Ruby on Rails

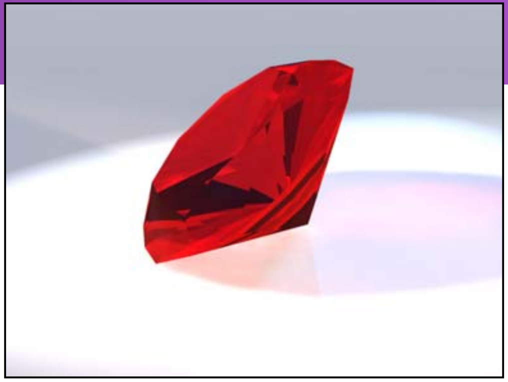
or,

why are you wasting your…

www.rubyonrails.org

Curt Hibbs <curt@hibbs.com>

Agenda



- What is Ruby?
- What is Rails?
- Live Demonstration
- Development Metrics for Real Rails Applications
- Rails Features
- Resources for more information

What is Ruby?

- Short Answer:
  - Ruby is the successful combination SmallTalk's conceptual elegance, Python's ease of use and learning, and Perl's pragmatism.
- Long Answer:
  - Well... see the following slides.

What is Ruby?

“Ruby is a very object oriented language with a super clean syntax that makes programming elegant and enjoyable.”

- James Edward Gray II

“Ruby is an elegant language in which it's easy and natural to express solutions. It's simple enough that a beginner can start using it immediately, yet powerful enough to deal with sophisticated needs. It's so fun that the Puritans would have banned it had they known about it.”

- Paul Sanchez

What is Ruby?

"Ruby is a language that's like the best parts of Smalltalk, Perl and Lisp, all in one, and no line noise."

- Aredridel

"Ruby is the programming language that makes you have more time for your girlfriend .. or less, if you fall in love with Ruby instead."

- Jan Krüger

What is Ruby?

"I use Ruby because it:

- *is easy and fun to learn and use*
- *strongly encourages structured, expressive and readable code*
- *makes object-orientation a natural approach of solving problems*
- *lets you solve the problem at hand instead of fighting against shortcomings of the language*
- *is highly addictive… once you've tried it you cannot imagine life without"*

- Josef Schugt

What is Ruby?

And my personal favorite:

"Ruby? Oh, you won't like this language.

(Slides Pickaxe II book out of view.)

It's entirely too fun and productive for most people."

- Mike Clark

Why Ruby?

- Easy to learn and maintain
- Powerful
- Language stays out of your way
- Rich libraries
- Rapid development
- Helpful community
- Open Source
- Fun

Why Not?

- Performance
  - although it rivals Perl and Python
- Threading model
  - Does not use native threads

Who Uses Ruby?

- Well Known Developer's Using Ruby
  - Evangelists
    - *Dave Thomas & Andrew Hunt*
      - Authors of "The Pragmatic Programmer"
  - Enthusiasts
    - *Ron Jefferies* *
    - *Martin Fowler* *
    - *Jack Herrington*
  - Positive Mentions
    - *Alistair Cockburn* *
    - *Kent Beck* *
    - *Ward Cunningham* *
    - *Bret Pettichord* *
    - *Brian Marick* *
    - *Paul Graham*
    - *Doug Lea*
    - *Bjarne Stroustrup*
    - *Brad Cox*
    - *Bruce Perens*
    - *Howard Lewis Ship*

* Notice the heavy representation from Agile Software Development community.

What is Rails?

- Short Answer:
  - An *extremely* productive web-application framework that is written in Ruby by David Heinemeier Hansson.
- Long Answer:
  - Well… see the following slides.

What is Rails?

- An open source web-application framework.
- It ships with an answer for every letter in MVC:
  - Action Controller
  - Action View
  - Active Record (for the model)

What is Rails?

- Less Code
  - Requires fewer total lines of code than other frameworks spend setting up their XML configuration files.
- Full Stack
  - Being a full-stack framework means that all layers are built to work seamlessly together. That way you Don’t Repeat Yourself (DRY).

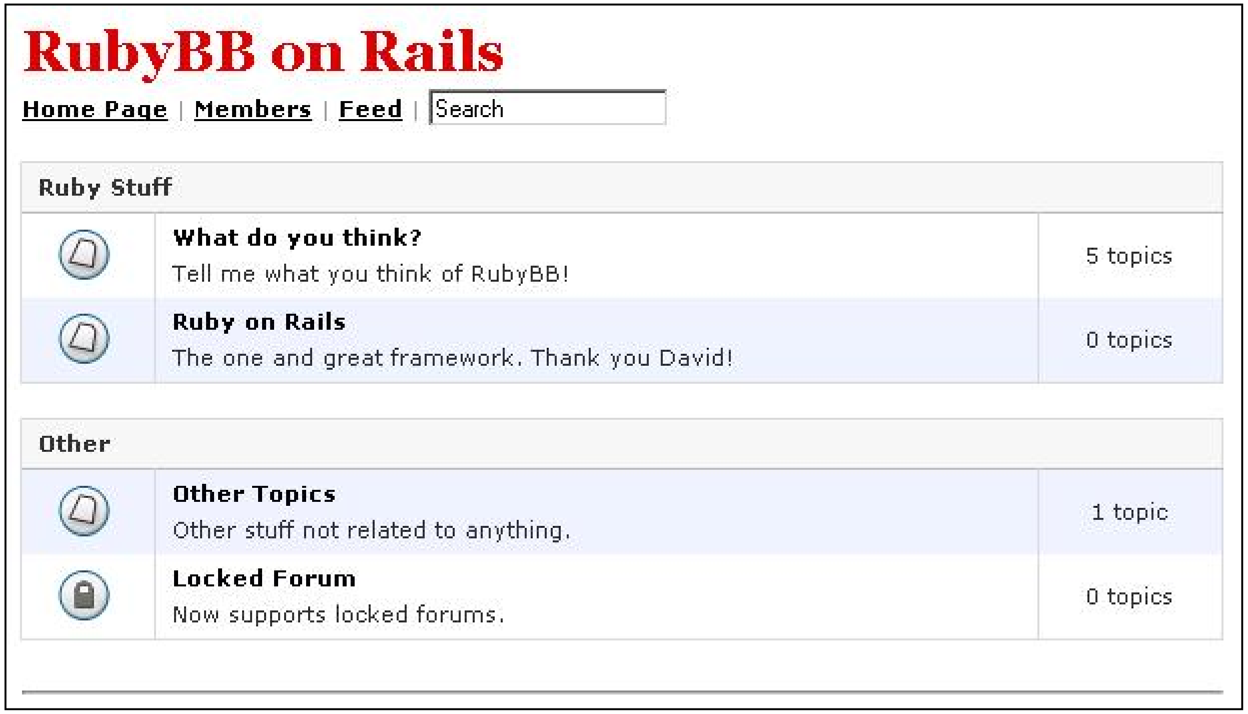
What is Rails?

- Convention over Configuration
  - Rails shuns configuration files in favor of conventions, reflection and dynamic run-time extensions.
- Configure your application by making it
  - Your code *is* the configuration!
  - This means the end of XML files telling a story that has already been told in your code.
  - It means no compilation phase: Make a change, see it work.
  - Meta-data is an implementation detail left for the framework to handle.

Rails Demonstration

The Finished Forum Example

- RubyBB
  - by Russ Smith
  - http://rubybb.readbim.com/
  - 384 Lines of Code



RubyBB on Rails

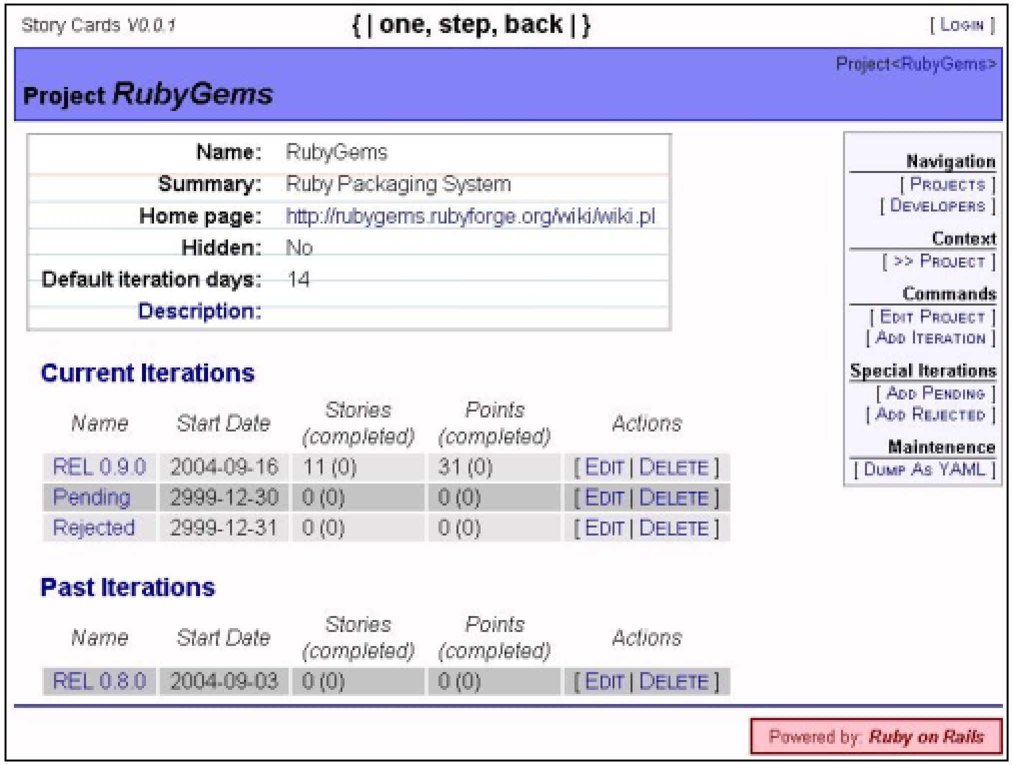
Home Page | Members | Feed | Search

Ruby Stuff		
	What do you think? Tell me what you think of RubyBB!	5 topics
	Ruby on Rails The one and great framework. Thank you David!	0 topics

Other		
	Other Topics Other stuff not related to anything.	1 topic
	Locked Forum Now supports locked forums.	0 topics

Development Metrics

- StoryCards
  - Web app to support XP-style development by Jim Weirich
  - http://onestepback.org:3030/
  - 1,250 Lines of code
  - 8 hours of development time



Story Cards V0.0.1 {| one, step, back |} [Login]

Project<RubyGems>

Project *RubyGems*

Name:	RubyGems
Summary:	Ruby Packaging System
Home page:	http://rubygems.rubyforge.org/wiki/wiki.pl
Hidden:	No
Default iteration days:	14
Description:	

Current Iterations

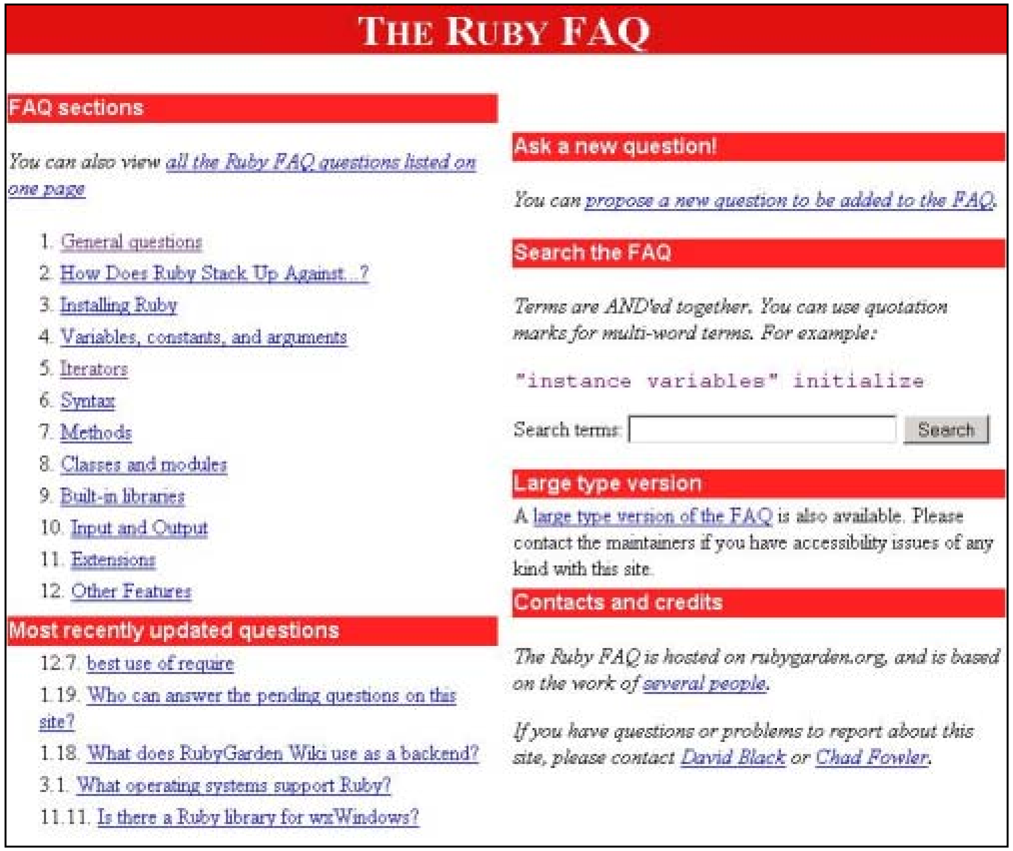
Name	Start Date	Stories (completed)	Points (completed)	Actions
REL 0.9.0	2004-09-16	11 (0)	31 (0)	[Edit | Delete]
Pending	2999-12-30	0 (0)	0 (0)	[Edit | Delete]
Rejected	2999-12-31	0 (0)	0 (0)	[Edit | Delete]

Past Iterations

Name	Start Date	Stories (completed)	Points (completed)	Actions
REL 0.8.0	2004-09-03	0 (0)	0 (0)	[Edit | Delete]

Development Metrics

- RubyFAQ
  - User contributed and commented FAQs (a production web-app) by David Black
  - http://www.rubygarden.org/faq/main/index
  - 573 Lines of code
  - 5 hours of development time.



THE RUBY FAQ

FAQ sections

You can also view all the Ruby FAQ questions listed on one page

1. General questions
2. How Does Ruby Stack Up Against...?
3. Installing Ruby
4. Variables, constants, and arguments
5. Iterators
6. Syntax
7. Methods
8. Classes and modules
9. Built-in libraries
10. Input and Output
11. Extensions
12. Other Features

Most recently updated questions

12.7. best use of require
1.19. Who can answer the pending questions on this site?
1.18. What does RubyGarden Wiki use as a backend?
3.1. What operating systems support Ruby?
11.11. Is there a Ruby library for wxWindows?

Ask a new question!

You can propose a new question to be added to the FAQ.

Search the FAQ

Terms are AND'ed together. You can use quotation marks for multi-word terms. For example:

```
"instance variables" initialize
```

Search terms: Search

Large type version

A large type version of the FAQ is also available. Please contact the maintainers if you have accessibility issues of any kind with this site.

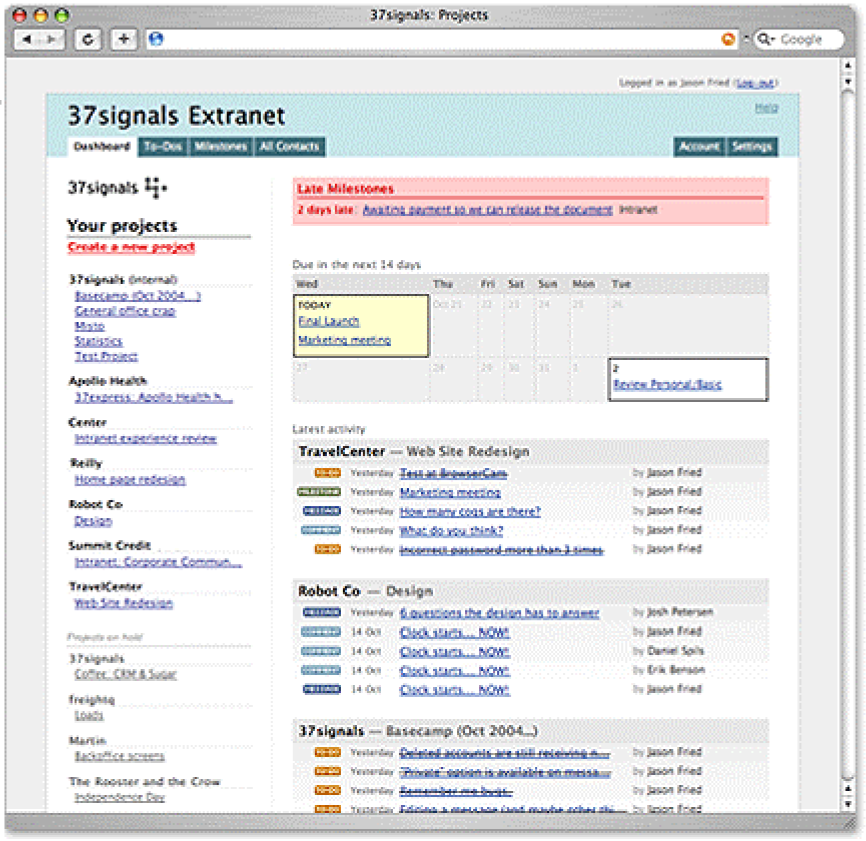
Contacts and credits

The Ruby FAQ is hosted on rubygarden.org, and is based on the work of several people.

If you have questions or problems to report about this site, please contact David Black or Chad Fowler.

Development Metrics

- **Basecamp: A commercial Rails web-app with over 10,000 users.**
  - "Web-based project management the way it should be"
  - http://www.basecamphq.com/
  - Launched after
  - 4,000 Lines of Code.
  - 2 man-months of programming by a single developer (the Rails author).


37signals: Projects

Logged in as Jason Fried (Log out)

37signals Extranet

Help

Dashboard | To-Dos | Milestones | All Contacts | Account | Settings

37signals

Your projects

Create a new project

37signals (internal)
- Basecamp (Oct 2004...)
- General office crap
- Misto
- Statistics
- Test Project

Apollo Health
- 37express: Apollo Health h...

Center
- Intranet experience review

Reilly
- Home page redesign

Robot Co
- Design

Summit Credit
- Intranet: Corporate Commun...

TravelCenter
- Web Site Redesign

Projects on hold

37signals
- Coffee, CRM & Sugar

freighting
- Loads

Martin
- Backoffice screens

The Rooster and the Crow
- Independence Day

Late Milestones

2 days late: Awaiting payment so we can release the document Intranet

Due in the next 14 days

Wed	Thu	Fri	Sat	Sun	Mon	Tue
TODAY Final Launch Marketing meeting	Oct 21	22	23	24	25	26
27	28	29	30	31	1	2 Review Personal/Basic

Latest activity

TravelCenter — Web Site Redesign

[illegible]	Yesterday	~~Test at BrowserCam~~	by Jason Fried
[illegible]	Yesterday	Marketing meeting	by Jason Fried
[illegible]	Yesterday	How many cogs are there?	by Jason Fried
[illegible]	Yesterday	What do you think?	by Jason Fried
[illegible]	Yesterday	~~Incorrect password more than 3 times~~	by Jason Fried

Robot Co — Design

[illegible]	Yesterday	6 questions the design has to answer	by Josh Petersen
[illegible]	14 Oct	Clock starts.... NOW!	by Jason Fried
[illegible]	14 Oct	Clock starts.... NOW!	by Daniel Spils
[illegible]	14 Oct	Clock starts.... NOW!	by Erik Benson
[illegible]	14 Oct	Clock starts.... NOW!	by Jason Fried

37signals — Basecamp (Oct 2004...)

[illegible]	Yesterday	~~Deleted accounts are still receiving e...~~	by Jason Fried
[illegible]	Yesterday	~~"Private" option is available on messa...~~	by Jason Fried
[illegible]	Yesterday	~~Remember me bugs~~	by Jason Fried
[illegible]	Yesterday	~~Editing a message (and maybe other thi...~~	by Jason Fried

Rails Testimonials

“I'm absolutely floored by how fast I'm developing with Rails. Stuff that would have taken me over a week in Java + Web Work2 + Velocity + Hibernate has taken me a little over a day with Rails. I'm not even going to try to compare it to my current client's project which requires Struts.”

- Anoop Ranganath

“I was but a lowly PHP programmer, slogging through thousands of lines of less-than-maintainable code. I was coercing Smarty, an innocent templating engine, to my evil whims. The innards of my latest PHP project looked like a multi-car accident.

Cue RubyOnRails. The sheer elegance of the thing has freed my mind from the mires of PHP. No longer is my vision limited by the tedium of PHP. Writing web-applications has become a pure joy.”

- Jorgen Hahn

Rails Testimonials

“I'm planning to demonstrate Rails to Amazon next week.”

- Dave Thomas

Author: The Pragmatic Programmer

Active Record

- Object-relation mapping put on rails
  - Active Record connects business objects and database tables to create a persistable domain model where logic and data is presented in one wrapping.
  - Active Record's main contribution to the ORM pattern is to relieve two stunting problems: lack of associations and inheritance.
  - A simple domain language-like set of macros describe associations.
  - Single Table inheritance narrows the gap of functionality between the data mapper and the active record approach.

Active Record Features

Active Record Features

Active Record

- **Automated mapping between classes and tables, attributes and columns.**

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

...is automatically mapped to the table named "products", such as:

```
CREATE TABLE products (
  id int(11) NOT NULL auto_increment,
  name varchar(255),
  PRIMARY KEY (id)
);
```

Active Record

- **Associations between objects controlled by simple meta-programming macros.**

```
class Firm < ActiveRecord::Base
  has_many :clients
  has_one :account
  belongs_to :conglomorate
end
```

...adds methods that allows code like this:

```
firm = Firm.find(id);
firm.clients.each do |client|
  ...some-interesting-processing...
end
```

Active Record

- **Validation rules can differ for new or existing objects.**

```
class Post < ActiveRecord::Base

  def validate # validates on both creates and updates
    errors.add_on_empty "title"
  end

  def validate_on_update
    errors.add_on_empty "password"
  end

end
```

Active Record

- Callbacks as methods or queues on the entire lifecycle (instantiation, saving, destroying, validating, etc).

```
class Subscription < ActiveRecord::Base
  # Automatically assign the signup date
  def before_create
    self.signed_up_on = Date.today
  end
end

class Firm < ActiveRecord::Base
  # Destroys the associated clients and people when
  #  the firm is destroyed
  def before_destroy
    Client.destroy_all "client_of = #{id}"
    Person.destroy_all "firm_id = #{id}"
  end
end
```

Active Record

- **Inheritance hierarchies**

```
class Company < ActiveRecord::Base; end

class Firm < Company; end
class Client < Company; end
class PriorityClient < Client; end
```

Active Record

- Transaction support on both a database and object level.

```
# Just database transaction
Account.transaction do
  david.withdrawal(100)
  mary.deposit(100)
end

# Object transaction
Account.transaction(david, mary) do
  david.withdrawal(100)
  mary.deposit(100)
end
```

Active Record

- Reflections on columns, associations, and aggregations

```
reflection = Firm.reflect_on_association(:clients)

reflection.klass  # => Client (class)

Firm.columns      # Returns an array of column
                  # descriptors for the firms table
```

Active Record

- Database abstraction through simple adapters
  - About 100 lines of code.

```
ActiveRecord::Base.establish_connection(
  :adapter => "sqlite",
  :dbfile  => "dbfile“
)

ActiveRecord::Base.establish_connection(
  :adapter  => "mysql",
  :host     => "localhost",
  :username => "me",
  :password => "secret",
  :database => "activerecord"
)
```

Active Record

- Data definitions are specified only in the database.
  - Active Record queries the database for the column names.
  - When an database object is instantiated, attributes are created for each column name.

```
# CREATE TABLE companies (
#   id int(11) unsigned NOT NULL auto_increment,
#   client_of int(11),
#   name varchar(255),
#   type varchar(100),
#   PRIMARY KEY (id)
# )
Active Record automatically links the "Company"
  object to the "companies" table
```

Active Record Sample Code

```
# SQL: INSERT INTO companies (name) VALUES ("Next Angle")
firm = Firm.new("name" => "Next Angle")
firm.save
```

Lots of different finders:

```
# SQL: SELECT * FROM companies WHERE id = 1
next_angle = Company.find(1)

# SQL: SELECT * FROM companies WHERE id = 1 AND
# name = 'Next Angle'
next_angle = Company.find_first "name = 'Next Angle'"
```

Dynamic methods are added by the has_many macro:

```
next_angle.has_clients? # true
next_angle.clients_count # total number of clients
all_clients = next_angle.clients
```

Action Controller Features

Action Controller Features

Actions

- Actions: from request to response
  - Rails splits the response to a web request into a controller part (performing the logic) and a view part (rendering a template).
  - This two-step approach is known as an action, which will normally create, read, update, or delete (CRUD for short) some sort of model part (often backed by a database) before choosing either to render a template or redirecting to another action.
  - Rails implements these actions as public methods on Action Controllers and uses Action Views to implement the template rendering.

Action Controllers

- Actions are grouped in the controller as methods instead of separate command objects and can therefore share helper methods.
  - Responsible for handling all the actions relating to a certain part of an application.
  - This grouping usually consists of actions for lists and for CRUDs revolving around a single (or a few) model objects.
  - So ContactController might be responsible for listing contacts, creating, deleting, and updating contacts.
  - A WeblogController could be responsible for both posts and comments to a blog.

Action Controllers

- **Example:**

```
class BlogController < ActionController::Base
  def display
    @customer = find_customer
  end
  def update
    @customer = find_customer
    @customer.attributes = @params["customer"]
    @customer.save ?
      redirect_to(:action => "display") :
      render("customer/edit")
  end
  def find_customer()
    Customer.find(@params["id"])
  end
end
```

Action View Features

Action View Features

Action Views

- Rails templates are written using embedded Ruby
  - Like JSP or ASP, Ruby code is embedded in tags that are mingled in with the HTML.
  - To avoid cluttering the templates with code, a bunch of helper classes provide common behavior for forms, dates, and strings.
  - It’s easy to add your own specific helpers to keep the separation as the application evolves.

Action Views

- **Example template:**

```
<html>
  <body>
    <% for post in @posts %>
      Title: <%= post.title %>
    <% end %>
    All post titles:
    <%= @post.collect{ |p| p.title }.join ", " %>
    <% unless @person.is_client? %>
      Not for clients to see...
    <% end %>
 </body>
</html>
```

Rails vs. Struts

Rails vs. Struts

Rails vs. Struts

- Rails is more focused on the goal (a web-application) than the means.
  - Rails is a full, integrated solution and has an answer for all three letters in MVC.
  - Struts is squarely focused on providing the controller.
  - Hibernate (often used with Struts) is just an ORM.
  - Honorable mention: Spring is designed to be more of a full solution.

Rails vs. Struts

- Actions and return values
  - All actions in Struts must be mapped in an XML file.
  - All return values use indirection (SUCCESS/FAILURE) that must be mapped in an XML file.
  - In Rails, this is all handled by reflection (the framework figures out how the configuration should look).

Rails vs. Struts

- **Pretty URLs**
  - Rails cares deeply about Pretty URLs, such as `/customers/show/154`. It's baked right into the framework.
  - Struts exposes technology in the URLs and generally doesn't work hard to care for the beauty of the URL.
  - (Third-party add-ons can alleviate that to some degree).

Rails vs. Struts

- Actions
  - Struts' actions are full-fledged classes.
  - Rails actions are just methods.

- For example:
  - The average code lines of code for an action in Basecamp[1] is 5.
  - Look at any struts example and find any action. It's not even funny.

[1] Basecamp is a commercial, production web-app written with Rails: http://www.basecamphq.com/

Rails vs. Struts

- Layouts
  - Rails supports the concept of layouts natively.
  - Struts requires the aid of Tiles, which in turn **requires its own set of XML files** for configuration -- on top of what's already required in Struts!

Rails vs. Struts

- Validation
  - Validation in Struts requires the use of ActionForms.
    - *These "model mirrors" have a 1-1 mapping to your model files (most of the time), but use a different syntax. This means you are "Repeating Yourself".*
  - In Rails, validation is pushed to the model.
    - *The presentation of validation errors are kept in the Action Controller.*

Rails vs. Struts

- Filters and Interceptors
  - Rails has deep support for filters and interceptors (running shared code before and after all actions).
  - Struts has a research project called SAIF to get this type of support, but nothing in the main framework.

Rails vs. Struts

- Scaffolding
  - Rails has scaffolding to quickly bring a model class "online" (provides CRUD operations without writing any code).
  - Struts has no equivalent.

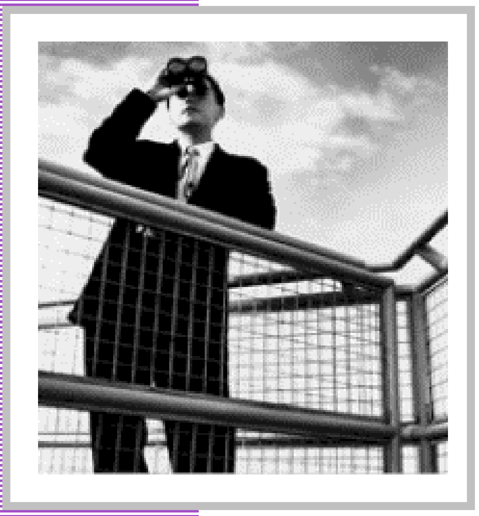
Rails vs. Struts

- Lines of Code
  - Rails is implemented in about 2,000 lines of Ruby code.
    - *This is easily understandable by us mere mortals.*
  - Struts is implemented in about ??? Lines of Java code.
    - *Is there anyone here that actually understands the internal implementation of Struts?*

Rails vs. Struts

- What I like about Struts
  - How it (somehow) positioned itself as the default controller in the majority of run-of-the-mill J2EE apps.
  - That's a freaking awesome piece of PR work!

Resources for more information



- Ruby
  - Main Ruby Site
    - http://www.ruby-lang.org/en/
  - Ruby Documentation
    - http://www.ruby-doc.org/
  - One-Click Ruby Installer for Windows
    - http://rubyinstaller.rubyforge.org/
  - RubyForge – open source project repository
    - http://rubyforge.org/
- Rails
  - Main Rails Site
    - http://www.rubyonrails.org/

Installing Rails

1. Install Ruby
   - On Windows you can use the One-Click installer. For other platforms, see the main Ruby web site.
2. Install *RubyGems*
   (a ruby package management system)
   - Download from *http://rubyforge.org/projects/rubygems/*
   - Unpack the downloaded archive and in that directory run the command:

     ```
     ruby install.rb
     ```
3. Install *Rails*
   - From the command line execute:

     ```
     gem install rails
     ```