Ruby on Rails

“Mon préciiiiieux” Gollum

Thibault Normand

Toulibre 2008

28/05/2008

Sommaire

1. Ruby
2. Rails
3. Application
4. Conclusion

Disclaimer

Bonjour à tous

- Présenter Ruby et Ruby on Rails.
- Pas forcément devenir expert programmeur de l'extreme RoR.
- Surtout présenter les concepts nécessaire à la prise en main.

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Ruby

La jeunesse

- Ruby est un langage orienté objet.
- Ecrit par Yukihiro Matsumoto en 1995.
- Sous influence pythonesque, perlienne et lispoise.
- Facile à prendre en main et à mettre en œuvre.
- Syntaxe simpliste.
- Des fonctionnalités de programmation dynamique, et de “métaprogrammation”.
- Un gestionnaire paquet performant *RubyGems*.

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Ruby

Toute une philosophie.

- Tout objet. (Plus proche de la philosophie objet que Python)
- Spécifications ouvertes : IRb, Ruby.Net, JRuby, ...
- Capacité d'embarquement de langage augmentée (Java, C, C++, ...)

Exemples

1. 5.times { print "Hello !" }
2. 5.integer ?
3. a = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
   a.sort !
4. class Livre
   attr_reader :titre, :auteur
   end
   monLivre = Livre.new("Risky", :auteur = "Thibault NORMAND")

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Ruby

La métaprogrammation

- La métaprogrammation : code qui décrit du code = code dynamique.
- Possibilité de modifications dynamiques : ajout d'attribut, modification de comportement en cours d'utilisation SANS RECOMPILATION.
- Mixin & Module.

Exemple

Source animals.rb

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Rails

Pas le premier mais c'est tout comme ...

- Framework de développement Web 2.0 complet.
- Entièrement écrit en Ruby.
- Communauté croissante depuis 2004.
- OpenSource & Free.

Code less, create more ;-)

Axé sur la productivité des développeurs qui l'utilisent.

Initier - Génerer - Executer

```
# rails discoman
```

- Initialise l'architecture de l'application RoR.
- Le serveur d'application est prêt à être executer.

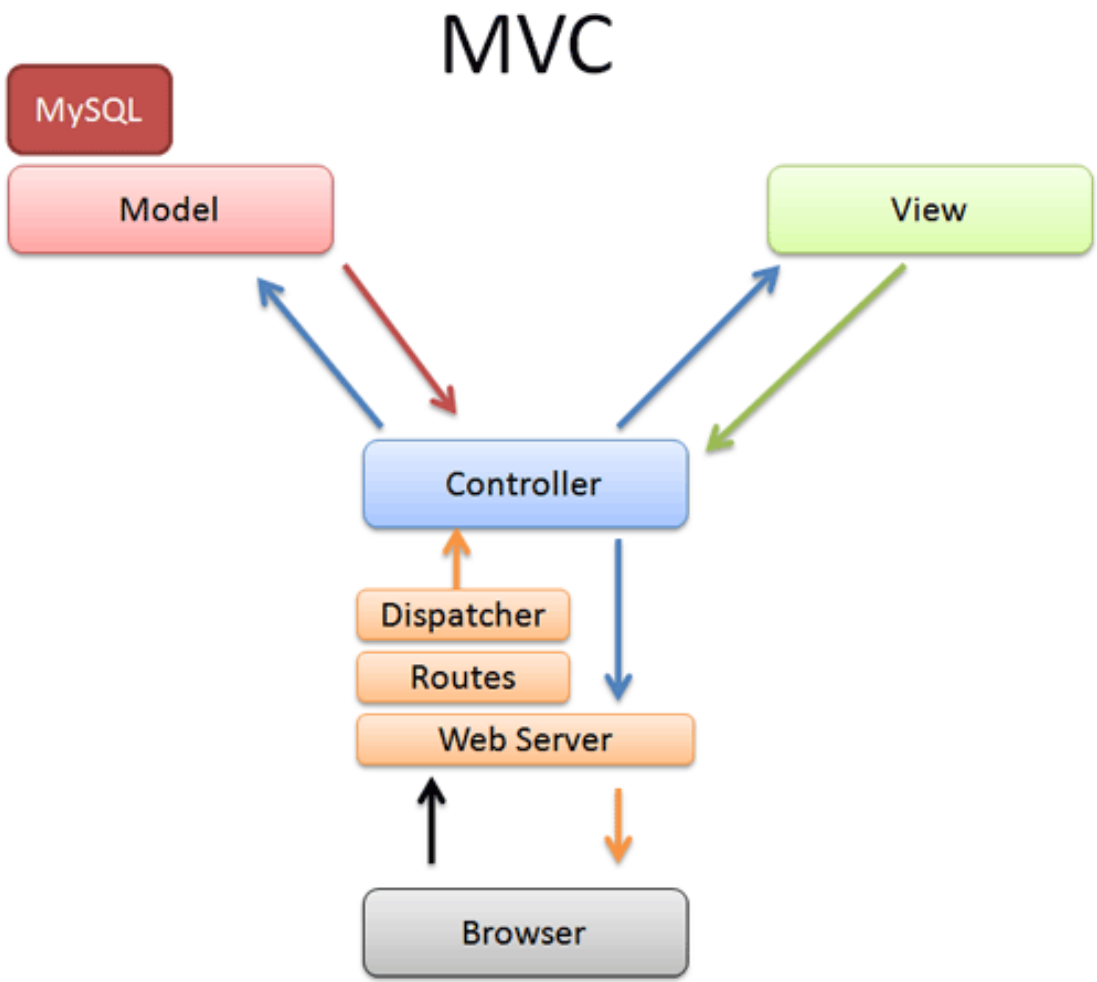
```
# script/server
```

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Rails

MVC : Ma Vieille Chaussette ?



Rails

MVC : Ma Vieille Chaussette ?

Modèle

- Gestion des données. ActiveRecord.
- Gère les relations entre tables.

Vue

- Détermine comment les informations sont affichées.
- Combinaison de Ruby et de HTML.
- Peut varier en fonction des requêtes (Javascript, iPhone, Mobile, etc . . .).

Contrôleur

- Réagit aux requêtes utilisateur.
- Répond à l'aide d'une vue.
- Héberge les opérations à effectuer à partir des données du modèle.

- Beaucoup de choses sont générées à partir des spécifications.

Rails

Generators, merci à vous !

Une personne

est qualifiée par :

- Un nom, de type chaine de caractères (string en ruby).
- Un prénom, de type chaine de caractères.
- Date de naissance, de type date.

Generator

script/generate [type] [name] [name :type]*

migration : opération sur base de données.

model : génère un objet et sa représentation en base (migration).

controller : génère un contrôleur associé à un modèle spécifié.

scaffold : génère une vue automatique, un modèle, sa migration (BD), et le contrôleur associé.

Rails

Generators, merci à vous !

Classe Person

```
# script/generate scaffold Person firstname :string lastname :string bithday :date
```

Génération

1. Migration (Script BD)
2. Modêle (Interface à la BD, et gestion des associations entre modêle(s).)
3. Controleur (Interface entre le modêle et la vue.)
4. Vue (Scaffold) (Interface Homme Machine)

Premier lancement

```
# script/server
```

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Guide

1 Ruby
- La jeunesse
- L'originalité
- La puissance

2 Rails
- Le catalyseur
- Conception MVC

3 Application
- Modèle
  - Domaine
  - Relations
  - Contraintes && Validations
- Vues
  - Dynamic Scaffolding
  - Exemple : ActiveScaffold
- Contrôleurs

4 Conclusion

Rails

Gestion d'une discographie

Domaine

Artist : un artiste d'œuvre.

Producer : un producteur d'artiste.

Song : une œuvre musicale d'un artiste.

Track : la publication d'une œuvre sur un support CD.

Album : l'ensemble des publications.

Rails

Gestion d'une discographie

Producteur

```
# script/generate model producer firstname :string lastname :string dob :date
dod :date
```

Artiste

```
# script/generate model artist firstname :string lastname :string dob :date dod :date
producer :references
```

Oeuvre

```
# script/generate model song name :string version :integer duration :float dop :date
artist :references
```

Rails

Gestion d'une discographie

Piste

```
# script/generate model track number :integer song :references album :references
```

Album

```
# script/generate model album name :string year :integer
```

Céation de la base de données

```
# rake db :migrate
```

- Nous venons de générer les modèles, ils ont été construit à partir des spécifications données précédement.
- Passons aux relations . . .

Relations inter-modêles

Gestion d'une discographie

Domaine

Artist : a composé plusieurs chansons (Songs).
has_many :songs
belongs_to :producer

Producer : produit plusieurs artistes.
has_many :artists

Song : appartient à un artiste.
belongs_to :artist

Track : est le support d'une chanson.
belongs_to :song
belongs_to :album

Album : contient plusieurs pistes (Track).
has_many :tracks

Relations inter-modêles

Petites vérifications

script/console

- # script/console (charge l'environnement interactif de rails pour le déboggage)
- Nous allons créer plusieurs objets à la mano :

```
# p = Producer.new(:firstname => "Bob", :lastname => "Sponge")
# p.save! (pour la sauvegarde en base)
# a = Artist.new(:firstname => "Barry", :lastname => "White",
  :producer => p)
# a.save!
# a.producer == p (?)
```

Contraintes & Validations

Vérifier les données avant utilisation.

- Vérifier la cohérence des données avant enregistrement.
- Appliquer des traitements conditionnels aux entrées.

Validator

validates_presence_of : vérifier que le champs n'est pas vide.

validates_length_of : valide la longueur du champs.

validates_uniqueness_of : valide une propriété d'unicité d'un champs par rapport à une étendu.

Restriction sur le modèle.

Artist : doit avoir un nom ET un prénom.

```
# validates_presence_of :firstname
# validates_presence_of :lastname
```

Track : le numéro de piste doit être unique pour un album.

```
# validates_uniqueness_of :number, :scope =¿ album_id
```

Contraintes & Validations

Vérification

Artiste sans nom

- # a2 = Artist.new(:firstname =¿ "Alan")
  a2.save !
- # *ActiveRecord : :RecordInvalid : Validation failed : Lastname can't be blank*

Numéro Piste identique

- # t2 = Track.new(:number =¿ 1, :song_id =¿1, :album_id =¿ 1)
  t2.save !
- # *ActiveRecord : :RecordInvalid : Validation failed : Number has already been taken*

Fin du modèle

Ouf 2/3 du boulot achevé.

Conclusion

- Les modèles sont générés.
- Les relations et les contrôles d'intégrité des données par rapport au modèle sont à saisir.
- Les modèles sont issus de la base de données. (Migration incrémentale)

Ce dont je n'ai pas parlé

- Relation polymorphique.
- Héritage de table (STI, MTI).

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Génération des vues

Scaffolding hé ? ? ? ? ké ké cé ?

Scaffold

Static Scaffold : “Scaffold” généré par generators.

Dynamic Scaffold : “Scaffold” généré à la requête utilisateur.

Scaffold static

- Générateur de base.
- Interface très sommaire (il peut pas faire le design pour vous non plus !)
- A partir d’un modèle :
  # script/generate scaffold artist firstname :string lastname :string dob :date --skip-migration

Génération des vues

Dynamic Scaffolding

Avantages

- Gagner (encore) du temps sur la réalisation du produit.
- Prototypage très rapide.
- Utilisation pour les parties d'administrations.

Inconvénients

- Peut faire perdre du temps, pour la prise en main.
- Modifications difficiles dans certains cas de personnalisation d'affichage.

Dynamic Scaffold

ActiveScaffold `http://www.activescaffold.com/`
Génération de vues AJAX (CRUD), XML, JSON, ...

StreamLined `http://www.streamlinedframework.org/`
Génération de formulaires dynamiques en fonction du modèle et de métadonnées.

Dynamic Scaffold

ActiveScaffold mon préféré hé !

Principe

- Aucune vue
- Tout est géré à partir du contrôleur.
- Vues HTML (.html), XML(.xml), JSON(.js/.json) par défaut.
- RESTful WebServices. (PUT, GET, DELETE, POST)

Contôleur

active_scaffold :artist (où artist est le nom du model associé au contrôleur).

Exemple

```
# active_scaffold :artist do |config|
#   config.columns = [:lastname,:firstname]
# end
```

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Contrôleurs

Pont de communication entre vues et données.

Principe

- Héberge les fonctions mises à disposition de l'utilisateur.
- Interface entre les données (modèles) et les vues.

Question

Comment faire correspondre un contrôleur à une requête utilisateur ?

Réponse

- En utilisant le concept de route.
- route par défaut : :controller/ :action/ :id. :format.

Guide

1. Ruby
   - La jeunesse
   - L'originalité
   - La puissance
2. Rails
   - Le catalyseur
   - Conception MVC
3. Application
   - Modèle
     - Domaine
     - Relations
     - Contraintes && Validations
   - Vues
     - Dynamic Scaffolding
     - Exemple : ActiveScaffold
   - Contrôleurs
4. Conclusion

Conclusion

Parce que tout à une fin.

Aie !

- Langage à la syntaxe clair mais à la philosophie qui peut faire peur quelques fois.
- On doit acquérir des réflexes : penser que cela existe déja . . .
- Peut devenir un véritable casse tête . . .
- Des soucis de performances à cause du modèle d'exploitation et surtout de l'interpréteur actuel. (cf Twitter)

Mouais !

- Certaines choses sont à revoir : liaison Modèle - BD
- Conventions un peu lourde.

Conclusion

Parce que tout à une fin.

Youpi !

- Productivité hors compétition.
- Séparation MVC aide beaucoup le développeur.
- Myriade de plugins et extensions.
- Des IDE performants gratuit : Aptana (Eclipse)
- Vecteur de propagation de nouvelles technologies (RESTful WS, JSON, ...)

Trouver mieux ailleurs ?

- CakePHP `http://www.cakephp.org/`
- CodeIgnitor (PHP) `http://codeigniter.com/`
- Grails (Java / Groovy) `http://grails.org/`
- Django (Python) `http://www.djangoproject.com/`
- Merb (Ruby) `http://www.merbivore.com/`

Questions ?

Thibault Normand
thibault.normand@gmail.com