Web Application Architectures

Module 4: The Ruby Programming Language
Lecture 1: Background

THE UNIVERSITY *of* NEW MEXICO

Ruby Programming Language

- Rails was built using the Ruby programming language.
- Ruby code shows up in models:

```ruby
class Post < ActiveRecord::Base
end
```

views:

```erb
<%= @post.title %>
```

and controllers:

```ruby
def destroy
  @post.destroy
  respond_to do |format|
    format.html { redirect_to posts_url }
    format.json { head :no_content }
  end
end
```

Ruby – History

- Yukihiro Matsumoto ("Matz") created Ruby in the mid-1990s.

  "I wanted a scripting language that was more powerful than Perl, and more OO than Python. That's why I decided to design my own language."

- Matz developed Ruby with a focus on the programmer, rather than the machine. The design goal was to maximize programmer efficiency (i.e., productivity), not the runtime efficiency of the their programs.

  "I hope to see Ruby help every programmer in the world to be productive, and to enjoy programming, and to be happy. That is the primary purpose of Ruby language."

Ruby – Design

- Matz's guiding philosophy for Ruby:

  "Ruby is designed to make programmers happy."

- Ruby is designed according to the Principle of Least Astonishment – the language should behave in a way that minimizes the confusion of experienced programmers (assuming you're experienced in Ruby, not operating with some other programming model in mind).

- Ruby is an object-oriented interpreted scripting language – many find it intuitive, flexible and extensible.
  For more information, documentation and tutorials, visit:
  http://www.ruby-lang.org

Ruby – Installation

- Recall that to find the version of Ruby you're running, use:

```
$ ruby --version
```

- Ruby gems is a package management system. To see the gems you have installed, use:

```
$ gem list
```

- Rails is a Ruby gem for building database-intensive web application frameworks. To install it, use:

```
$ gem install rails
```

The Ruby Interpreter

- Ruby is an interpreted language. You invoke the interpreter using the `ruby` command:
  Ex.

  ```
  $ ruby -e 'puts "Hello World!"'
  Hello World!
  $
  ```

  The `-e` prompt tells the interpreter to execute the line of Ruby code contained in the single quotes.

- Typically you will place your Ruby code in a file, with a .rb extension. E.g., put the previous code in the file hello.rb, and tell the interpreter to execute it using:

  ```
  $ ruby hello.rb
  Hello World!
  $
  ```

The Ruby Interpreter

- Interactive Ruby Shell (IRB) is an interpreter shell that allows you to execute Ruby code from a command prompt – a REPL. It's very useful for debugging purposes.
- To open up a Ruby shell, type:

```
$ irb
2.0.0p195 :001 >
```

- At the prompt provided by interactive ruby, you can type ruby expressions, and they will be evaluated:

```
2.0.0p195 :001 > 2+2
 => 4
2.0.0p195 :002 >
```

The Ruby Interpreter

- You can invoke IRB from the root of a Rails application directory as follows:

```
$ rails console
Loading development environment (Rails 4.0.0.rc1)
2.0.0-p195 :001 >
```

- The Rails environment (including everything defined in the current Rails application) is loaded when you do this.
- You can directly manipulate your rails application from the console command line – add/delete database items, inspect and manipulate object, etc.
- This is very useful, and common, way to debug Rails applications.

Language Features

Ruby is a multi-paradigm programming language:

- Scripting – It can be used to write scripts that automate the execution of tasks within some environment.
- Imperative (procedure-oriented) programming – It has the traditional control structures found in imperative programs. You can create functions with variables (that store state); however, defining functions/variables outside classes actually makes them methods of the root `Object` class.
- Object-oriented programming – Everything is an object, derived from the `Object` class.
- Functional programming – Computation proceeds via the evaluation of functions that depend only on their input, not the program state.