

Introduction to **Ruby on Rails**

Prasad

Why Ruby?

- A pure-object oriented programming language – Everything is an object!

```
5.times do print "hello".capitalize end
```

- Easy to learn and use
- Powerful code blocks

```
"hello".gsub!(/./) { |match| match.capitalize }
```

- Extending at run-time
 - Add functionality to Arrays and Integers!
- Extensive standard library

Why Rails? (1/2)

- Primarily a framework for database driven web applications
- Many times faster development ;)
- Scripting and not coding/compiling
- MVC
 - Model defines the data model
 - View manages the visual components
 - Controller forms the basis for interactions and controls the model and views.

Why Rails? (2/2)

- Guiding principles of Rails
 - Less software
 - Fewer lines of code
 - Less code means lesser the number of bugs
 - Faster development
 - Easy to understand, maintain and enhance
 - Convention over Configuration
 - Almost zero configuration, everything goes by conventions that need to be followed
 - Eg: Every table has an “id” field. For every action, a view with the same name is rendered

A couple of examples

1. “Hello World” Application

2. Weblog application

- Create posts
- Edit posts
- Delete posts

3. More features to the Weblog application

- Adding Comments

Hello World

- Create the application

```
rails hello; cd hello
```

```
ruby script/generate controller hello
```

- Edit app/controller/hello_controller.rb

```
def index
```

```
  render :text => "Hello World"
```

```
end
```

- Test the web application

```
ruby script/server
```

```
http://localhost:3000/hello
```

- *Demo – Hello World application*

Weblog – Starting steps

- Create the Rails application

```
rails weblog; cd weblog
```

- Generate the “Blog” controller

```
ruby script/generate controller Blog
```

- Check if properly setup

```
ruby scripts/server
```

<http://localhost:3000/> (Rails welcome message)

<http://localhost:3000/blog> (Unknown action – no index action!)

Weblog – Database (1/2)

- Convention over configuration ;)
- Database is the only required configuration for a rails application
- config/database.yml

```
development:
```

```
  adapter: mysql
```

```
  database: weblog_dev
```

```
  username: root
```

```
  password:
```

```
  host: localhost
```

Weblog – Database (2/2)

- The first table - posts
 - “id” - required by conventions of Rails
 - “title” - title of the post
 - “body” - body of the post
 - “created_at” - create time for the post
 - “updated_at” - update time for the post
- Create the database and table in MySQL

Weblog – The First Post

- Generate the model
 - `ruby script/generate model post`
 - Rails creates a model and automatically populates the class “Post” with methods to access data from the database table “posts”
- Start posting!
 - `app/controllers/blog_controller.rb`

```
class BlogController < ApplicationController
  scaffold :post
end
```
 - `http://localhost:3000/blog` and start blogging!

Making it Usable

- Generate the scaffold code

```
ruby script/generate scaffold post blog
```

- The view templates
 - Naming conventions
 - Inline ruby
 - Sharing variables with controller
 - Partials
- Modify view template for a more visually pleasing and stylish look

Extending Weblog - Comments

- The comments table
 - “id” - primary key
 - “title” - title of the post
 - “body” - body of the post
 - “post_id” - foreign key for the attached post
 - “created_at” - create time for the post
 - “updated_at” - update time for the post
- Create the table in MySQL

Weblog Comments

- Generate the comment model

```
ruby script/generate model comment
```
- Establish the comments-posts relation
 - In app/models/post.rb

```
has_many :comments
```
 - In app/models/comment.rb

```
belongs_to :post
```
- Modify the templates to show comments
- *Demo: Comments in View and Controller*

Summary - Model

- Database tables
 - Conventions: “id”, foreign key “_id”
- Model (app/models folder)
 - Create models

```
ruby script/generate model <model_name>
```
 - Establish associations

```
belongs_to, has_many, has_one etc;
```
 - Write validations

```
validates_presence_of, validates_length_of etc;
```

Summary - Controller

- Create a controller

```
ruby script/generate controller <controller_name>
```

- Controller (app/controllers folder)

- Automating the action creation

```
scaffold :<model_name>
```

- Actions for additional functionality

- Extract scaffold code

```
ruby script/generate scaffold <model> <controller>
```

- Edit the generated code

Summary - View

- `app/views/<controller_name>`
- Convention: Views with same name as action are rendered when an action is called
- Uses plain Ruby, class variables from controller can be used in View
- Generate the scaffold code to see the default views

More Features

- Supports AJAX via inbuilt function calls and Javascript
- Models can be extended to handle file uploads to file-system or to database
- Authentication modules
- more modules being developed...

Rails in Action

- 37 Signals -
 - <http://www.37signals.com/>
 - Basecamp
 - <http://www.basecamphq.com/>
 - Created in less than 2 man-months!
 - Writeboard
 - <http://www.writeboard.com>
- More web applications at <http://wiki.rubyonrails.com/rails/pages/RealWorldUsage>

Where to Start?

- <http://www.ruby-lang.org/>
- <http://www.rubyonrails.org/>
- Documentations
 - Ruby
 - <http://www.rubycentral.com/book/>
 - Rails
 - Google for “Rolling with Ruby on Rails”
 - Google for “Four days on Rails”
 - Google for “Ajax on Rails”
 - <http://wiki.rubyonrails.org/rails>

Questions?

Thank You!

s.prasad@gmail.com