IBM Research

```
class WsOnRailsTutorial < Tutorial
```

Web Services on Rails

Using Ruby on Rails for Web Services Development and Mashups

E. Michael Maximilien
http://maximilien.org

Research Staff Member
IBM Almaden Research Center
Almaden Services Research
San Jose, CA USA

```
date “12 September 2006”
last_modified “3 April 2007”
version “0.9.7”
```

Why programming is a good medium for expressing poorly understood and sloppily formulated ideas

Programming provides us with new tools to express ourselves. We now have intellectual tools to describe "how to" as well as "what is." This is a profound transformation: it is a revolution in the way we think and the way we express what we think.

[…] one often hears a student or teacher complain that the student knows "the theory" of some subject but cannot effectively solve problems. We should not be surprised […] The student is never given any instructions on how to abstract from examples, nor is the student given any language for expressing what has been learned. It is hard to learn what one cannot express.

[…] expressing methodology in a computer language forces it to be unambiguous and computationally effective. […] The programmer expresses his/her poorly or sloppily formulated idea in a precise way, so that it (can) become clearer […]

G. Jay Sussman, MIT Professor

Keynote talk at OOPSLA 2005

Title "stolen" from a paper by Marvin Minsky (circa 1960's)

About the speaker

```
def turorial.about_the_speaker
```

- Research Staff Member
- IBM Almaden Research Center
- Almaden Services Research group
- 13 years experience in software research, architecture, design, and engineering
- Research interests in SOA, Web services, service mashups, service ecosystems, OOD/P, SE, and Agile methods and practices
- Ph.D. in computer science from NC State University, Raleigh

```
end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services or APIs (REST, RSS, and SOAP)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

At a glance

```
def tutorial.at_a_glance
```

- Ruby and RoR setup (15 min)
- Intro to Ruby (25 min)
- Break (10 min)

- Intro to RoR (10 min)
- MVC in RoR (15 min)
- RoR exercise and break (20 min)

- Intro to RoR Web services (10 min)
- Exposing an API (REST, RSS, and SOAP) (10 min)
- Consuming APIs exercise and break (15 min)

- Example Web service (or API) mashup (15 min)
- Final exercise (15 min)

```
end
```

Objectives

```
def tutorial.objectives
```

- Get a quick overview of the Ruby language
- Get a quick overview of RoR
  - Setup RoR
  - Create basic RoR application
  - Learn basics of ActiveRecord
- Learn basics of Web services support in RoR
  - Expose RoR application Web APIs
  - Consume Web APIs
  - Compose or mashup Web APIs

```
end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and RoR
- Ruby language overview
- RoR overview
- RoR Web services or APIs (REST, RSS, and SOAP)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

Ruby setup

```
def Ruby.setup
```

- Use tutorial CD
  - Copy folder appropriate to your operating system (Windows, Mac, or Linux)
- Windows
  - *InstantRails* - provides Ruby, Rails, Apache, and MySQL
  - RadRails - Eclipse-based RoR IDE (need to install the rest)
- Mac OS X
  - *Locomotive* - provides Ruby, Rails, Apache, and MySQL
  - Manual install (Google “install ruby on rails on Mac OS X”)
- Linux
  - Most distros come with Ruby, check: **`$ruby -v`**
  - Manual install (Google “install ruby on rails on [your bistro]”)

Ruby setup

- Mac
  - Ruby 1.8.4

```
$tar xvf ruby-1.8.4.tar
$cd ruby-1.8.4
$make
$sudo make install
```

  - MySQL setup
  - RoR

```
$gem install rails --
  include-dependencies
```

  - See http://developer.appler.com/tools/rubyonrails.html
- Linux
  - Ubuntu 6.06 (Dapper Drake)

```
$sudo apt-get ruby
$sudo apt-get mysql
$sudo gem install rails -
  -include-dependencies
```

  - RPM or manual install for others
  - Google “install ruby and rails for [distro]”

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services or APIs (REST, RSS, and SOAP)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

Ruby language overview

```
def Ruby.language_overview
```

- Modern “pure” object-oriented language
- Created by Yukihiro Matsumoto (aka “Matz”) circa 1993
- “Everything” is an object (including numbers)
- Features at a glance
  - Interpreted
  - Dynamic typing (duck typing)
  - Single inheritance
  - Supports mixins
  - Supports blocks or Procs (e.g., for iterators)
  - Rich class libraries (growing)
  - DSL == domain specific languages
- Open source with fast growing community
  - http://ruby-language.org and http://rubycentral.org
  - http://rubyonrails.org

```
end
```

Ruby tools

```
def Ruby.tools
```

- **`ruby`** – Ruby interpreter
- **`gem`** – jar-like packager and installer
- **`gem_server`** – local API documentation at http://localhost:8088/
- **`irb`** – interactive Ruby
- **`ri`** – command line documentation viewer
- **`rdoc`** – JavaDOC-like document generator
- **`rake`** – **`make`**-like tool though more like Apache **`ant`**

```
end
```

Ruby language basics

```
irb = “Interactive Ruby”
def Ruby.language_basics
```

- Variables, console printing, and assignments

```
irb>name = “scc/icws”
=>scc/icsws  #irb and ruby statements return last object
irb>a_string = “Hello #{name.swapcase} from ruby”
irb>puts a_string
```

- Looping

```
irb>10.times { puts a_string }
irb>an_array = [1,2,3,4,5]
irb>for i in  an_array #for i in 1..5
irb>  puts “we are at iteration #{i}”
irb>end
irb>index = 0
irb>while index < 10; puts “we are at iteration #{index}”; index += 1; end
```

- Conditional

```
irb>ruby_is_cool = 1 #You can also use objects true and false
irb>if ruby_is_cool
irb>  puts “Ruby is cool”
irb>end
irb>puts “Ruby is cool” if ruby_is_cool #if, unless, and while modifiers
irb>puts name if a_string #Do you expect scc/icws or SCC/ICWS?
```

Ruby language basics

```
alias RDoc JavaDOC_tool_like_for_Ruby
ri = “Ruby interactive documentation or RDoc”
```

- Classes

```
irb>class Tutorial
irb>  def initialize name
irb>    @name = name
irb>  end
irb>  def name
irb>    @name
irb>  end
irb>  def length
irb>    @length
irb>  end
irb>  def length= num_hours
irb>    @length = num_hours
irb>  end
irb>end
irb>t=Tutorial.new “WS on Rails”
irb>t.name #=> “WS on Rails”
irb>t.length=3
```

- More about classes

```
irb>class Tutorial
irb>  attr_accessor :description
irb>  def objectives
irb>    @objectives ||= []
irb>  end
irb>end
irb>t.description=“Why Ruby and RoR
   are best platform for Web services”
irb>t.objectives<<“learn Ruby basics”
irb>t.objectives<<“learn RoR basics”
irb>t.objectives<<“learn AWS basics”
irb>puts “Tutorial #{t.name} has
   #{t.objectives.size} objectives”
irb>for o in t.objectives
irb>  o.capitalize!
irb>end
irb>t.objectives
```

```
tip “Use ri class|method|class#method|’some text’ to get documentation”
```

Ruby language basics

- Inheritance and overriding methods

```
irb>class WsOnRailsTutorial < Tutorial
irb>  def initialize
irb>    super “Ws on Rails” #calls super class version
irb>    @description = “SCC/ICWS 2006 tutorial #4”
irb>    @objectives = [“Learn ruby”, “Learn RoR”, “Learn AWS”]
irb>  end
irb>end
```

- Private and protected methods

```
irb>class WsOnRailsTutorial < Tutorial
irb>  def attendees
irb>    @attendees ||= {} #creates a hash (map or dictionary) – ruby idiom
irb>  end
irb>  private
irb>  def attendees_grades
irb>    @grades ||= {}
irb>    @attendees.each_key {|key| @grades[key]=“A” if @grades.empty? }
irb>    @grades
irb>  end
irb>end
irb>t = WsOnRailsTutorial.new
irb>t.attendees = {:max => “Michael Maximilien”, :john => “John Doe”}
irb>t.attendees_grades #Causes a NoMethodError exception
```

Ruby language basics

- Variable method arguments

```
irb>class WsOnRailsTutorial < Tutorial
irb>  def initialize *args
irb>    super “Ws on Rails” #calls super class version
irb>    @description = “SCC/ICWS 2006 tutorial #4”
irb>    @objectives = [“Learn ruby”, “Learn RoR”, “Learn AWS”]
irb>    @attendees = args.first if args.first
irb>  end
irb>end
irb>t=WsOnRailsTutorial.new :max => “Michael Maximilien”, :john => “John Doe”
irb>t.attendees #prints hash {:max => ..., :john => ...}
```

- Class variables and class methods

```
irb>class WsOnRailsTutorial < Tutorial
irb>  def initialize
irb>    super “Ws on Rails” #calls super class version
irb>  end
irb>  def material; @@material ||= “http://maximilien.org”; end
irb>  def material= material
irb>    @@material = material
irb>  end
irb>end
irb>t1, t2 = WsOnRailsTutorial.new, WsOnRailsTutorial.new
irb>t1.material=“http://maximilien.org/turorials/2006/ws_on_rails”
irb>t2.material
```

end

Ruby language modules

```
def Ruby.language_modules
```

- Used as namespace mechanism
- Can be included (mixed in) to classes
- Similar to classes but cannot be instantiated
- Ruby's library makes heavy use of modules, e.g., `Enumerable`

```
irb>module MyModule
irb>  def check_positive value
irb>    raise "Value #{value} is not positive!" if value < 0
irb>  end
irb>  def to_s
irb>    "MyModule#to_s"
irb>  end
irb>end

irb>class MyClass
irb>  alias old_to_s to_s
irb>  include MyModule
irb>end
irb>MyClass.new.check_positive -3 #exception raised
irb>MyClass.new.to_s #=>MyModule#to_s
```

```
end
```

Ruby language procs

```
def Ruby.language_procs
```

- Used for iterators and extending methods – very powerful

```
irb>['H','A','L'].collect {|l| l.succ} #=>['I','B','M']
irb>[1,2,3].inject {|sum, i| sum += i } #=>6

irb>File.open("file_name.txt", "r") do |file|
irb>  while line = file.gets
irb>    puts line
irb>  end
irb>end #file is closed and released
```

- Make your methods accept proc

```
irb>def fibonacci max
irb>  i1, i2 = 1, 1
irb>  while i1 <= max
irb>    yield i1
irb>    i1, i2 = i2, i1 + i2
irb>  end
irb>  i2
irb>end
irb>fibonacci(30) {|i| print "#{i} "} #1 1 2 3 5 8 13 21 => 55
```

```
  credit "Two examples from PickAxe book"
end
```

Ruby language arrays and hashes

`def Ruby.language_arrays_and_hashes`

- More about arrays and lists
  - Similar to Java `List`
  - Includes `Enumerable` module
  - For list of supported methods `$ri Array`

    `[]`, `[]=`, `clear`, `empty?`, `include?`, `reverse`, `reverse!`, `sort`, `sort!`, `size`, `slice`, and so on

- More about hashes
  - Collection of `{ key => value}` pairs – keys are usually symbols
  - Access value via: `irb>some_hash[:key]`
  - Non existing keys return `nil` – useful for conditional tests
  - Similar to Java `Map`
  - Includes `Enumerable` module
  - Used widely in RoR
  - For list of supported methods do `$ri Hash`

    `key?`, `keys`, `has_key?`, `has_value?`, `each_pair`, `each_key`, `merge`, `merge!`, `size`, `sort`, `values`, and so on

`end`

Ruby conventions and idioms

```
def Ruby.conventions_and_idioms do |convention, idiom|
```

- Conventions
  - Some enforced (by interpreter) and some are not enforced
  - Generally accepted in community (Ruby and Rails)
- Variables use _ instead of camel case
  - E.g., `this_is_a_variable` vs. `thisIsNotAVariable`
- Methods
  - Instance or class
  - Start with lower case
  - Use "_" to separate words
  - Last statement value is returned
  - E.g., `String` class has methods `capitalize` and also `sort_by`
- Method names can end in:
  - ? for query methods, e.g., `["a", "b", 1].include? 1 => true`
  - ! for methods with side effects, e.g., `"ibm".upcase! => "IBM"`
  - = for assignment methods, e.g., `tutorial.name = "Web APIs on Rails"`
- Operators are supported as method names
- Constants start with upper case, e.g., `Math::PI` and `Math::E`

Ruby conventions and idioms

- Use `nil` for `false` and `true` or also other values in conditional expressions
- Parenthesis are “optional” for some method calls
  - E.g., `-123.abs` is same as `-123.abs()`
  - E.g., `“some string”.sub “s”, “S” => “Some string”` and `“some string”.gsub “s”, “S” => “Some String”`
  - When last parameter is a `Hash`, may omit `{}`, e.g., `Location.create :name => “Silicon Valley”, :city => “Mountain View”`
  - Must use parenthesis when call is ambiguous, e.g., `[“u”, “c”, “s”, “c”].join(“.”).upcase! = > “U.C.S.C”`
  - No space before `()`, e.g., `%w{this is an array}.length()`
- Classes
  - Name start with upper case, e.g., `Hash`, `String`, and `Array`
  - Use camel case, e.g., `IO`, `TrueClass`, `NilClass`, and `ObjectSpace`
  - Instance variables are lower case with _ and start with @, e.g., `@name`
  - Class variables are lower case with _ and start with @@, e.g., `@@count`
- Modules have similar conventions as classes
- Global variables are named with $ prefix
  - E.g., $&, $’, and $` globals for regular expressions matching (as in Perl)

end

Ruby summary

```
def Ruby.summary
```

- What we covered?
  - Basic Ruby: conditional, looping, printing on console
  - Classes, methods, instance variables, class variables
  - Hash, arrays, iterators, multiple assignments
  - Usage of `irb`, and `ri`
  - Some Ruby conventions and idioms

- Other features
  - More about IO, modules, and exceptions
  - Reflection and metaprogramming (e.g., `method_missing`, `define_class, eval, module_eval, class_eval,` and `undef_method`, and so on)
  - Regular expressions, e.g., `/[aeiou]/`
  - `yield` expression and more about iterators and generators
  - More Ruby libraries, e.g., Web, Socket, CGI, REXML, Tk, Thread
  - Creating your own DSLs
  - More Ruby idioms

```
end
```

Break

```
def tutorial.break
```

- 10 minutes
- Optional exercises (use **`irb`**)
  - Experiment with arrays and hashes
  - Execute a **`.rb`** script file
  - Browse Ruby library with **`ri`**
  - Implement **`Array#search`** using iterator
  - Learn about basic metaprogramming – **`ri Kernel#method_missing`**
  - Lean about basic regular expressions – **`ri Regexp`**

```
end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services or APIs (REST, RSS, and SOAP)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

Ruby on Rails

```
alias RoR Ruby_on_Rails
def tutorial.RoR_overview
```

- Brief overview
  - Created by David Heinemeier Hanson (aka DHH) at 37 Signals
  - Abstracted from experience building real-world Web applications
  - Open source (MIT liscense), MVC-based framework, and toolset
- Philosophy
  - Convention over configuration
  - DRY == Don’t Repeat Yourself
  - Do it in Ruby
  - Small tools

```
end
```

RoR features

```
def RoR.features
```

- Metaprogramming, generators, and scaffolding
- Built-in ORM (object relational mapping) via `ActiveRecord`
- Built-in testing via `Test::Unit` (JUnit-like framework)
- `ActiveSupport` which “enhances” Ruby and its libraries
- `ActionPack` framework
  - Mailer – `ActionMailer`
  - Web services and APIs – via XML builder and `ActionWebService`
  - and others, e.g., views with RHTML
- Support for AJAX, YAML, and JSON
- Support for various Web servers

  Apache, Lighttpd, WEBrick, Mongres, and so on
- Support for various relational databases

  DB2, MySQL, MS SQL, Postgres, and so on
- Built-in supports for three environments

  production, development, and test

```
end
```

RoR tools and scripts

```
def RoR.tools_and_scripts
```

- Mac and Linux do **`$./script/<name>`**
- Windows do **`$ruby ./script/<name>`**
- Use **`-h`** for usage information (mostly)

- **Available automatically for any RoR applications**
  - **`rails`** – create a new RoR application skeleton
  - **`script/console`** – runs a Ruby console with RoR libs loaded
  - **`script/server`** – runs the RoR server
  - **`script/breakpointer`** – starts the breakpoint server
  - **`script/generate xyz`** – invoke the `xyz` generator
  - **`script/destroy`** – removes generated code
  - **`script/about`** – prints version of RoR components

  - **`script/performance/benchmarker`**
  - **`script/performance/profiler`**

```
end
```

RoR up and running

```
def RoR.up_and_running
```

- Create a new RoR Web application

  `$rails <application_name>`

- Creates basic application structure
- Start the WEBrick server

  `$cd <application_name>`

  `$./script/server webrick`

- Default server is WEBrick (unless another server is setup)
- Default application port is **`3000 – use –p 80 option to change`**
- Point browser to **`http://localhost:3000/`**

```
end
```

RoR application structure

```
def RoR.application_structure
```

- **app**
- **app/apis** `#Web services`
- **app/controllers** `#All controller .rb`
- **app/helpers** `#Application modules (helpers)`
- **app/models** `#All models (active record)`
- **app/views** `#.rhtml templates`
- **app/views/<controller>** `#.rthml controller templates`
- **app/views/layouts** `#Common layout templates`

- **config** `#Config scripts, e.g., routes.rb`
- **config/environments** `#DB YAML file for each environment`

- **db** `#Any DB .sql and schema.rb`
- **db/migrate** `#DB migration .rb files`

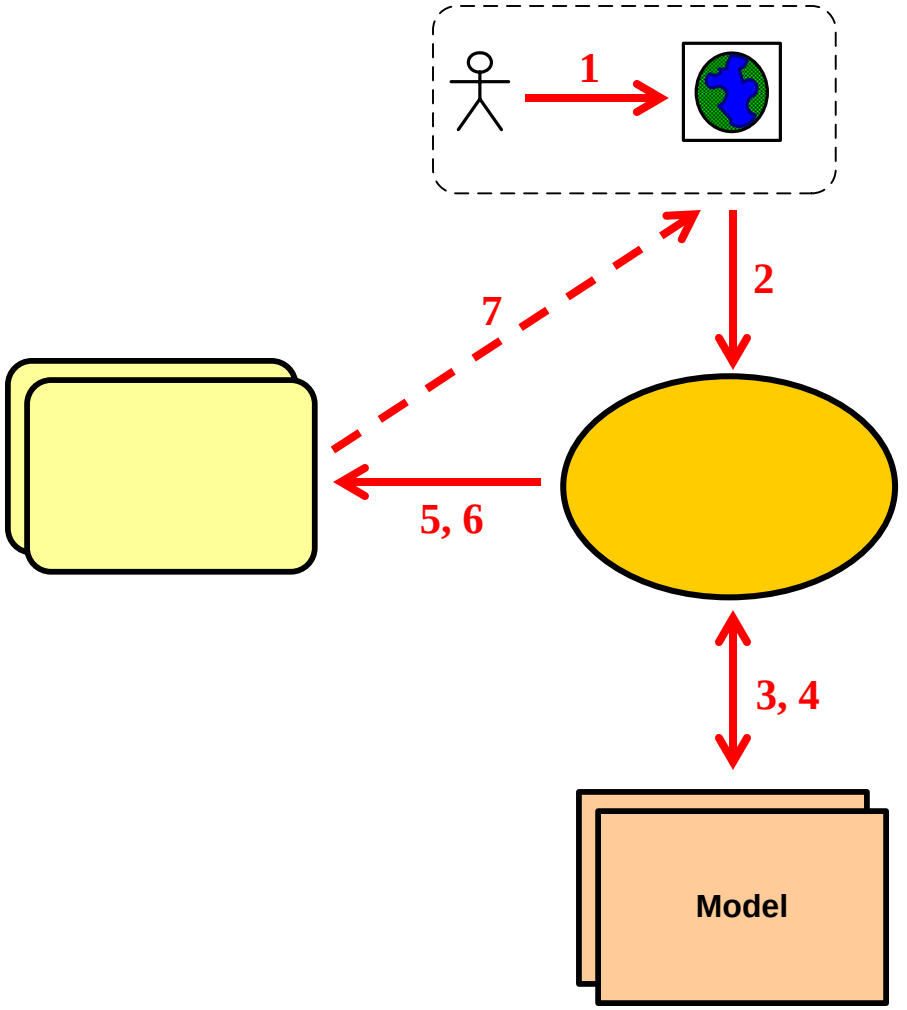
RoR application structure

```
doc                  #Generated RDoc .html
log                  #Log files: production.log, test.log, and
                     #development.log
public               #All static Web application files
public/images        #All images

script               #generator, server, console, breakpointer, ...
script/performance   #benchmarker and profiler
script/process       #spawner and reaper

test                 #Root dir for tests also for helpers
test/fixtures        #.yaml for setting up test fixtures
test/unit            #.rb unit tests
test/functional      #.rb functional tests

tmp                  #Temporary working dir (default session caches)
```

end

RoR ideal MVC architecture

```
def RoR.ideal_mvc
```




```
end
```

RoR ActiveRecords basics

```
def RoR.active_records_basic
```

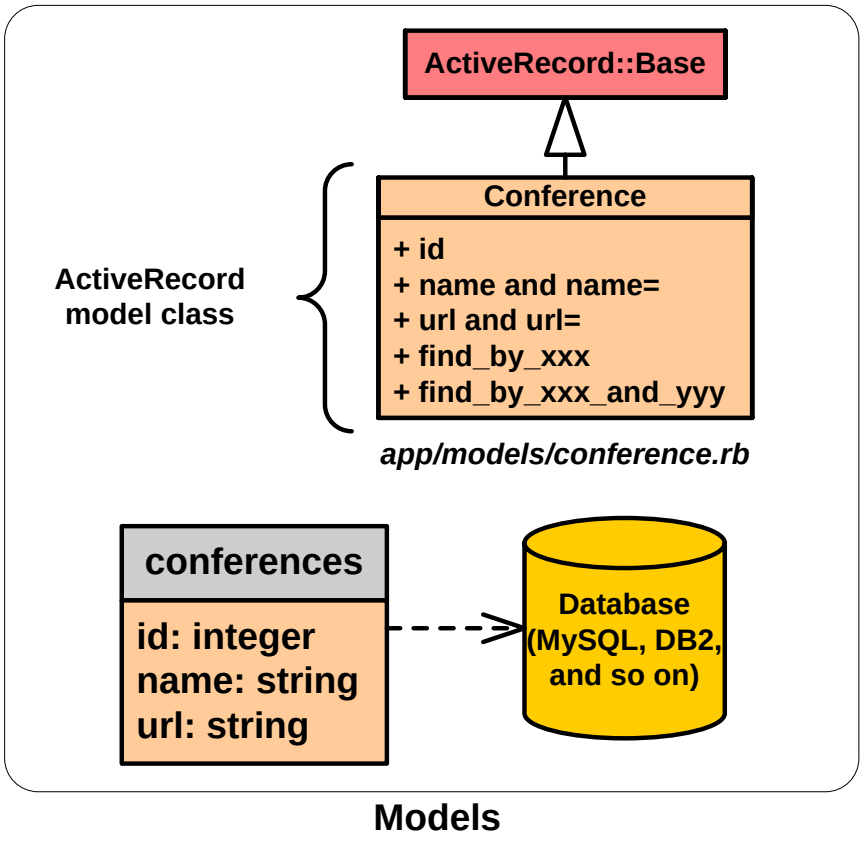
- Object-relational mapping
- Used for RoR models
- Simple conventions
  - RoR model class maps to a DB table
    - **`Employee`** => **`employees`**
    - **`Person`** => **`people`**
  - Camel case singular class names map to tables with "_" name in plural
  - **`ActiveRecord`** class instances maps to table rows
- Use **`ActiveRecord`** DSL methods calls to define relationships between models
- Use **`ActiveMigration`** to create tables in a DB-independent fashion

```
end
```

RoR ActiveRecords mapping

```
def RoR.active_records_mapping
```

- Extend **`ActiveRecord::Base`**
- Located in **`app/models/`** directory
- Attributes added from associated DB table (same name as class but lower case and plural)
- **`id`** attribute is primary key
- All attributes added via metaprogramming
- Finder methods, e.g, **`find_by_id`**, **`find_by_name`**, and **`find_by_sql`**
- Complex classes
  - Inheritance by mapping attributes into one table
  - Composition using one table and special **`composed_of`** DSL call

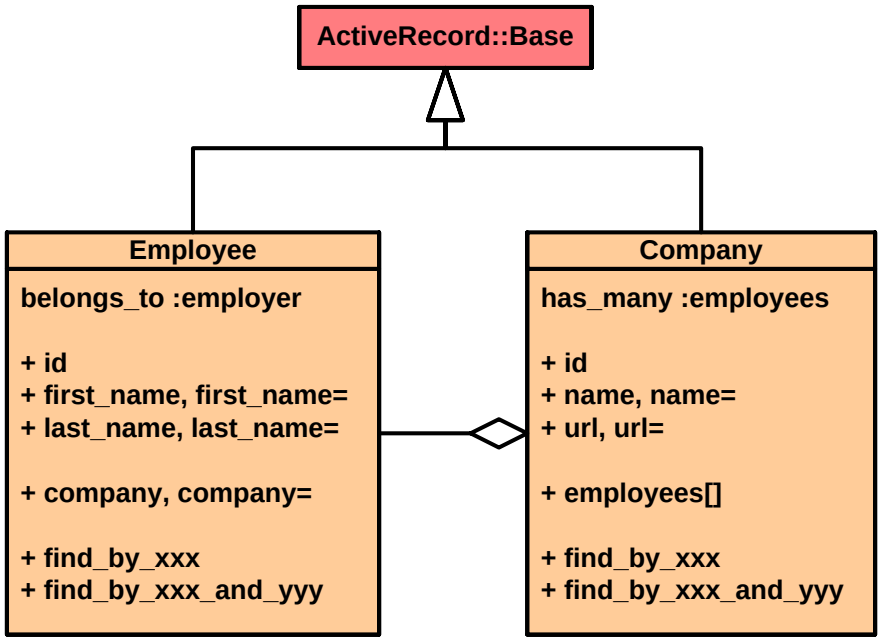


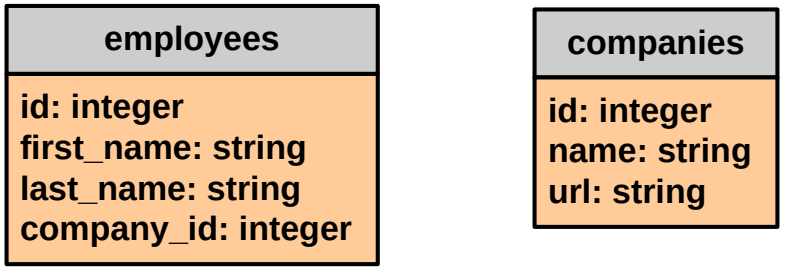

```
end
```

RoR ActiveRecords relationships

```
def RoR.active_records_relationships
```

- One-to-one
  has_one :office
  belongs_to :employee
- One-to-many
  has_many :employees
  belongs_to :company
- Many-to-many
  has_and_belongs_to_many :projects
- List
  act_as_list :scope => "company_id"
- Tree
  act_as_tree



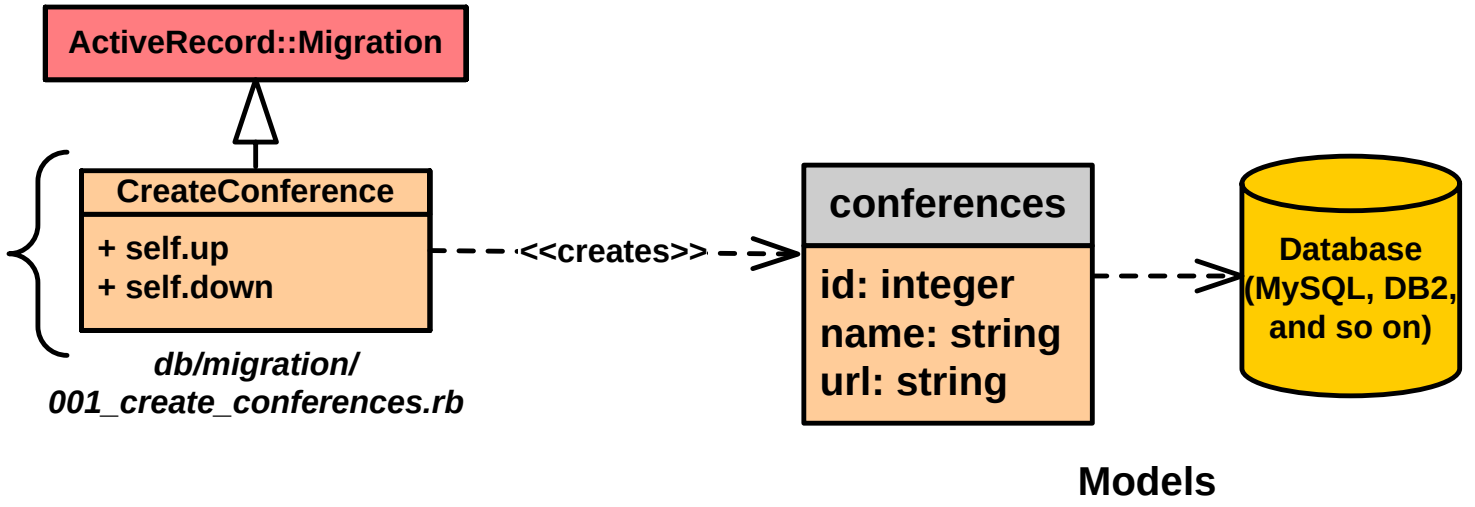



```
  note "Foreign key added to table for model with belongs_to DSL call"
  note "Many-to-many requires one additional association mapping table"
end
```

RoR ActiveRecords migrations

```
def RoR.active_records_migrations
```

- Programmatically define DB tables
- Portable across **`ActiveRecord`** compatible DBs
- Allows for gradual definition of table's schema and easy migration across versions



RoR ActiveRecords migrations

```
class CreateEmployee < ActiveRecord::Migration
  def self.up
    create_table “employees” do |table|
      table.column :first_name, :string
      table.column “last_name”, :string
      table.column :employee_id, :integer
    end
  end

  def self.down
    drop_table “employees”
  end
end
```

- Migration class into **`db/migration/<version>_<names>.rb`**, e.g, **`001_create_employees.rb`**
- Issue command **`$rake db:migrate VERSION=<version>`**
- Can also initialize instances in migration class **`self.up`** method

end

RoR ActiveRecords advanced

```
def RoR.active_records_advanced
```

- Transactions

```
def transfer from_account, to_account, amount
  Account.transaction do
    from_account.debit amount
    to_account.credit amount
  end
end
```

- **`act_as_list`** comes with many methods

```
first?, last?, insert_at_position, remove_from_list, ...
```

- **`act_as_tree`** defines

```
parent_id, children[]
```

- Association methods (added for each association)

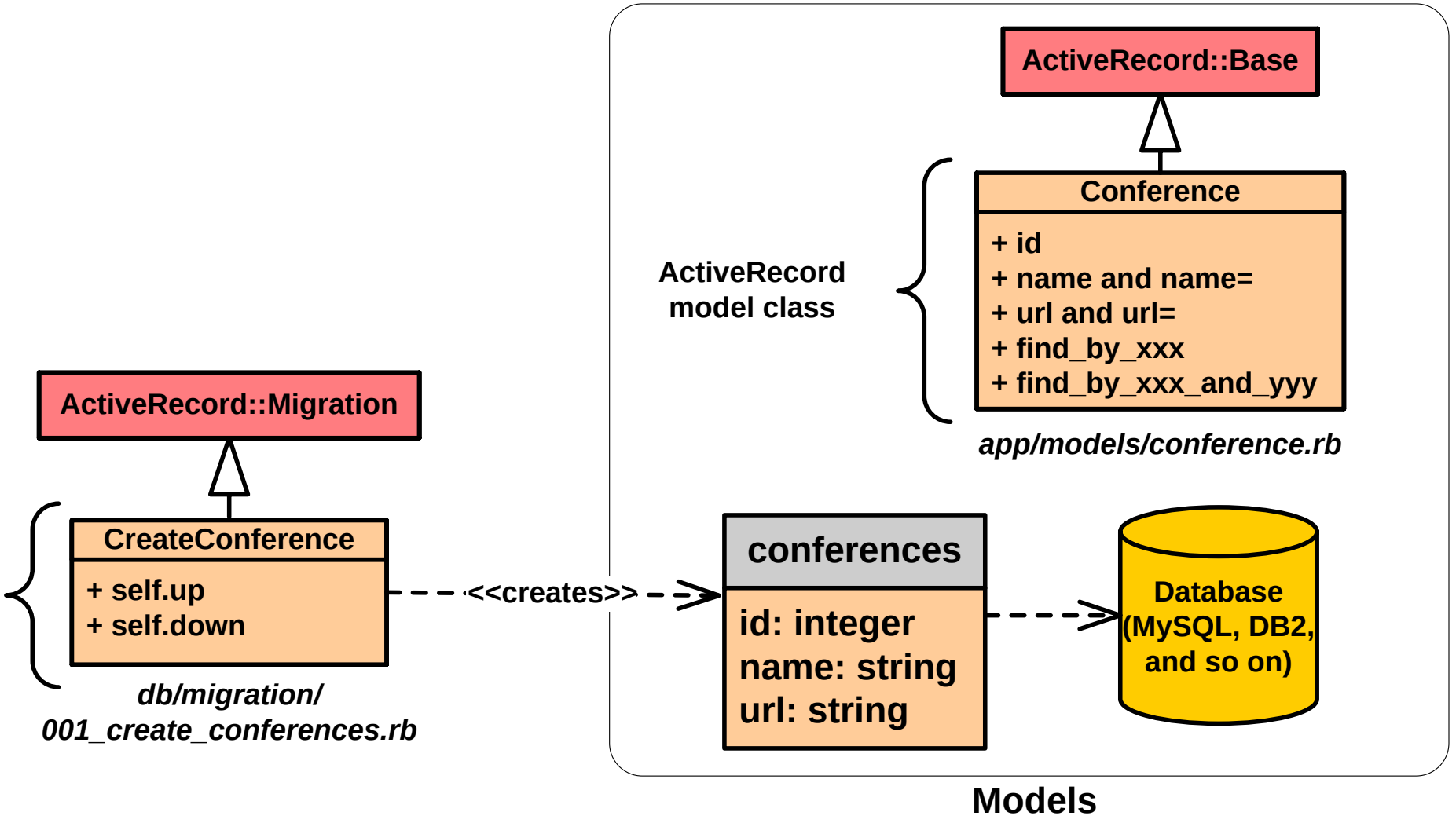
```
<association_name>.create, <association_name>.build,
  <association_name><<, <association_name>.delete,
  <association_name>.find, <association_name>.size,
  <association_name>.empty?, <association_name>.clear, ...
```

- Nested sets for trees, versioning, count caching for faster SQL

```
end
```

RoR models

```
def RoR.models_overview
```




```
end
```

RoR ActionController basics

```
def RoR.action_controller_basics
```

- Manages interactions between models and views
- Generator

```
./script/generate controller Welcome index
  logout
```

```
WelcomeController < ApplicationController
```

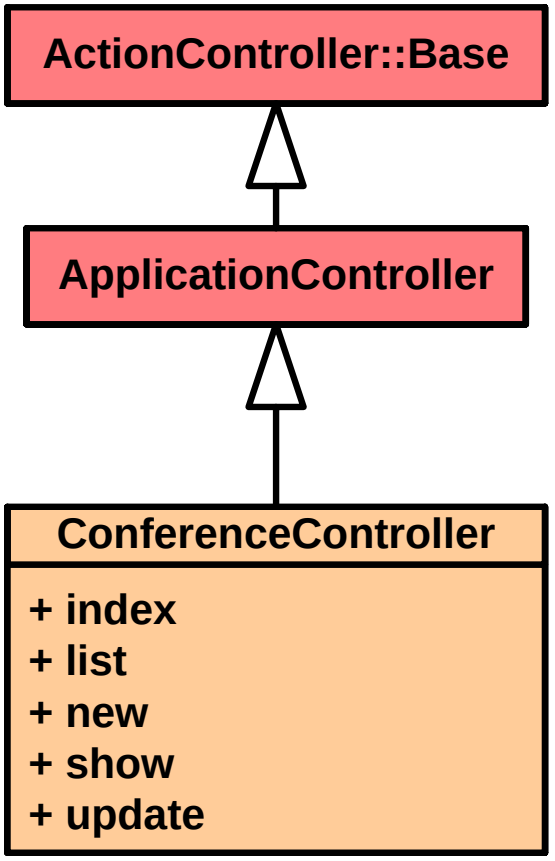
```
ApplicationController < ActiveController::Base
```

- Controller
  - public methods corresponds to actions (no parameters)
  - method names match view templates names
- RoR applications can have many controllers
- Unlike other MVC frameworks, controllers are instantiated per HTTP request (no need to worry about threading)
- Supports basic aspect-oriented programming with **`invoke_before`** and **`invoke_after`** DSL calls

```
end
```

RoR controllers

```
def RoR.controllers
```




```ruby
class ConferenceController <
    ApplicationController

  def index
    render_text "Welcome to ConfApp"
  end

  def list
    @conference_page, @conferences = paginate
    :conferences, :per_page => 10
  end

  def new
    @conference = Conference.new
  end

  def show
    @conference = Conference.find params[:id]
  end

  #...
end
```

```
end
```

RoR views

```
def RoR.views
```

- View template corresponds to a controller action
- Views are implemented using RHTML (similar to JSP and ASP)
- ERb to embed Ruby in HTML

```
<%= link_to( image_tag “image_file.gif”, :size => “50x50”,
                       url_for( :action => “show”,
                                :id => conference.id ) ) %>
```

- Use controller layout for common view parts, i.e., header and footer
- Partial views (starts with _, e.g., **`_form.rhtml`**)
- Full support for CSS, JavaScript, and AJAX

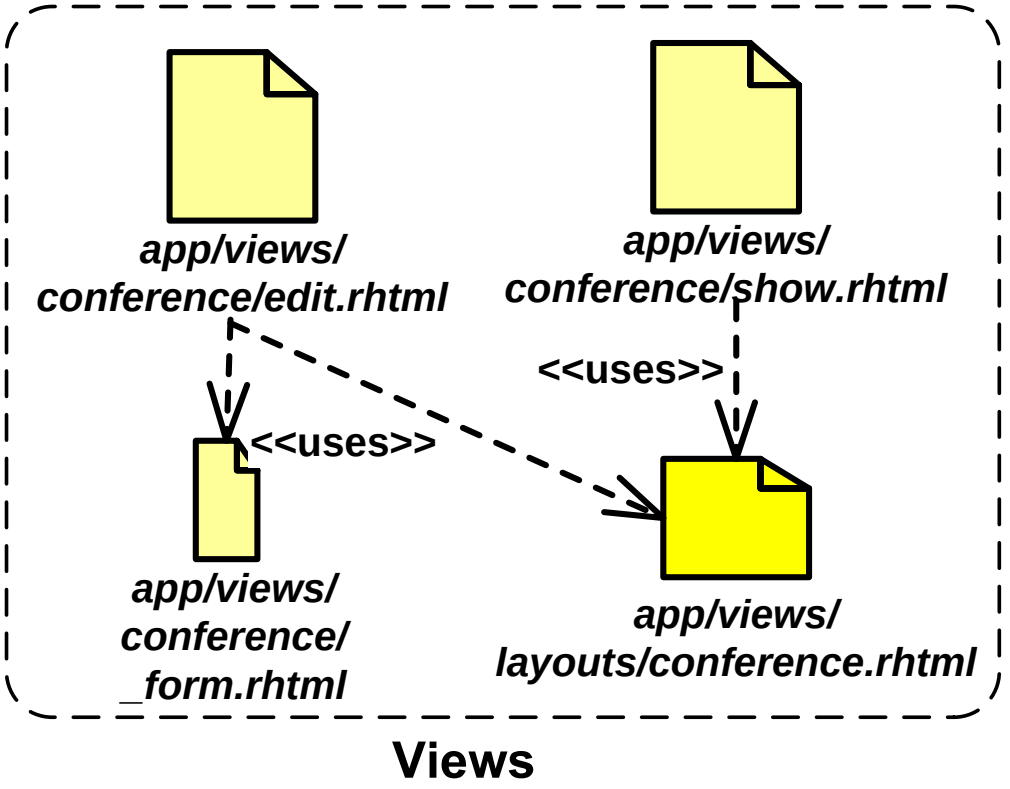
```
end
```

RoR views

```
def RoR.views_overview
```

- Each controller have subdirectory for its views
- Layouts directory with controller common view layout with name of controller
- Partial views with names starting with “_”

```
app\
  …
  controllers\conference_controller.rb
  …
  views\layouts\conference.rhtml
                \conferences\
                      \edit.rhtml
                       \show.rhtml
                       \_form.rhtml
                        \…
```




```
end
```

RoR RHTML basics

```
def RoR.rhtml_basics
```

- Access to all instance and class variables of controller
- Access to all classes visible to controller
- **`h()`** for converting strings safely to HTML
- Use `-%>` to remove “\n” in HTML output when needed
- Various methods (inherited from `ApplicationController`) to generate HTML tags, e.g., **`link_to`**, **`image_tag`**, …

app/views/conferences/list.rhtml

```
<% for column in Conference.content_columns %>
  <p>
    <b><%= column.human_name %>: </b> <%=h @conference.send
   column.name %>
  </p>
<% end -%>
<%= link_to “Edit”, :action => “edit”, :id => @conference.id %>
<%= link_to “Back”, :action => “list” %>
```

```
end
```

RoR views scaffoldings

```
def RoR.views_scaffoldings
```

- Use scaffolding to automatically generate default views
- Generate scaffold for a model and a controller

```
./script/generate scaffold <model> <controller> [<action>
  ...]
E.g., $./script/generate scaffold Conference conferences
  index list edit show
```

- If no actions are specified then `scaffold` generates CRUD actions (create, read, update, destroy) using **`create, show, update,`** and **`destroy`** methods in controller
- Gradually replace scaffold by editing generated views and methods
- Scaffold metaprogramming by adding **`scaffold`** DSL call to controller (no RHTML generated thus cannot edit)

```
class ConferenceController <
  ApplicationController
   scaffold :model_name
   #...
end
```

```
end
```

RoR partial views

```
def RoR.partial_views
```

- Allows decomposition of views
- Enables reuse of views parts
- Great for reusing headers, footers, and forms
- Name of file starts with _, e.g., _form.rhtml

app/views/conferences/_form.rhtml

```
<% errors_messages_for “conference” %>
<!--[form:conference]-->
<% form_tag :action => “update”, :id => @conference.id do %>
  <p><label for=“conference_title”>Title</label><br/>
  <%= text_field “conference”, “title”, :size => 20 %></p>
<% end -%>
<!--[eoform:conference]-->

#Use partials in other RHTML with render :partial => “name” call
render :partial => “form”
```

```
end
```

RoR URL parsing and flow

```
def RoR.url_parsing_and_flow
```

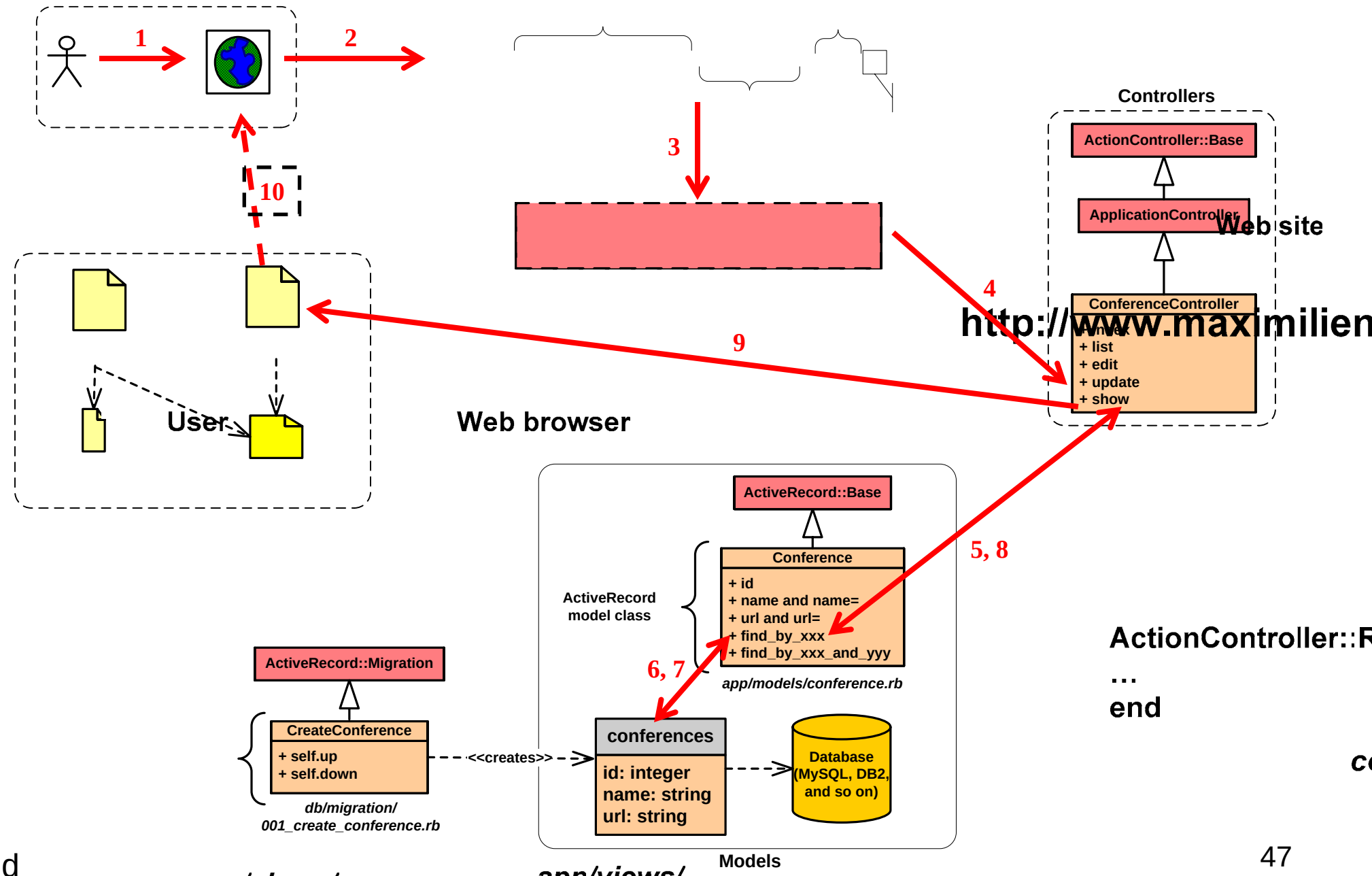
- Controller name is `conference_controller` class is `ConferenceController`
- View name is `show.rhtml` also matches `ConferenceController#show` method
- Parameters passed as a hash to controller method
  - `params` hash for request parameters
  - `session` hash for session data
  - `request` and `response` objects

Web site URL

```
end
```

def RoR.mvc

RoR MVC overview



Ruby on Rails

- What we covered?
  - Rails philosophy, features, and MVC architecture
  - **`ActiveRecords`** basics
    - Models
    - Migrations
    - Relationships
  - `ActionController`
  - RHTML basics
- Other features
  - AJAX via `prototype` and `effect` library integration into Rails
  - CSS support
  - More details about **`act_as_list`** and **`act_as_tree`**
  - RJS, RXML, and ROXML
  - YAML and JSON support

end

Break 2

```
def tutorial.break_2
```

- 25 minutes
- Optional exercises
  - Create **`confapp`** application
  - Add **`Conference`**, **`Attendee, Address`**, and **`Paper`** models
  - Add **`Conference`** controller
  - Add basic scaffolding views
  - Edit **`list.html`** view to make more usable

```
end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services (SOAP, REST, and RSS)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

What are Web services or APIs?

```
def tutorial.what_are_web_services_or_apis?
```

- Basic idea is to make the Web, it's resources, and applications accessible to software agents, i.e., *make it programmable*

- Web services
  - Root in distributed computing, i.e., CORBA, DCOM, messaging systems, etc.
  - Intended to facilitate interoperation across systems and languages
  - Industry definition and support and wide-range of standards, i.e., WS-*

- REST – **re**presentational **s**tate **t**ransfer (Web applications as "virtual state machine")
  - Roy Fielding's Ph.D. thesis at UC Irvine, 2000
  - Simple; uses the Web's architecture to expose services
  - Use HTTP protocol and GET, POST, PUT, and DELETE method for CRUD (create, read, update, and destroy) actions

- RSS – **r**eally **s**imple **s**yndication
  - Syndication for data
  - Feed for Web data on the Web which can be viewed by modern browsers
  - Simple XML format for delivering data with historical context
  - RSS 0.9, 1.0, 1.1, 2.0, and now **Atom**

```
end
```

REST

```
def REST.what_is_it?
```

- Architectural style
  - Conceptualize a Web application as a virtual state machine
  - User moves around states by invoking HTTP methods (GET, POST, PUT, and DELETE)
  - Each state transition results in “new page” (stylized XML content) or application state transfer to user’s browser
- In practice
  - GET and POST are mostly used (use Accept: application/xml header)
  - URI as “operation” names, e.g., http://api.evdb.com/rest/events/search?app_key=test_key&q=mashup
  - Pass operation’s parameters as URL parameters
  - Return response as XML
- Easily supported by any language with HTTP support
- Can be invoked directly from browser via JavaScript
- Limitations
  - Security and privacy (encode data using XML security?)
  - Transactions and recovery
  - Bound to HTTP so cannot use other TCP/IP protocols for large, reliable, or asynchronous transmission, i.e., SMTP, FTP, and so on

```
end
```

REST serving

```
def REST.serving
```

- Add a controller, e.g., `rest_controller.rb`
- Add REST actions, e.g., `RestController#conferences`
- Implement responses using RXML templates using the XML builder library

app/views/rest/conferences.rxml

```
  #@conferences is an array of all Conferences setup in action
  xml.conferences
  @conferences.each do |conf|
    xml.conference(:name => conf.name, :title => conf.title) do
      xml.description conf.description
      xml.attendees
        conf.attendees.each do |attendee|
        #add attendee XML represenation using XML builder
        end
      end
    end; end; end
end
```

REST consuming

```
def REST.consuming
```

- Use Ruby library to make HTTP request formatted according to API documentation
  - Typically use `open-uri`, `rexml`, and `cgi`
- Parse resulting XML using REXML or ROXML into Ruby objects

```
  #Calling the Yahoo! Flickr REST API
  require "open-uri"; require "rexml/document"; require "cgi"
  API_KEY = "<string_key_here>"
  def flickr_call method_name, args_hash={}
    args = args_hash.collect do |key, value|
      CGI.escape(key) + '=' + CGI.escape(value)
    end
    args.join '&'
    url = "http://www.flickr.com/services/rest/?api_key=%s&method=%s&%s"
          % [API_KEY, method_name, args]
    doc = REXML::Document.new open(url).read #Returns results as XML doc
  end
   credit "Inspired from Ruby Cookbook book"
end
```

RSS and Atom

```
def RSS.what_is_it?
```

- Web data (content) syndication XML format
- Includes historical log of data over time
- Includes metadata, e.g., date, author, title, and updated dates
- Examples
  - Text data, e.g., news, blogs, and so on
  - Multimedia, e.g., pictures and video
- Can also be used to syndicate data from a Web service, e.g., calendar entries from Google Calendar
- RSS has gone through various iterations; version 2.0 is latest
- Atom is “standard” version
  - Precise definitions of parts, e.g., date format
  - Official namespace
  - Includes official MIME type: `application/atom+xml`
- Specialized readers and direct browser support to read RSS/Atom feeds directly

```
end
```

RSS and Atom Example

```
def RSS_and_Atom.example
```

```
<?xml version="1.0" ?>
  <rss version="2.0">
  <channel>
    <title>Conferences RSS Feed</title>
    <link>http://localhost:3000/conferences/rss</link>
    <description>Conference RSS feed.</description>
    <language>en-us</language>
    <pubDate>Tue, 03 Apr 2007 04:00:00 GMT</pubDate>
    <lastBuildDate>Tue, 03 Apr 2007 09:41:01
        GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <generator>Ruby on Rails</generator>
    <managingEditor>maximilien@acm.org</managingEditor>
    <webMaster>maximilien@acm.org</webMaster>
     <item>
      <title>RoR Mashup talk at Fujitsu Labs</title>
       <link>http://localhost:3000/conferences/1</link>
       <description>
       Ruby on Rails Mashup talk at Fujitsu Labs in
       Sunnyvale, CA <a
       href="http://maximilien.org/tutorials/2007/ws_om_rail
       s/WS-on_Rails.ppt">PPT slides</a>.
       </description>
       <pubDate>Tue, 03 Apr 2007 09:39:21 GMT</pubDate>
        <guid>
          http://localhost:3000/conferences/1
        </guid>
     </item>
     <!-- other items -->
  </channel>
 </rss>
```

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">

  <title>Conferences Atom Feed</title>
  <subtitle>An Atom feed of Conference data.</subtitle>
  <link href="http://localhost:3000/conferences/atom"/>
  <updated>2007-04-03T18:30:02Z</updated>

  <author>
    <name>E. M. Maximilien</name>
    <email>maximilien@acm.org</email>
  </author>
  <id>
     urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6
  </id>

  <entry>
    <title>RoR Mashup talk at Fujitsu Labs</title>
    <link href="http://localhost:3000/conferences/1"/>
    <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
    <updated>2007-04-03T18:30:02Z</updated>
    <summary>
        Ruby on Rails Mashup talk at Fujitsu Labs in
        Sunnyvale, CA <a
        href="http://maximilien.org/tutorials/2007/ws_om_rail
        s/WS-on_Rails.ppt">PPT slides</a>.
    </summary>
  </entry>

<!-- other entries -->
</feed>
```

```
end
```

Atom feeding

`def Atom.feeding`

- Create `AtomController`
- Add `conference_feed` action and RXML view

app/views/conference_atom/conferences.rxml

```
xml.instruct!
    xml.feed "xmlns" => "http://www.w3.org/2005/Atom" do
      xml.title “Conferences Atom feed"
      xml.link "rel" => "self", "href" => url_for(:only_path => false, :controller => 'atom', :action =>
    ‘conferences_feed')
      xml.id url_for(:only_path => false, :controller => ‘conferences')

      xml.updated @conferences.first.updated_on.strftime("%Y-%m-%dT%H:%M:%SZ") if @conferences.any?
      xml.author { xml.name "#{@user.formatted_name if @user}" }

      @conferences.each do |conference|
        xml.entry do
          xml.title conference.title
          xml.link "rel" => "alternate", "href" => url_for(:only_path => false, :controller => 'conferences',
                                                :action => 'show', :id => conference.id)
          xml.id url_for(:only_path => false, :controller => 'conferences',
                        :action => 'show', :id => conference.id)
          xml.updated conference.updated_on.strftime("%Y-%m-%dT%H:%M:%SZ")
          xml.author { xml.name "#{@user.formatted_name if @user}" }
          xml.summary “Conference summary"
          xml.content "type" => "html" do
            xml.text! render(:partial => "conferences/show", :locals =>{:conference => conference})
end; end; end; end
```

`end`

Action Web Services

```
include RoR::ActionWebServices
alias AWS Action_Web_Services
def AWS.overview
```

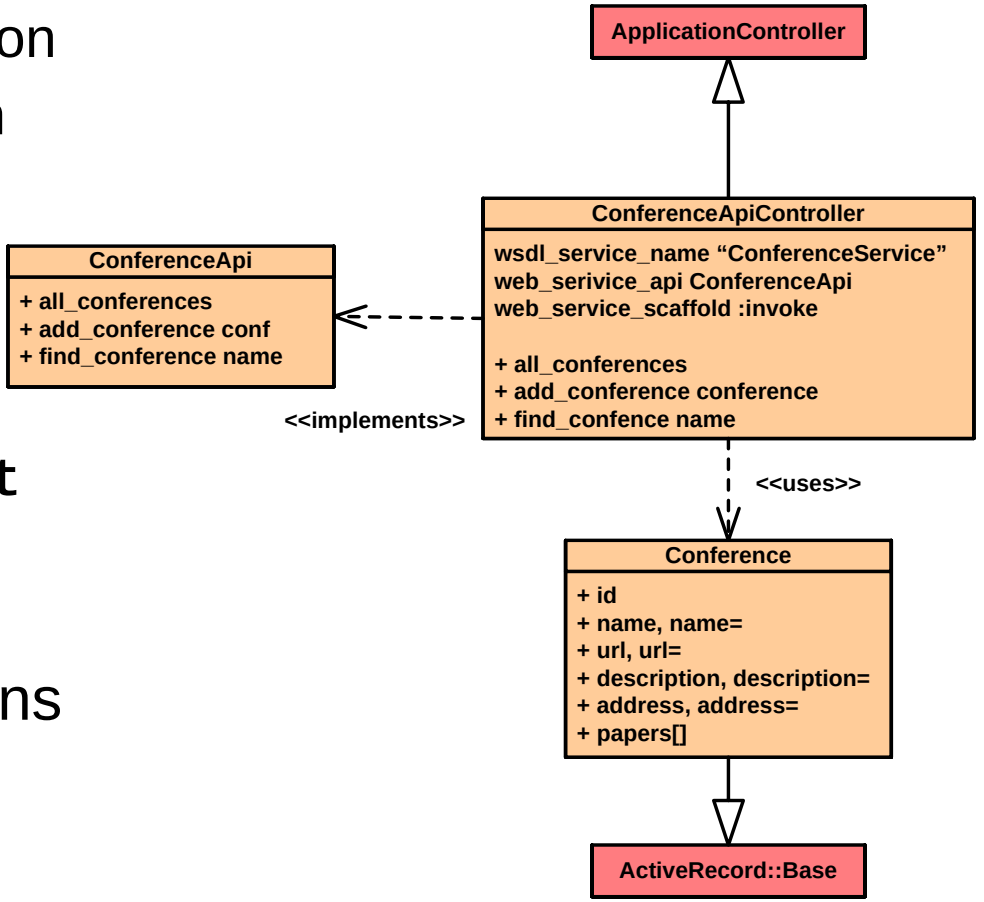
- Originally created by Leon Breedt of Cape Town, South Africa
- Server-side support for SOAP and XML-RPC Web services
- Supports WSDL (but not complete)
- Defines a DSL to define Web service types
- Supports
  - Nested types
  - Arrays
  - Exceptions
- Library to create SOAP/WSDL clients

```
end
```

AWS basics

```
def AWS.basics
```

- Define an API class
  - **`api_method`** for each operation
  - Specify parameters and return types
- Parameter types can be
  - Simple types
  - **`ActiveRecord`** models
  - **`ActionWebService::Struct`** subclasses
- Define dispatching mode
- Supports invocation interceptions
  - **`before_invocation`**
  - **`after_invocation`**
- Supports scaffolding




```
end
```

AWS API interface

```
def AWS.api_interface
```

app/apis/conference_api.rb

```
  class ConferenceApi < ActionWebService::API::Base
    api_method :all_conferences,
               :expects => [],
               :returns => [[Conference]]

    api_method :add_conference,
               :expects => [Conference],
               :returns => []

    api_method :find_conference,
               :expects => [{:name => :string}],
               :returns => [{:conference => Conference}]
    #...
  end

end
```

Conferen

+ all_conferenc

+ add_conferer

+ find_conferer

AWS dispatching mode

```
def AWS.dispatching_mode
```

- Direct
  - API definition attached to controller
  - API methods as controller public methods
- Layered
  - Multiple APIs per controller
  - One endpoint URL for all APIs
- Delegated
  - Like layered but results in a different endpoint for each API (or service)
- **`web_service_dispatching_mode`** DSL call to specify dispatching mode (default is **`:direct`**)

```
end
```

AWS scaffolding

```
def AWS.scaffold_invoke
```

- Add **`web_service_scaffold :invoke`** (or name of method to use) to controller
- Automatically generates views for services
  - Simple pages to invoke each service operations
  - Pass and submit parameters to auto-generated forms
  - Supports SOAP and XML-RPC
- Does not support complex types
- Uses names in API definition for parameter names
- http://localhost:3000/conference_api/invoke
- http://localhost:3000/conference_api/service.wsdl

```
end
```

AWS unit test as client

```
def AWS.unit_test_client
```

- Using **`web_service`** generator
  `./script/generate web_service <name> [action, …]`
- Automatically generates stub unit tests
- Additional Test::Unit methods to **`invoke`** and **`invoke_layered`** services

`test/functional/conference_api_test.rb`

```
#...
class ConferenceApiTest < Test::Unit::TestCase
  fixtures :conferences

  def setup
    @controller = ConferenceApiController.new
    @request = ActionController::TestRequest.new
    @response = ActionController::TestResponse.new
  end

  def all_conferences
    confs = invoke :all_conferences
    assert_equal conferences.size, confs.size
  end
end
```

```
end
```

Consume Web services

```
client.consume(web_service) do |ws|
```

- Java using Axis wsdl2java
  - Generate JAX-RPC stubs pointing to WSDL
    http://site-url.com:3000/controller_api/service.wsdl
  - Use Locator API to create instance of service port
  - Use port to make API calls

- Ruby

```ruby
class SomeController < ApplicationController
  web_client_api :conference_api,
                 :soap,
                 "http:://../conference_api/api" # service's EPR
  def some_method
    confs = @conference_api.all_conferences
  end
end
#...
conference_api = ActionWebService::Client::Soap.new
                      ConferenceApi,
                      "http:://../conference_api/api"
```

```
end
```

What are service mashups?

```
def tutorial.what_are_service_mashups?
```

- Repurposing and remixing Web data and services
- Examples
  - Combining data from multiple sources (search and pictures and music)
  - Combining services, e.g., Eventful event management and Google Calendar and Flickr pictures
- Conceptually can be seen as expanded MVC Web applications
- **Data mediation**
  - Transforming, converting, and manipulating data
  - Create new mixed content
  - Create necessary input to get more data from a service
- **Protocol mediation**
  - Passing correct parameters
  - Correct sequencing of service operation calls
  - Sync and asynchronous service calls
- **UI customization**
  - Create new UI for remixed content and service
  - Deal with service access latency using AJAX

```
end
```

Web services mashup with RoR

```
web_services.each_mashup(other_web_services) do |ws|
```

- Composition of multiple Web services
- Composition of UI – using RHTML
- AJAX user interface
- Apply same principles of service consumption before
  - Expose parts of Web application as REST, RSS, or SOAP API
  - Create proxy clients for remote services
  - Generate composed service using Ruby
- With composed service
  - Expose API for composed service (if necessary)
  - Generate UI views

```
end
```

Break 3

```
def tutorial.break_3
```

- 15 minutes

- Optional exercises
  - Create C**`onference`** API
  - Use scaffolding to test API
  - Finish unit tests
  - Add a Ruby client for API

```
end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services (SOAP, REST, and RSS)
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

Example

```
def tutorial.example
```

- Mashup
  - **`ConferenceApi`** Web service
  - Map conference addresses
  - Use Yahoo! Flickr and Eventful's REST services
  - Use Google Map service
- Generate RHTML views for location photo and Eventful event data
- Generate views using Google Map JavaScript

- Get key for Google Map
- Setup Google Map with default marker
- Create some conference instances in DB or via **`edit.rhtml`** scaffold
- Pass address parts to Flickr, Eventful, and Google Map to search and map address

```
end
```

Example code

```
def tutorial.example_code

end
```

Agenda

```
def tutorial.agenda
```

- Tutorial at a glance and objectives
- Setting up Ruby and Rails
- Ruby language overview
- RoR overview
- RoR Web services action pack
  - Consuming Web services
  - Mashing up Web services
- Example
- Closing remarks and references

```
end
```

Closing remarks

```
def tutorial.closing_remarks
```

- Scratched surface of RoR
- Support for other WS-* standards?
- REST vs. RSS vs. SOAP vs. XML-RPC vs. …
- Various other generators, e.g., login
- Lots more information and documentation online
  - Active community
  - Blogs, videos, articles, and so on
- Scalability
  - Mainly via hardware
  - May be an issue for big sites
  - Next version of Ruby will likely help
- RJS, RXML, ROXML, and other DSLs

```
end
```

References

```
def tutorial.references
```

- http://rubyonrails.org and http://api.rubyonrails.org and http://ruby-doc.org
- http://railshelp.com and http://rails.outertrack.com
- ZenTest at http://www.zenspider.com/ZSS/Products/ZenTest
- RubyCentral, http://rubycentral.org

- Thomas, D. et al. "Programming Ruby" The Pragmatic Bookshelf, 2005 (aka PickAxe book)
- Thomas, D. and Hansson, D. "Agile Web Development with Rails", The Pragmatic Bookshelf, 2005
- Tate, B. and Hibbs, C. "Ruby on Rails: Up and Running", O'Reilley, 2006
- Fowler, C. "Rails Recipes", The Pragmatic Bookshelf, 2006
- Carlson, L. and Richardson, L. "Ruby Cookbook", O'Reilly, 2006

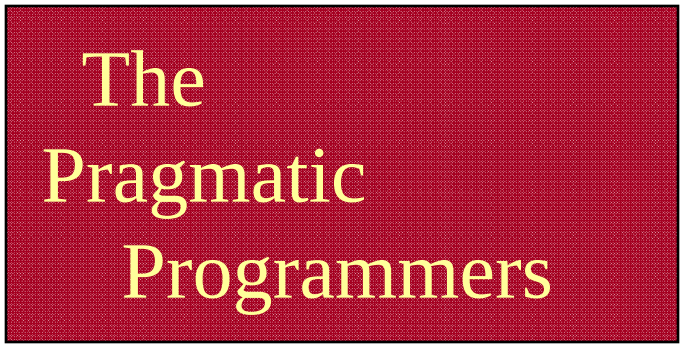
- InstantRails for Windows
- Locomotive for Mac
- RadRails Eclipse-based IDE, http://radrails.org
- Editors see http://wiki.rubyonrails.org/rails/pages/Editors

```
end
```

Acknowledgments

```
def tutorial.acknowledgements
```

http://rubyonrails.org



http://www.pragmaticprogrammer.com

http://eclipse.org

```
end
```

```
def tutorial.thank_you
```

धन्यवाद
Hindi

多謝
Traditional Chinese

ขอบคุณ
Thai

Спасибо
Russian

Gracias
Spanish

شكرا
Arabic

Thank You
English

Obrigado
Brazilian Portuguese

Danke
German

Grazie
Italian

多谢
Simplified Chinese

Merci
French

நன்றி
Tamil

ありがとうございました
Japanese

감사합니다
Korean

```
end; end
```