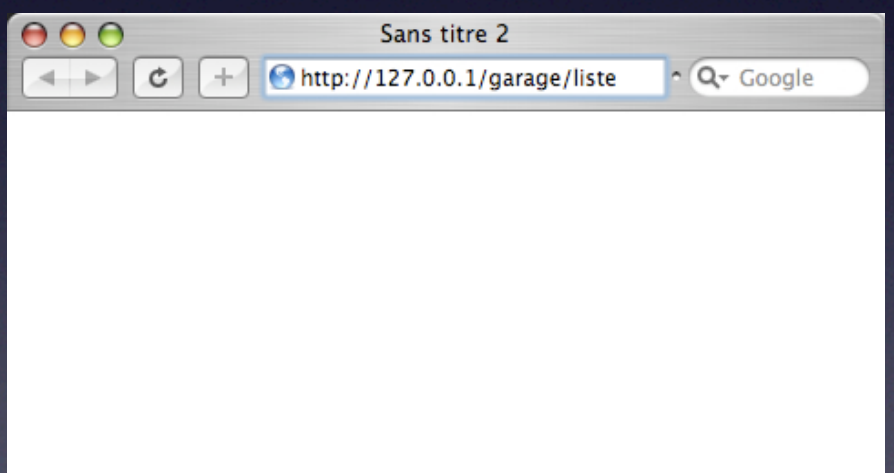

webpulser
interactive business engines

Un exemple

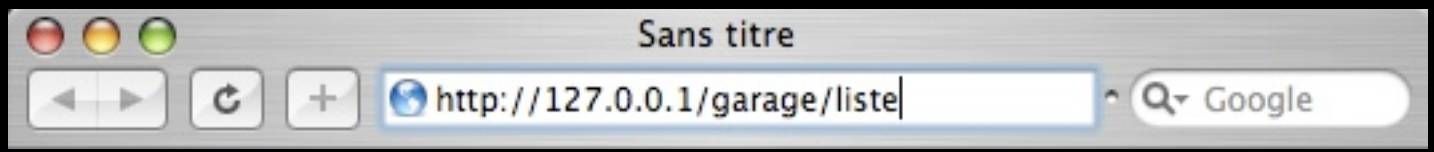


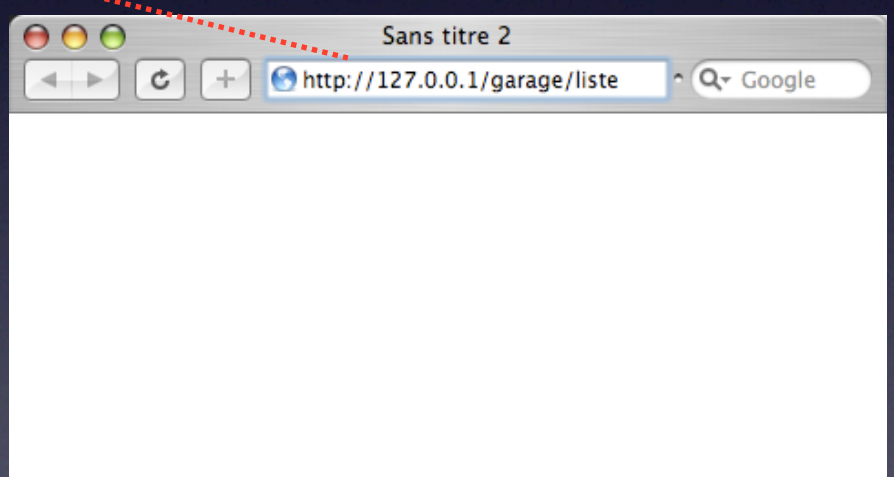
Un exemple


Sans titre
http://127.0.0.1/garage/liste
Google

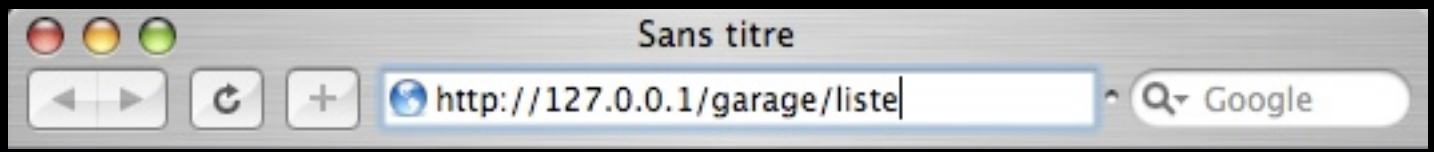

Sans titre 2
http://127.0.0.1/garage/liste
Google

Un exemple

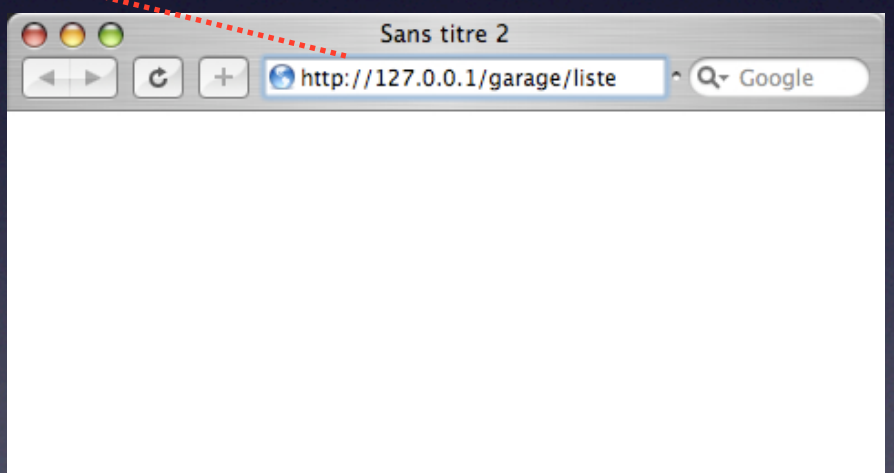

Sans titre
http://127.0.0.1/garage/liste
Google


Sans titre 2
http://127.0.0.1/garage/liste
Google

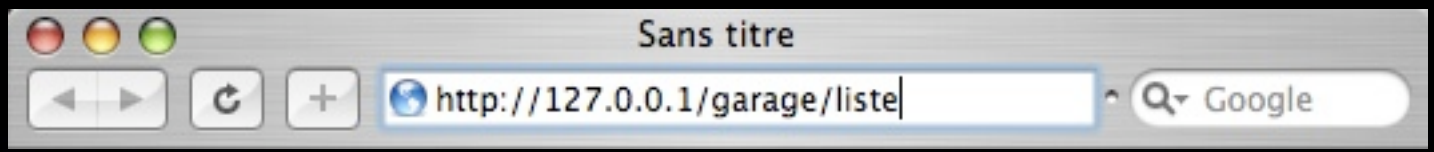
Un exemple



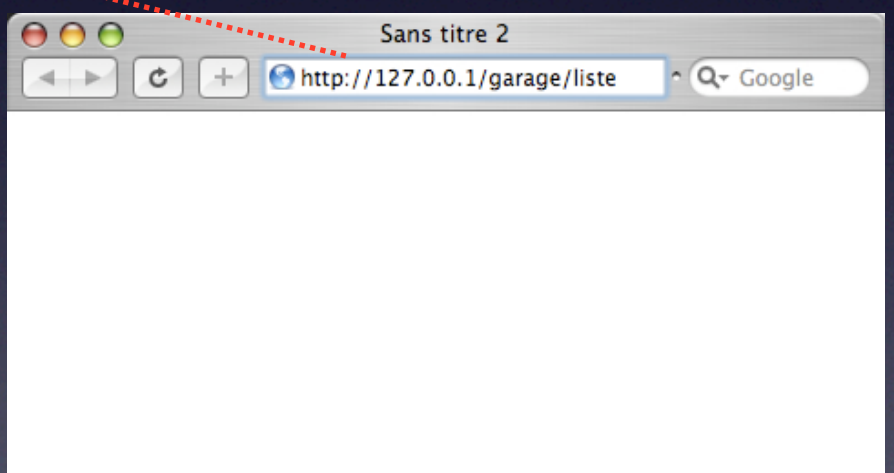

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```



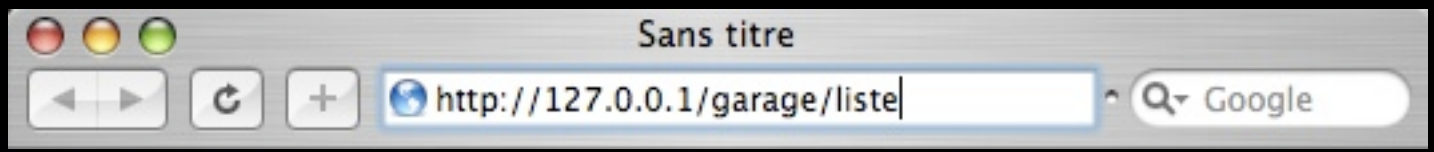
Un exemple




```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```

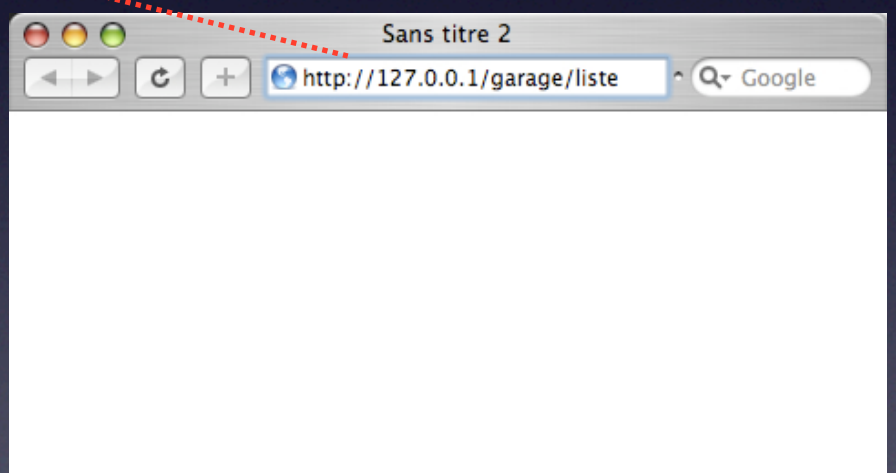


Un exemple

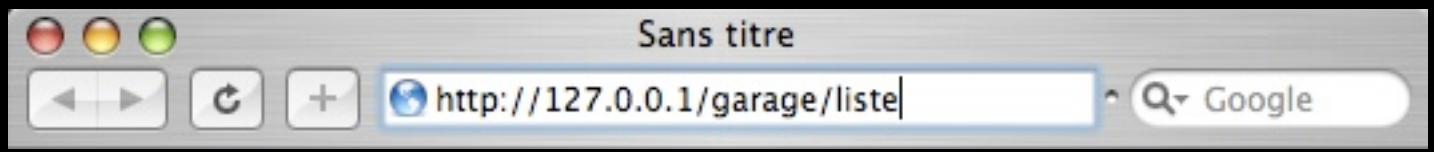



```
class Voiture < ActiveRecord::Base
end
```

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```

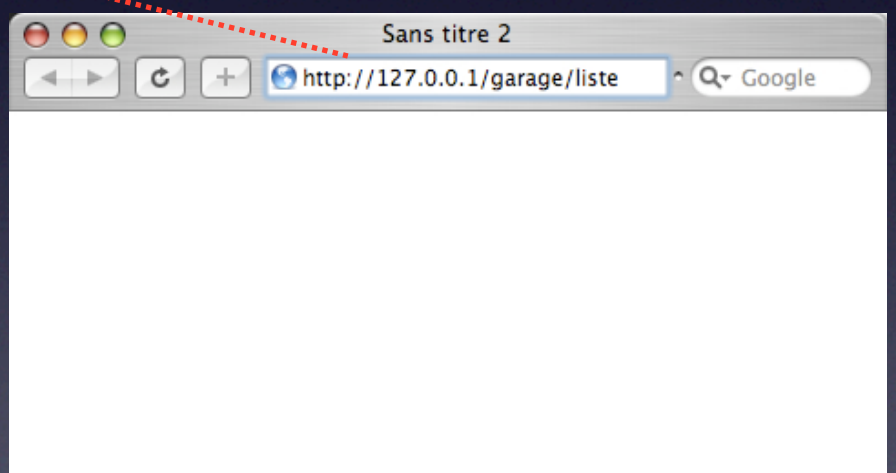


Un exemple

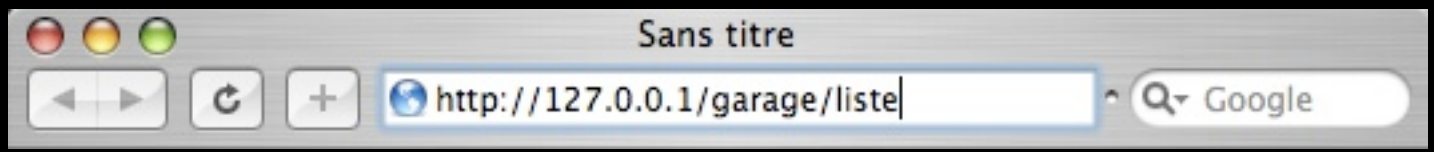



```
class Voiture < ActiveRecord::Base
end
```

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```



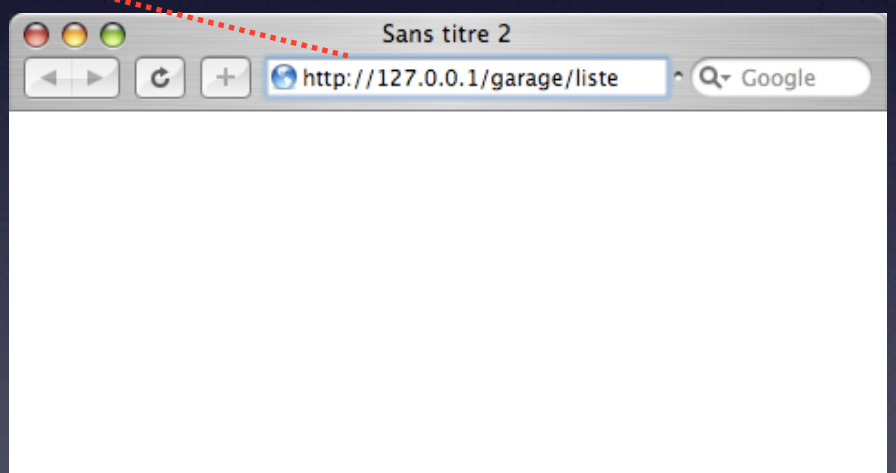
Un exemple



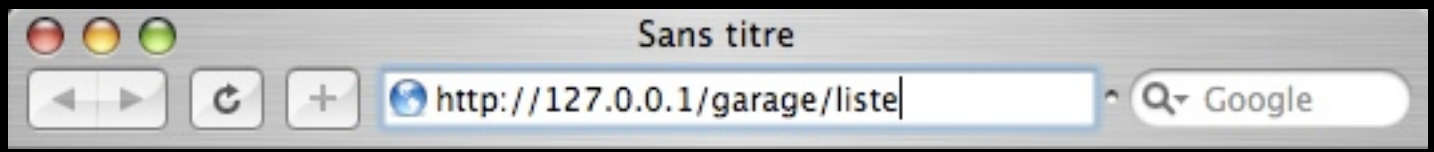

```
class Voiture < ActiveRecord::Base
end
```

id	marque	modele
1	Honda	Civic
2	Nissan	Micra

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```



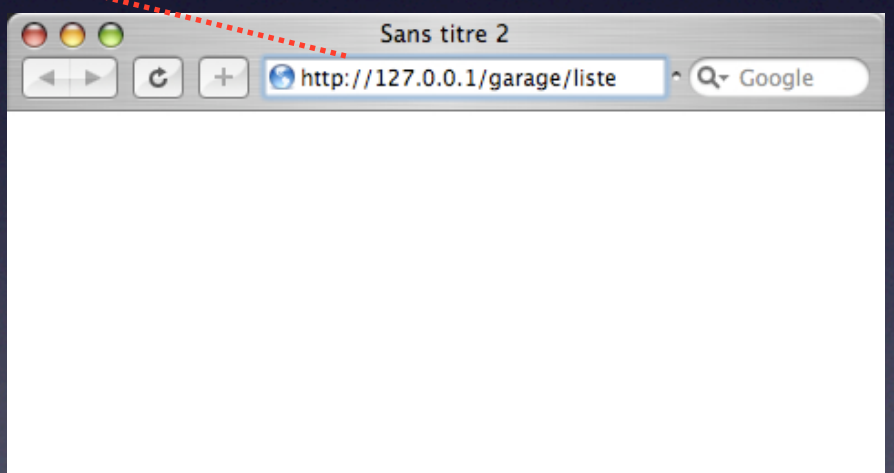
Un exemple



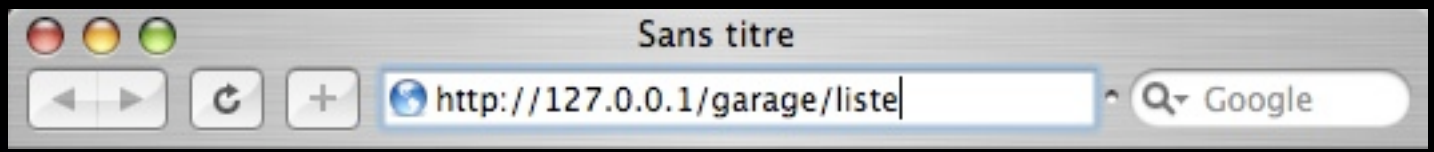

```
class Voiture < ActiveRecord::Base
end
```

id	marque	modele
1	Honda	Civic
2	Nissan	Micra

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```



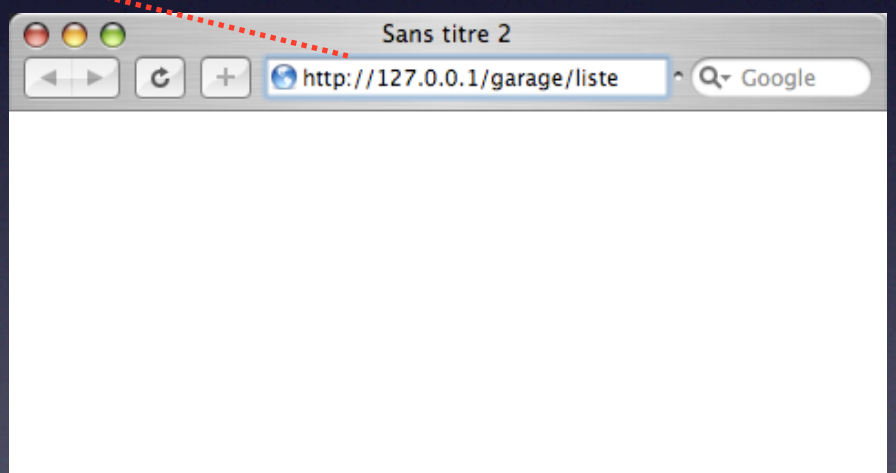
Un exemple




```
class Voiture < ActiveRecord::Base
end
```

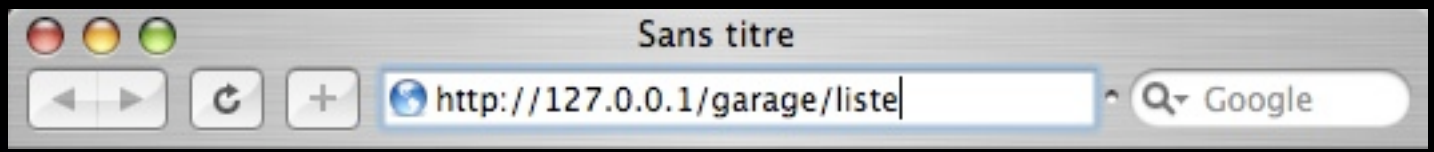
id	marque	modele
1	Honda	Civic
2	Nissan	Micra

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```




```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>
</body></html>
```

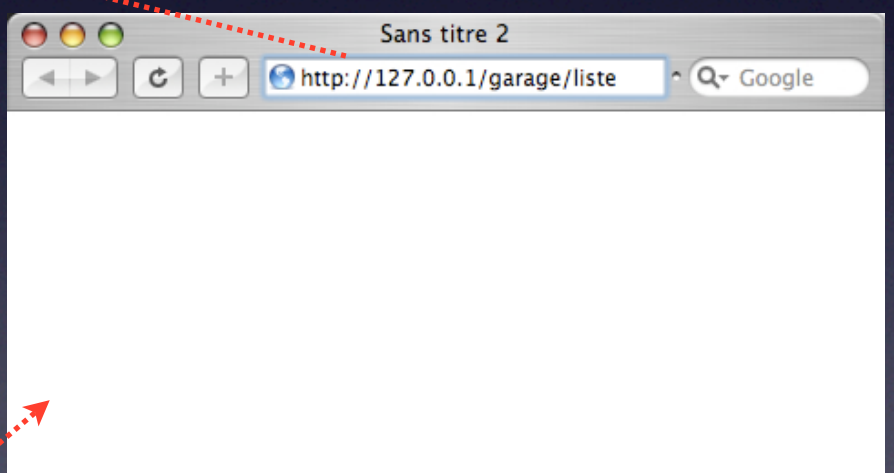
Un exemple




```
class Voiture < ActiveRecord::Base
end
```

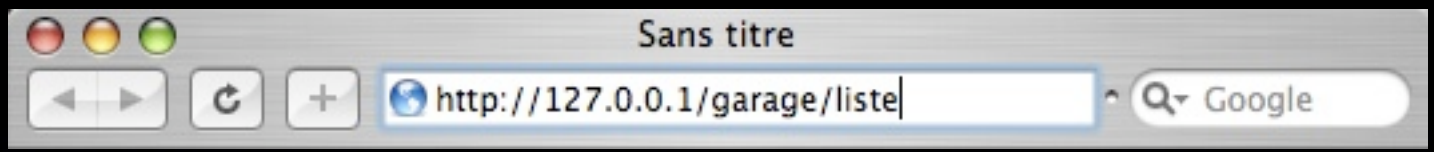
id	marque	modele
1	Honda	Civic
2	Nissan	Micra

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```




```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>
</body></html>
```

Un exemple

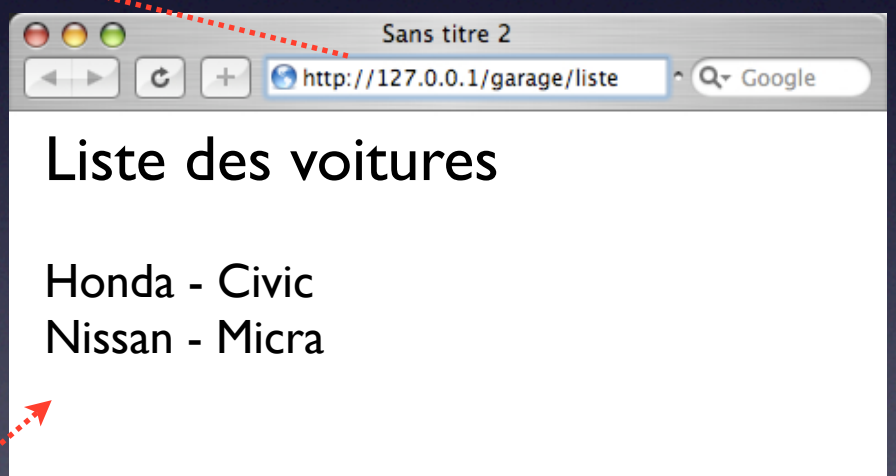



```
class Voiture < ActiveRecord::Base
end
```

id	marque	modele
1	Honda	Civic
2	Nissan	Micra

```
class GarageController
  < ApplicationController
  def liste
    @voitures = Voiture.find(:all)
  end
end
```

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>
</body></html>
```



Définition d'un modèle

ActiveRecord

Les tables correspondent à des classes Ruby → voitures

Chaque colonne correspond à un attribut de la classe

Chaque ligne représente un objet Ruby

ActiveRecord est un **ORM** (Object/Relational Mapper) :

Il assure le lien entre le monde objet de Ruby et le monde relationnel de la base de données.

Définition d'un modèle

ActiveRecord

Les tables correspondent à des classes Ruby

Chaque colonne correspond à un attribut de la classe

voitures

id	marque	modele
1	Honda	Civic
2	Nissan	Micra

Chaque ligne représente un objet Ruby

ActiveRecord est un **ORM** (Object/Relational Mapper) :

Il assure le lien entre le monde objet de Ruby et le monde relationnel de la base de données.

Gestion de l'information

Gestion de l'information

- Rails découvre la structure de la base de données

Gestion de l'information

- Rails découvre la structure de la base de données
- Le développeur respecte quelques conventions

Gestion de l'information

- Rails découvre la structure de la base de données
- Le développeur respecte quelques conventions
- L'objectif est d'éviter la répétition et de simplifier la maintenance

Gestion de l'information

Gestion de l'information

```
v = Voiture.new
v.marque = "Renault"
v.modele = "Clio"
v.save
```

Gestion de l'information

```
v = Voiture.new
v.marque = "Renault"
v.modele = "Clio"
v.save
```

Gestion de l'information

```
v = Voiture.new
v.marque = “Renault”
v.modele = “Clio”
v.save
```

```
INSERT INTO voitures ("marque",
"modele") VALUES('Renault', 'Clio')
```

Gestion de l'information

```
v = Voiture.new
v.marque = “Renault”
v.modele = “Clio”
v.save
```

```
INSERT INTO voitures ("marque",
"modele") VALUES('Renault', 'Clio')
```

```
v = Voiture.find(:first)
puts v.marque

v.destroy
```

Gestion de l'information

```
v = Voiture.new
v.marque = "Renault"
v.modele = "Clio"
v.save
```

```
INSERT INTO voitures ("marque",
"modele") VALUES('Renault', 'Clio')
```

```
v = Voiture.find(:first)
puts v.marque

v.destroy
```

Gestion de l'information

```
v = Voiture.new
v.marque = "Renault"
v.modele = "Clio"
v.save
```

```
INSERT INTO voitures ("marque",
"modele") VALUES('Renault', 'Clio')
```

```
v = Voiture.find(:first)
puts v.marque

v.destroy
```

```
SELECT * FROM voitures LIMIT 1

DELETE FROM voitures WHERE id = 1
```

Gestion de l'information

```
v = Voiture.new
v.marque = "Renault"
v.modele = "Clio"
v.save
```

```
INSERT INTO voitures ("marque",
"modele") VALUES('Renault', 'Clio')
```

```
v = Voiture.find(:first)
puts v.marque

v.destroy
```

```
SELECT * FROM voitures LIMIT 1

DELETE FROM voitures WHERE id = 1
```

Et les associations ?

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

```
class Personne < ActiveRecord::Base
  has_many :voitures
end
```

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

id	marque	modele	personne_id
1	Honda	Civic	2
2	Nissan	Micra	1

```
class Personne < ActiveRecord::Base
  has_many :voitures
end
```

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

id	marque	modele	personne_id
1	Honda	Civic	2
2	Nissan	Micra	1

```
class Personne < ActiveRecord::Base
  has_many :voitures
end
```

id	nom	prenom
1	Dupont	Luc
2	Cavigneaux	Nicolas

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

```
class Personne < ActiveRecord::Base
  has_many :voitures
end
```

id	marque	modele	personne_id
1	Honda	Civic	2
2	Nissan	Micra	1

id	nom	prenom
1	Dupont	Luc
2	Cavigneaux	Nicolas

Et les associations ?

```
class Voiture < ActiveRecord::Base
  belongs_to :personne
end
```

id	marque	modele	personne_id
1	Honda	Civic	2
2	Nissan	Micra	1

```
class Personne < ActiveRecord::Base
  has_many :voitures
end
```

id	nom	prenom
1	Dupont	Luc
2	Cavigneaux	Nicolas

Conventions !

Utilisation

Rails

SQL

Utilisation

Rails

```
p = Personne.find_by_prenom(“Nicolas”)

p.voitures
```

SQL

Utilisation

Rails

```
p = Personne.find_by_prenom(“Nicolas”)

p.voitures
```

SQL

```
SELECT * FROM personnes WHERE
(personnes."prenom" = 'Nicolas')

SELECT * FROM voitures WHERE
(voitures.personne_id = 2)
```

Utilisation

Rails

```
p = Personne.find_by_prenom(“Nicolas”)

p.voitures
```

```
p = Personne.find_by_prenom
(“Nicolas”, :include => :voitures)
```

SQL

```
SELECT * FROM personnes WHERE
(personnes."prenom" = 'Nicolas')

SELECT * FROM voitures WHERE
(voitures.personne_id = 2)
```

Utilisation

Rails

```
p = Personne.find_by_prenom(“Nicolas”)

p.voitures
```

```
p = Personne.find_by_prenom
(“Nicolas”, :include => :voitures)
```

SQL

```
SELECT * FROM personnes WHERE
(personnes."prenom" = 'Nicolas')

SELECT * FROM voitures WHERE
(voitures.personne_id = 2)
```

```
SELECT personnes.id, personnes.nom,
personnes.prenom,
voitures.id, voitures.marque,
voitures.modele,
voitures.personne_id FROM personnes
LEFT OUTER JOIN voitures ON
voitures.personne_id =
personnes.id WHERE (personne.prenom =
'Nicolas')
```

Mais j'ai des contraintes !

Mais j'ai des contraintes !

- ma base existe déjà
- mes tables sont préfixées
- mes index sont spécifiques
- …

Mais j'ai des contraintes !

- ma base existe déjà
- mes tables sont préfixées
- mes index sont spécifiques
- …

Convention ne veut pas dire obligation !

Vous pouvez préciser un comportement spécifique table par table et continuer à construire vos requêtes à la main.

Les contrôleurs

ActionController

Les contrôleurs

ActionController

http://localhost/garage/voir/2

Les contrôleurs

ActionController

http://localhost/garage/voir/2

```
class GarageController < ApplicationController
  def voir
    @voiture = Voiture.find(params[:id])
  end
end
```

Les contrôleurs

ActionController

http://localhost/garage/voir/2

```
class GarageController < ApplicationController
  def voir
    @voiture = Voiture.find(params[:id])
  end
end
```

Finalement , la vue adéquate sera rendue

Que mettre dans le contrôleur ?

Que mettre dans le contrôleur ?

- Logique de l'application

Que mettre dans le contrôleur ?

- Logique de l'application
  - préparer les données à afficher

Que mettre dans le contrôleur ?

- Logique de l'application
  - préparer les données à afficher
  - récupérer les informations d'un formulaire

Que mettre dans le contrôleur ?

- Logique de l'application
  - préparer les données à afficher
  - récupérer les informations d'un formulaire
  - faire des vérifications (identification, …)

Que mettre dans le contrôleur ?

- Logique de l'application
  - préparer les données à afficher
  - récupérer les informations d'un formulaire
  - faire des vérifications (identification, …)
  - gérer les redirections

Que mettre dans le contrôleur ?

- Logique de l'application
  - préparer les données à afficher
  - récupérer les informations d'un formulaire
  - faire des vérifications (identification, …)
  - gérer les redirections
  - récupérer des données externes

Les outils disponibles

Les outils disponibles

- Les filtres

Les outils disponibles

- Les filtres
  - before_filter

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter
- Le cache

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter
- Le cache
  - cache de page

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter
- Le cache
  - cache de page
  - cache d'action

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter
- Le cache
  - cache de page
  - cache d'action
  - cache de fragment

Les outils disponibles

- Les filtres
  - before_filter
  - after_filter
  - around_filter
- Le cache
  - cache de page
  - cache d'action
  - cache de fragment
- Cookies et Sessions

Un exemple

```
class GarageController < ApplicationController
  before_filter :logged_in?, :only => :nouveau
  after_filter :compress
  caches_action :liste

  def nouveau
    if request.get?
      @voiture = Voiture.new
    else
      @voiture = Voiture.create(params[:voiture])
      redirect_to :action => :liste
    end
  end

  def liste
    @voitures = Voiture.find(:all)
  end
end
```

Les vues

ActionView

Les vues

ActionView

- Gabarits HTML avec un peu de code Ruby

Les vues

ActionView

- Gabarits HTML avec un peu de code Ruby
- Accès aux variables d'instances générées par le contrôleur

Les vues

ActionView

- Gabarits HTML avec un peu de code Ruby
- Accès aux variables d'instances générées par le contrôleur
- De nombreux "helpers" pour simplifier l'écriture des balises, formulaires, appels AJAX, …

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to
- link_to_remote

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to
- link_to_remote
- form_tag

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to
- link_to_remote
- form_tag
- text_field

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to
- link_to_remote
- form_tag
- text_field
- date_select

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

- link_to
- link_to_remote
- form_tag
- text_field
- date_select
- …

Un exemple

```
<html><body>
  <h1>Liste des voitures</h1>
  <% @voitures.each do |v| %>
    <%= v.marque %> - <%= v.modele %><br/>
  <% end %>

  <%= link_to('Nouvelle voiture', :action => :nouveau) %>
</body></html>
```

Quelques helpers:

- link_to
- link_to_remote
- form_tag
- text_field
- date_select
- …

Développer avec RoR

Développer avec RoR

- Structuration de l'application

Développer avec RoR

- Structuration de l'application
  - dans le code

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement
  - rapide

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement
  - rapide
  - interactif

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement
  - rapide
  - interactif
  - échaffaudage

Développer avec RoR

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement
  - rapide
  - interactif
  - échaffaudage
  - générateur de code

Développer avec RoR

Les petits plus :

- Structuration de l'application
  - dans le code
  - dans l'arborescence du projet
- Encouragement aux bonnes pratiques
  - tests unitaires
  - conventions de nommage
  - migration des schémas
- Facilite le développement
  - rapide
  - interactif
  - échaffaudage
  - générateur de code

Références


RAILS


Rencontres Mondiales du Logiciel Libre


webpulser
interactive business engines

Références

- **Livres** :

Références

- **Livres** :
  - Programming Ruby

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org
  - http://www.rubyonrails.org

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org
  - http://www.rubyonrails.org
  - http://www.rubyforge.org

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org
  - http://www.rubyonrails.org
  - http://www.rubyforge.org
  - http://www.railsfrance.org

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org
  - http://www.rubyonrails.org
  - http://www.rubyforge.org
  - http://www.railsfrance.org
  - http://blog.webpulser.com

Références

- **Livres** :
  - Programming Ruby
  - The Ruby Way
  - Ruby For Rails
  - Rails Recipes
  - Agile Web Development with Rails
- **Web** :
  - http://www.ruby-lang.org
  - http://www.rubyonrails.org
  - http://www.rubyforge.org
  - http://www.railsfrance.org
  - http://blog.webpulser.com
  - http://www.bounga.org

Des questions ?

Réutilisation du contenu

Les contenus originaux de cette présentation sont diffusés sous licence Creative Commons avec les options :

- Paternité (obligation de mentionner l'auteur)
- Pas d'utilisation commerciale (sans accord explicite)
- Pas de modifications (contenu d'opinion)

La licence complète est disponible à l'adresse :
http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

Toute autre utilisation nécessite un accord explicite et écrit de la part de l'auteur.

Contact : nicolas.cavigneaux@webpulser.com