Ruby on Rails

Object Oriented Analysis & Design

CSCI-5448

University of Colorado, Boulder

-Dheeraj Potlapally

INTRODUCTION

What is Ruby on Rails

- Ruby on Rails is a web application framework written in Ruby.
- David Heinemeier Hansson developed it in 2003.
- Since then Improvised by Rails Core Team.
- Released in 2004 termed as RoR.
- Works on Windows, Macintosh, Linux.
- Compatible to all common databases.

What else are available in Market

Many other frameworks exist in market and more importantly many other languages for web development are available.

- Python-TurboGears, WebPy, Django
- Java-JSP/Servlets, Struts
- PHP-Symfony

Why again a new tool?(1)

Fast Development and Prototyping

- Old Techniques: When developing website for a client there may be issues like creating specific to client's domain which would slow down the development process.
- RoR: Uses a concept called Convention over Configuration which leaves with little configuration while writing the code.

Why again a new tool?(2)

Structured Code and Neat Markup

- Old Techniques: Problem here is separating the view markup from business logic. Asp and PHP applications have business logic code scattered throughout HTML. This also makes code hard to maintain.
- RoR: Rails uses MVC (model-view-controller) compound design pattern to solve this problem.
- By using MVC we have a clean, structured code, which is easily maintainable and very flexible.

Why again a new tool?(3)

Interchangeable Databases

- Old Techniques: Once the website is build using MySQL and suppose there is a change in decision and we need to use Sybase or Oracle. Then the problem of rewriting the code comes in since the embedded SQL Query strings are scattered in our code.
- RoR: Ruby on Rails is Database independent, meaning we can make only few configuration changes and we can start using Microsoft SQL, Oracle etc.

Why again a new tool?(4)

Database Schema Control

- Old Techniques: When working in a big group unexpected changes to database may occur by other people because of old revision of scema.
- RoR: This problem is solved by a method called 'Migrations'. When changes are made, fellow developers can run a simple command and they will be on the latest version of the database. Migrations are basically version control for the database schema.

FEATURES OF RUBY

What is Ruby

- It is an Object-Oriented and Dynamic Programming Language.
- Written in 1995 by Yukihiro Matsumoto.
- Influenced by Python, Perl and Lisp.
- Easy to understand and work with.
- Simple syntax.
- Powerful dynamic features and metaprogramming capabilities.

Ruby is Truly Object-Oriented

All classes derived from **Object** including **Class** (like Java) but there are no primitives (not like Java at all)

- Ruby uses single-inheritance.
- Mixins give the power of multiple inheritance.
- Modules allow addition of behaviors to a class.
- Reflection is built in along with lots of other highly dynamic metadata features.
- Memory management via garbage collector as in Java.

Overall we can think it as a programmer friendly language with strong capabilities.

Dynamic Programming

- Duck Typing
  Based on signatures, not class inheritance
- Dynamic Dispatch
  A key concept of OOP: methods are actually messages that are *sent* to an object instance
- Dynamic Behavior
  - Reflection
  - Scope Reopening (Kind of like AOP)
  - Eval
  - Breakpoint debugger

RAILS IN A NUTSHELL

Features of Rails in Detail(1)

- Full Stack MVC Framework:

  Rails is an MVC (model, view, controller) framework where Rails provides all the layers and they work together seamlessly. Other frameworks often implement only part of the solution, requiring the developer to integrate multiple frameworks into the application and then coerce them into working together. (For example, a Java developer might use Hibernate, Struts, and Tiles to get full MVC support.)

Features of Rails in Detail(2)

- Convention over Configuration

  Convention over configuration means an end to verbose XML configuration files--in Rails, there aren't any! Instead of XML sit-ups, a Rails application uses a few simple programming conventions that allow it to figure everything out through reflection and discovery. For example, Rails uses intelligent reflection to automatically map database tables to Ruby objects.

Features of Rails in Detail(3)

- Scaffolding

  Rails can automatically create a full set of CRUD (Create, Retrieve, Update, and Delete) operations and views on any database table. This scaffolding can get us up and running quickly with manipulating our database tables. Over time, we can incrementally replace the generated CRUD operations and views with our own--presumably much prettier and more functional.

Problems with Scaffolding(1)

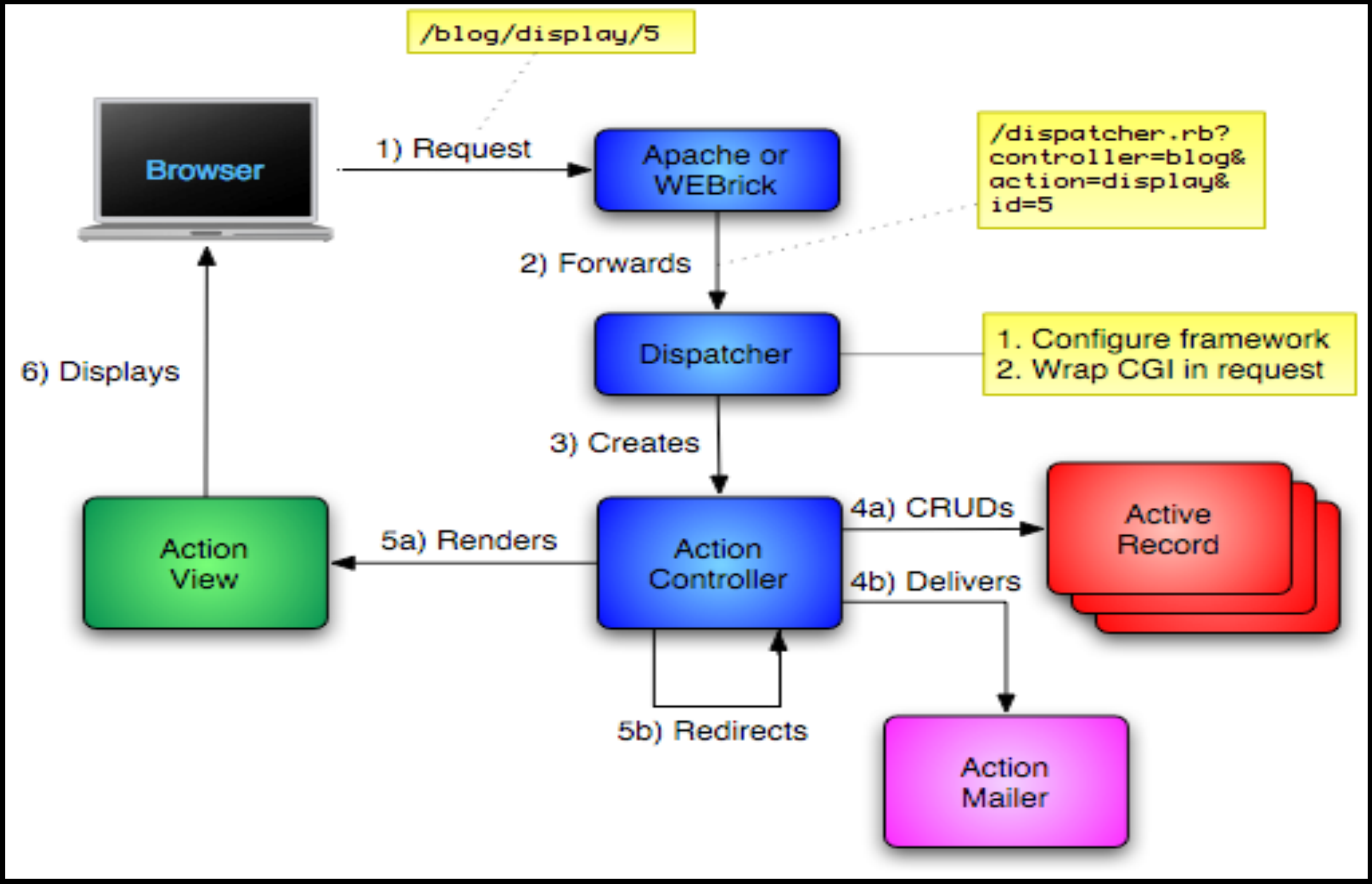
- No data validations. Our User model accepts data such as blank names and invalid email addresses without complaint.
- No authentication. We have no notion signing in or out, and no way to prevent any user from performing any operation.
- No tests. This isn't technically true—the scaffolding includes rudimentary tests.

Problems with Scaffolding(2)

The scaffolding includes rudimentary tests—but the generated tests are ugly and inflexible, and they don’t test for data validation, authentication, or any other custom requirements.

- No layout. There is no consistent site styling or navigation.

The RoR Architecture

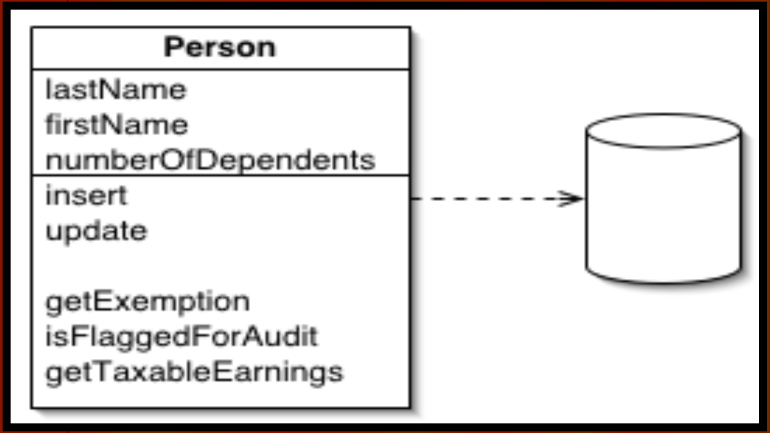


Components of Ruby on Rails

Model Classes

Based on Martin Fowler's ActiveRecord pattern

From Patterns of Enterprise Architecture



An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.

ActiveRecord

- No XML files needed.
- Lots of reflection and run-time extension provided.
- Magic is not inherently a bad word.
- We can drop down to SQL for odd cases and performance if needed.
- Doesn ‘t attempt to duplicate or replace data definitions.

ActiveRecord API

The Following are the features of API

- Automatic mapping between columns and class attributes.
- Declarative configuration via macros.
- Dynamic finders.
- Associations, Aggregations.
- Locking.
- Validation rules

ActiveRecord Aggregations

Aggregation expresses a ‘*composed of* relationship’

- Define *value* objects by using composed_of method
  - Tells Rails how value objects are created from the attributes of the entity object when the entity is initialized and…
  - How it can be turned back into attributes when the entity is saved to the database
  - Adds a reader and writer method for manipulating a value object
- Attempting to change value objects result in a TypeError
  - Value objects should be immutable and that requirement is enforced by Active Record by freezing any object assigned as a value object.

ActionController API

Controllers defined as classes that execute and then either render a template or redirects

An action is a public method on the controller

Getting data in and out of controllers

- Request parameters available in the @params hash
- Web session exposed as @session hash
- Cookies exposed as @cookies hash
- Redirect scope provided by @flash hash (unique to Rails)

V-View

Rails gives you many rendering options…

- Default template rendering.
- Explicitly render to particular action.
- Redirect to another action.
- Render a string response.

Viewing Templates

ERB – Embedded Ruby

- Similar to JSPs
- Easy to learn and teach to designers
- Execute in scope of controller
- Denoted with .rhtml extension

XmlMarkup – Programmatic View Construction

- Great for writing xhtml and xml content
- Denoted with .rxml extension
- Embeddable in ERB templates

Exception Handling

Exceptions do arise knowingly or unknowingly where the behavior is different from expected.

Exceptions can either be rescued…

- for public view (with a nice user-friendly explanation)
- for developers view (with tons of debugging information)

By default, requests from localhost get developers view

ActiveSupport API

Rails utility methods

- Number handling
- Date conversion
- Time calculations
- String conversions and inflection

ActionMailer API

Rails' built-in email service

- We can write email controllers in same way as web controllers making it is easy to use.
- Integrated with 'templating system' for end user friendliness.

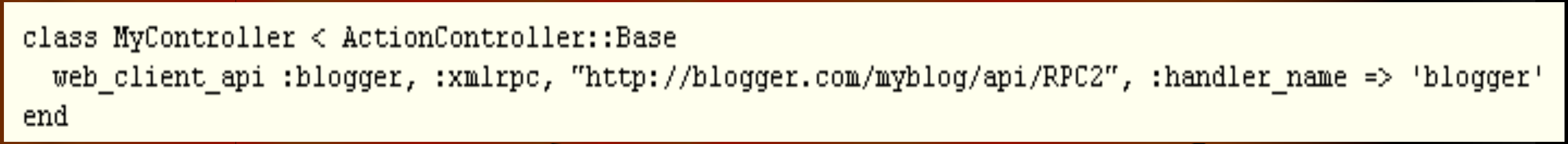
ActionWebService API

Web Services is a prime application of RoR.

- Rails has built-in support for SOAP and XML-RPC
- **Struct** base class can be used to represent structured types that don’t have ActiveRecord implementations or to avoid exposing an entire model to callers

Examples

- Define a web services client in one line
- Add a web service to a controller (next slide)



```
class MyController < ActionController::Base
  web_client_api :blogger, :xmlrpc, "http://blogger.com/myblog/api/RPC2", :handler_name => 'blogger'
end
```

Defining a WebService

```
class PersonService < ActionWebService::Base
  web_service_api PersonAPI

  def find_person(criteria)
    Person.find_all [...]
  end

  def delete_person(id)
    Person.find_by_id(id).destroy
  end
end

class PersonAPI < ActionWebService::API::Base
  api_method :find_person,   :expects => [SearchCriteria], :returns => [[Person]]
  api_method :delete_person, :expects => [:int]
end

class SearchCriteria < ActionStruct::Base
  member :firstname, :string
  member :lastname,  :string
  member :email,     :string
end
```

Security Aspect of RoR

Sessions and Security(1)

- Rails provides storage mechanisms for session hashes importantly ActiveRecordStore and CookieStore.
- CookieStore saves the session hash directly in a cookie on the client-side. The server retrieves the session hash from the cookie and eliminates the need for a session id.
- CookieStore implimentation makes it a fast application.
- Session Fixation could be overcome by creating a new session with a single line command using Rails.

reset_session

Sessions and Security(2)

- Replay attacks could be countered using maintaining certain type of data as a database rather than in sessions.
- Cross Site Request Forgery (CSRF) could be countered by using GET, POST or a Security Token Implementation.
- Possible areas to be considered further: Redirection, File Uploads, File Downloads, Executable code usage in Files etc.

Sessions and Security(3)

- Mass Assignment- Countered by providing attributes in ActiveRecord for Admin.
- User Management includes updates and plug-ins. This could be maintained by RoR with the help of restful_authentication feature.
- Injections: SQL Injection and Cross Site Scripting (XSS).
- Measures from RoR include built in filter for SQL special characters, Helper methods to fend XSS attacks, etc.
- There are many other aspects but I have restricted myself only until this point.

Resources

The Rails Community

The developer community around Rails is very helpful and excited

- Rails Wiki - http://wiki.rubyonrails.com/
- Rails Mailing List – very active
- IRC – Get instant answers to most questions. David and other Rails commiters are all regularly present

References

- http://sixrevisions.com/web-development/four-ways-ruby-on-rails-can-help-you/
- http://www.smashingmagazine.com/2009/03/19/getting-started-with-ruby-on-rails/
- http://onlamp.com/pub/a/onlamp/2005/10/13/what_is_rails.html?page=7
- http://rubyonrails.org/
- http://files.fredbrunel.com/docs/Ruby%20on%20Rails%20-%20Free%20Presentation.pdf
- http://www.scribd.com/doc/12353866/Ruby-on-Rails-Power-Point
- http://www.tutorialspoint.com/ruby-on-rails/rails-examples.htm
- *www.**ruby**doc.org/.../Introduction%20to%20**Ruby**%20and%20RoR%20given%20to%20the%20...*

Thank You