Rails Basics

Sang Shin
www.javapassion.com
“Learning is fun!”

Topics

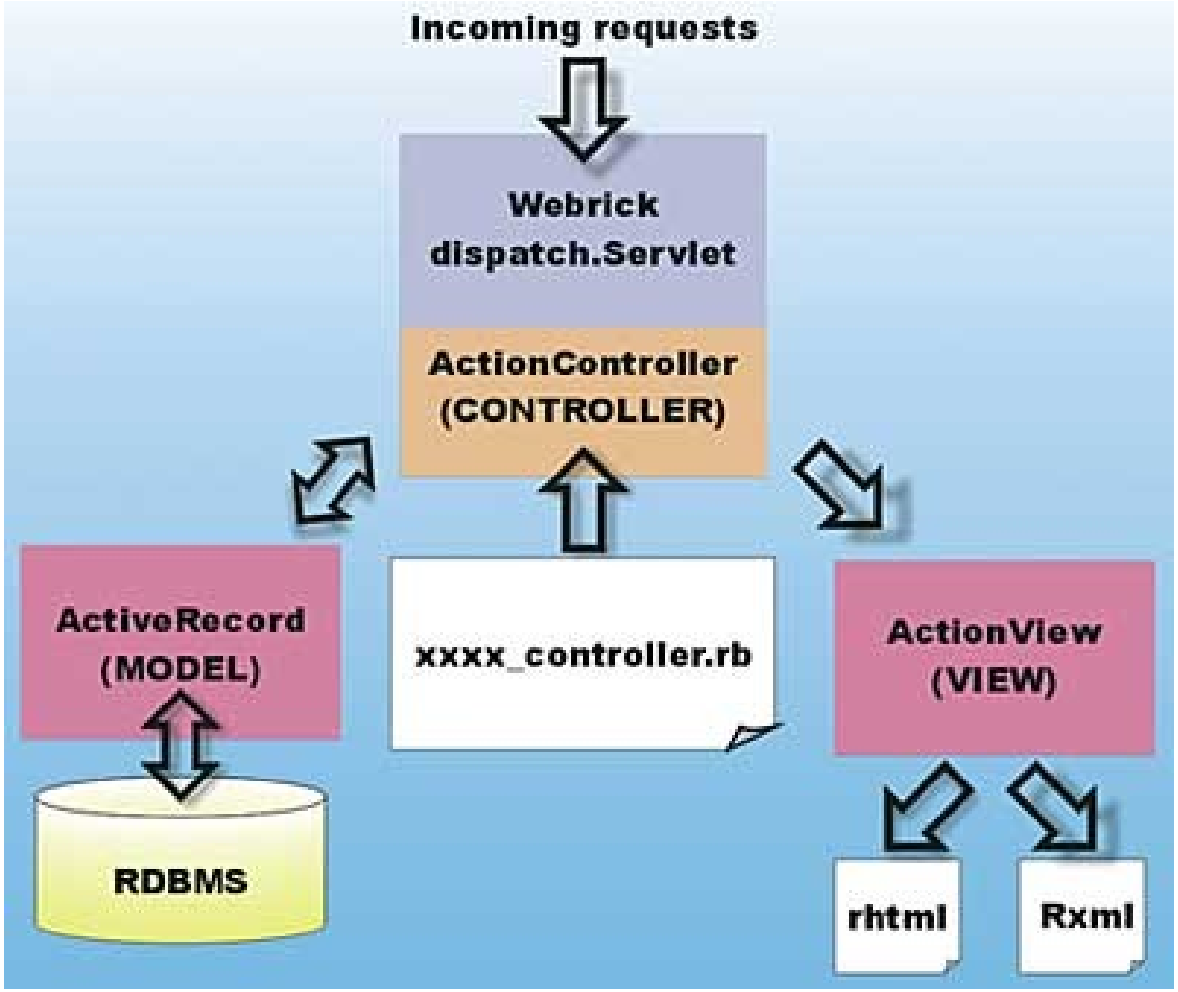
- What is and Why Ruby on Rails?
- Building HelloWorld Rails application step by step
- Key concepts of Rails application development
  - App directory structure (MVC)
  - Environment
  - Rake
  - Generator
  - Migration
  - Rails console

What is and Why Ruby on Rails (RoR)?

What Is “Ruby on Rails”?

- A full-stack MVC web development framework
- Written in Ruby
  - Rails leverages various characteristics of Ruby language - meta-programming, closure, etc.
- First released in 2004 by David Heinemeier Hansson
- Gaining popularity

“Ruby on Rails” MVC



source: http://www.ilug-cal.org

“Ruby on Rails” Principles

- Convention over configuration
  - Why punish the common cases?
  - Encourages standard practices
  - Everything simpler and smaller
- Don’t Repeat Yourself (DRY)
  - Framework written around minimizing repetition
  - Repetitive code harmful to adaptability
- Agile development environment
  - No recompile, deploy, restart cycles
  - Simple tools to generate code quickly
  - Testing built into the framework

Building “Hello World” Rails Application Step by Step

Steps to Follow

1. Create “Ruby on Rails” project
   - Rails generates directory structure
2. Create Database (using Rake)
3. Create Models (using Generator)
4. Create Database Tables (using Migration)
5. Create Controllers (using Generator)
6. Create Views
7. Set URL Routing
   - Map URL to controller and action

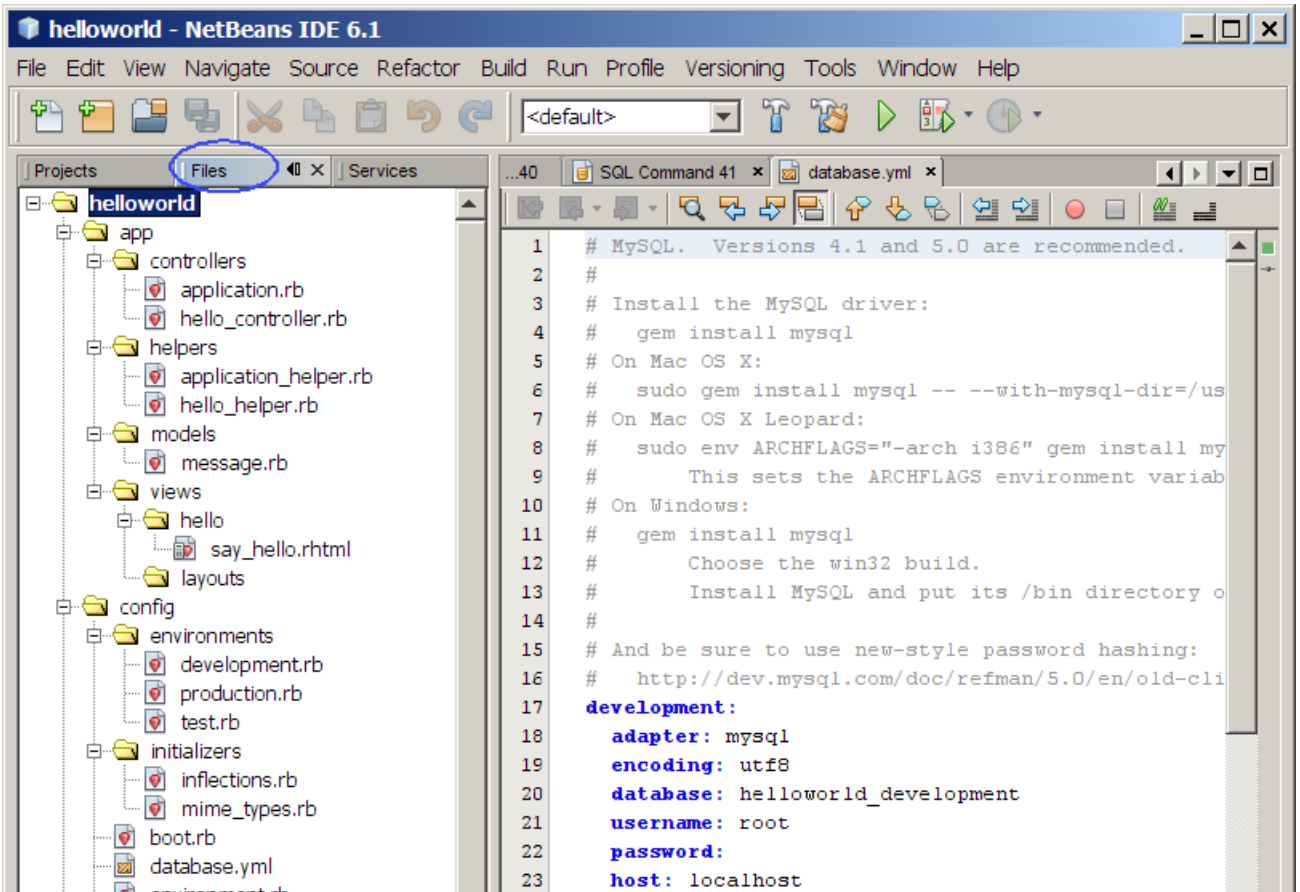
Demo:

Building “Hello World” Rails Application Step by Step.

1. Create “Ruby on Rails” Project

1. Create “Ruby on Rails” Project

- The directory structure along with boilerplate files of the application is created



Directory Structure of a Rails Application

- When you ask NetBeans to create a Rails project - internally NetBeans uses the rails' helper script -, it creates the entire directory structure for your application.
  - The boiler plate files are also created
  - The names of the directories and files are the same for all Rails projects
- Rails knows where to find things it needs within this structure, so you don't have to tell it explicitly.

Directory Structure of a Rails Application

- *app*: Holds all the code that's specific to this particular application.
  - *app/controllers*: Holds controllers that should be named like *hello_controller.rb* for automated URL mapping. All controllers should descend from *ApplicationController* which itself descends from *ActionController::Base*.
  - *app/models*: Holds models that should be named like *message.rb*. Most models will descend from ActiveRecord::Base.
  - *app/views*: Holds the template files for the view that should be named like *hello/say_hello.rhtml* for the HelloController#say_hello action.

Directory Structure of a Rails Application

- *app*
  - *app/views/layouts*: Holds the template files for layouts to be used with views. This models the common header/footer method of wrapping views. In your views, define a layout using the *<tt>layout :default</tt>* and create a file named *default.rhtml*. Inside *default.rhtml*, call *<% yield %>* to render the view using this layout.
  - *app/helpers*: Holds view helpers that should be named like *hello_helper.rb*. These are generated for you automatically when using script/generate (Generator) for controllers. Helpers can be used to wrap functionality for your views into methods.

Directory Structure of a Rails Application

- *config:* Holds configuration files for the Rails environment, the routing map, the database, and other dependencies.
  - *config/environments*
  - *config/initializers*
  - *boot.rb*
  - *database.yml*
  - *environment.rb*
  - *routes.rb*

Learning Point:
Environments

What is an Environment?

- Rails provides the concept of environments - *development, test, production*
- As a default, different database is going to be used for different environment.
  - Therefore each environment has its own database connection settings.
- It is easy to add custom environments
  - For example, staging server environment
- Rails always runs in only one environment
  - Dictated by ENV['RAILS_ENV'] (same as RAILS_ENV)

config/database.yml

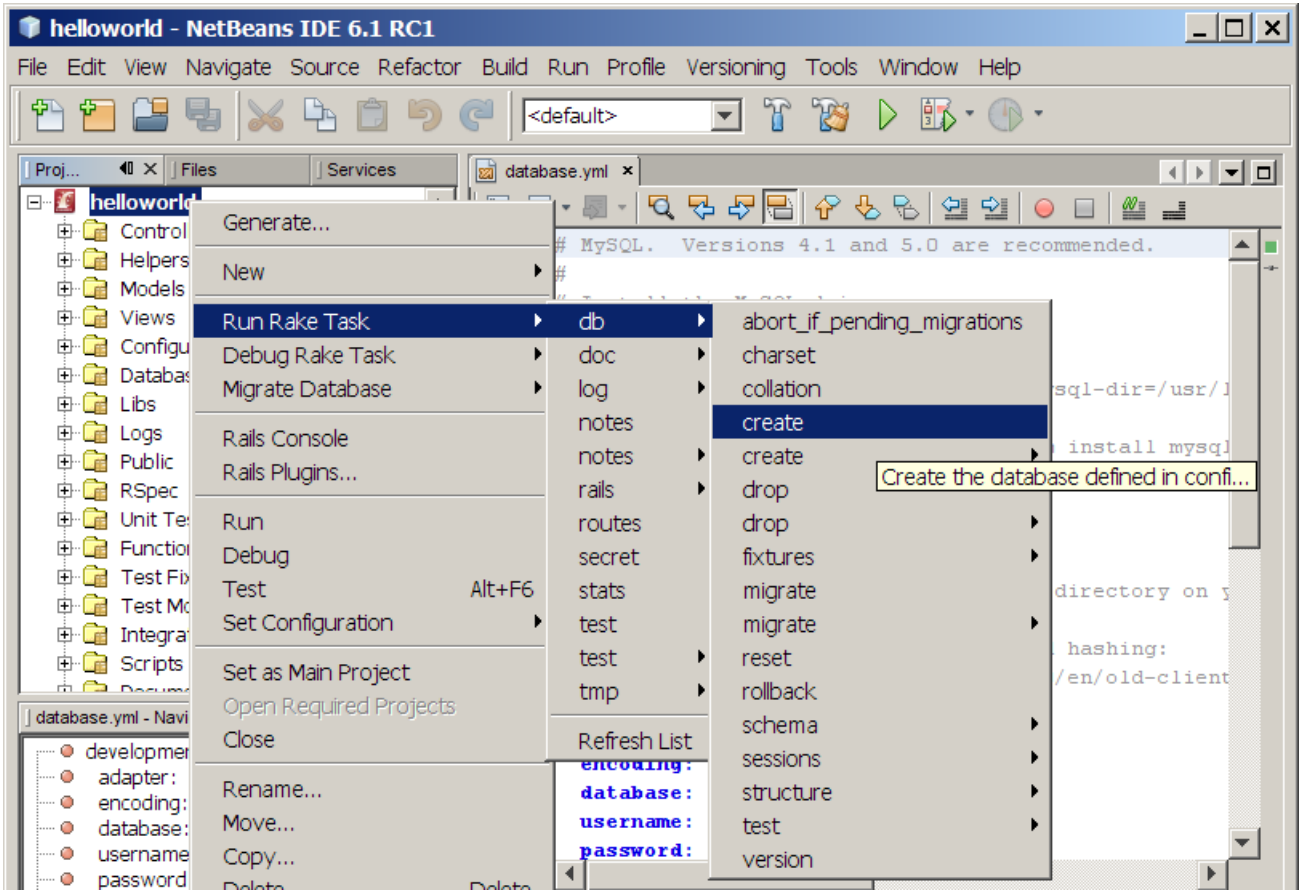
```
development:
  adapter: mysql
  encoding: utf8
  database: helloname_development
  username: root
  password:
  host: localhost

test:
  adapter: mysql
  encoding: utf8
  database: helloname_test
  username: root
  password:
  host: localhost
```

2. Create Database using “Rake”

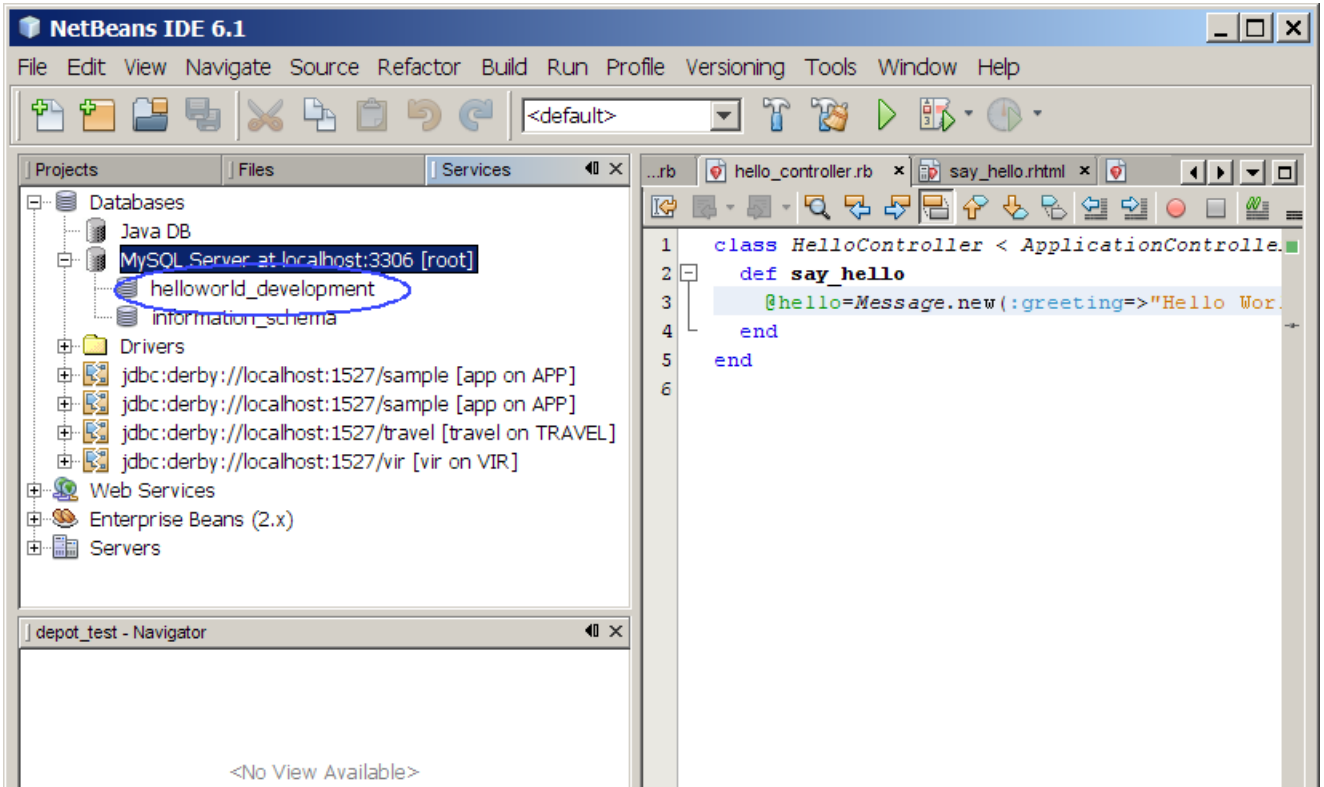
Creating Database

- Creating and dropping of databases are done using “Rake”



Creating Database

- After create rake task is performed, *<project_name>_development* database, for example, *helloworld_development* is created



Learning Point:
What is Rake?

What is “Rake”?

- Rake is a build language for Ruby.
- Rails uses Rake to automate several tasks such as
  - creating and dropping databases
  - running tests
  - updating Rails support files.
- Rake lets you define a dependency tree of tasks to be executed

How does “Rake” Work?

- Rake tasks are loaded from the file *Rakefile*
- Rails rake tasks are under *<project-name>/lib/tasks*
- You can put your custom tasks under *lib/tasks*

Useful Rake Tasks

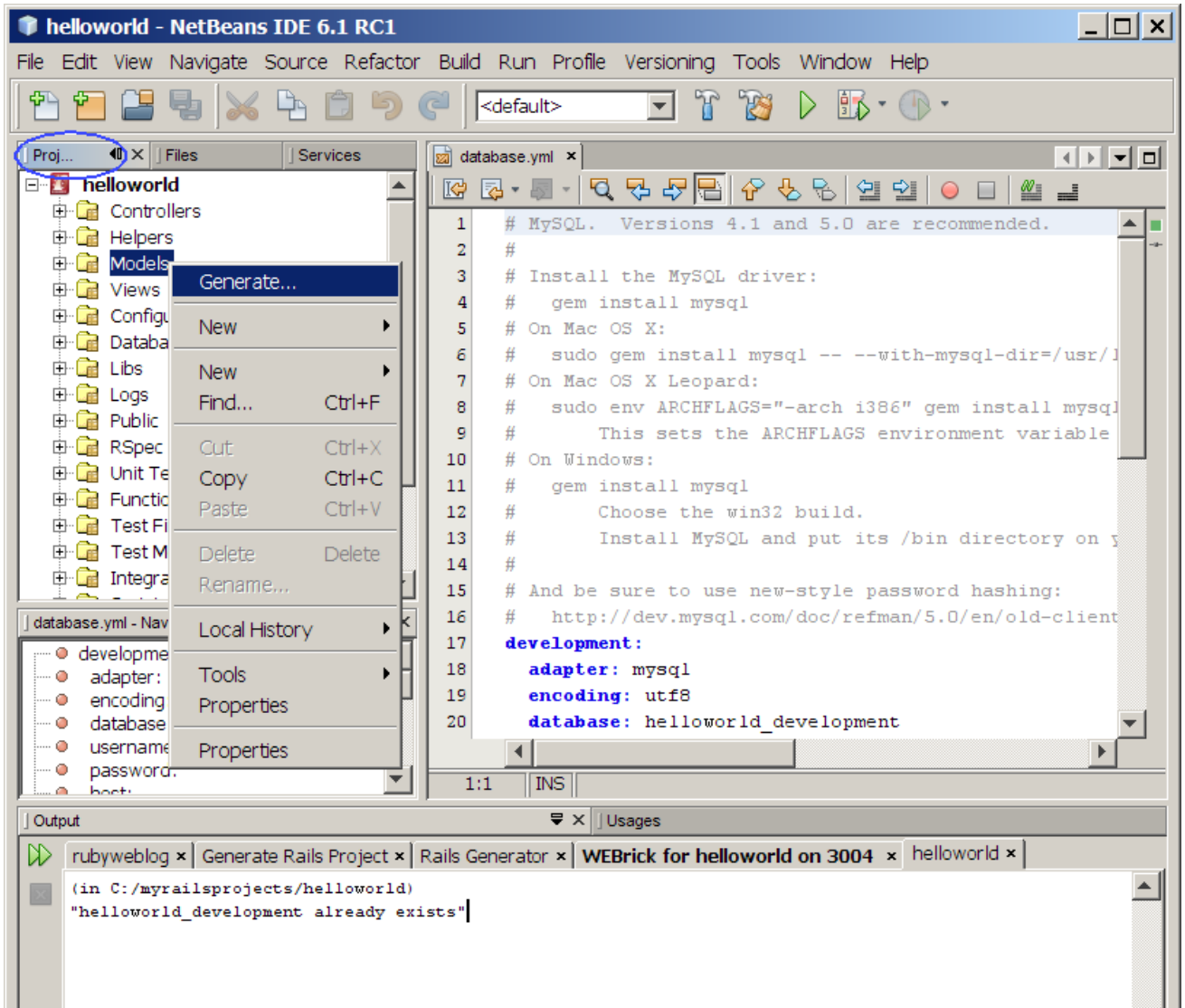
- db:migrate
- db:sessions:create
- doc:app
- doc:rails
- log:clear
- rails:freeze:gems
- rails:freeze:edge
- rails:update
- :test
- :stats

3. Create a Model through “Generator”

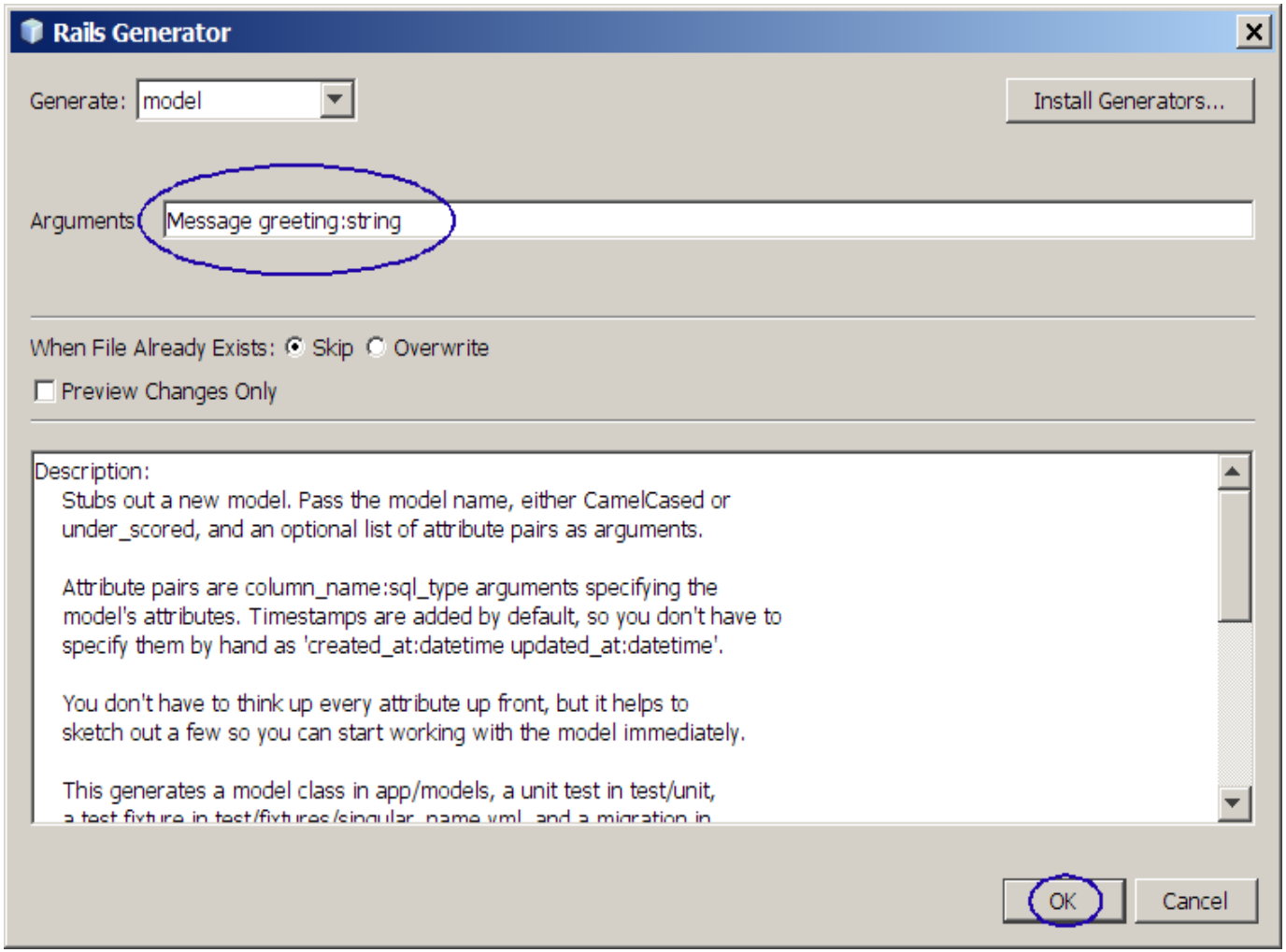
What is a Model?

- In the context of MVC pattern, a Model represents domain objects such as message, school, product, etc.
- A model has attributes and methods.
  - The attributes represents the characteristics of the domain object, for example, a message model might have length, creator as attributes.
  - The methods in a model contains some business logic.
- Most models have corresponding database tables. For example, a *message* model will have *messages* table.
- Most model classes are *ActiveRecord* type

Creating a Model using Generator



Creating a Model using Generator



Files That Are Created

- app/models/message.rb (Model file)
  - Models/message.rb in logical view
  - A file that holds the methods for the Message model.
- test/unit/message_test.rb
  - Unit Tests/message_test.rb in logical view
  - A unit test for checking the Message model.
- test/fixtures/messages.yml
  - Test Fixtures/messages.yml in logical view
  - A test fixture for populating the model.
- db/migrate/migrate/001_create_messages.rb
  - Database Migrations/migrate/001_create_messages.rb in logical view
  - A migration file for defining the initial structure of the database.

Model Class Example

- *Message* mode in *messages.rb* file

```
class Message < ActiveRecord::Base
end
```

Learning Point:
What is Generator?

What is “Generator”?

- You can often avoid writing boilerplate code by using the built-in generator scripts of Rails to create it for you.
  - This leaves you with more time to concentrate on the code that really matters--your business logic.

Learning Point:
What is Rails Console?

What is Rails Console?

- The Rails console gives you access to your Rails Environment, for example, you can interact with the domain models of your application as if the application is actually running.
  - Things you can do include performing find operations or creating a new active record object and then saving it to the database.
- A great tool for impromptu testing
- NetBeans runs a script to start Rails Console

Learning Point:
What is Rails Script?

Script

- NetBeans runs Rails Script internally
  - You can run the Script at the commandline
- Useful scripts
  - console
  - generate
  - plugin
  - server

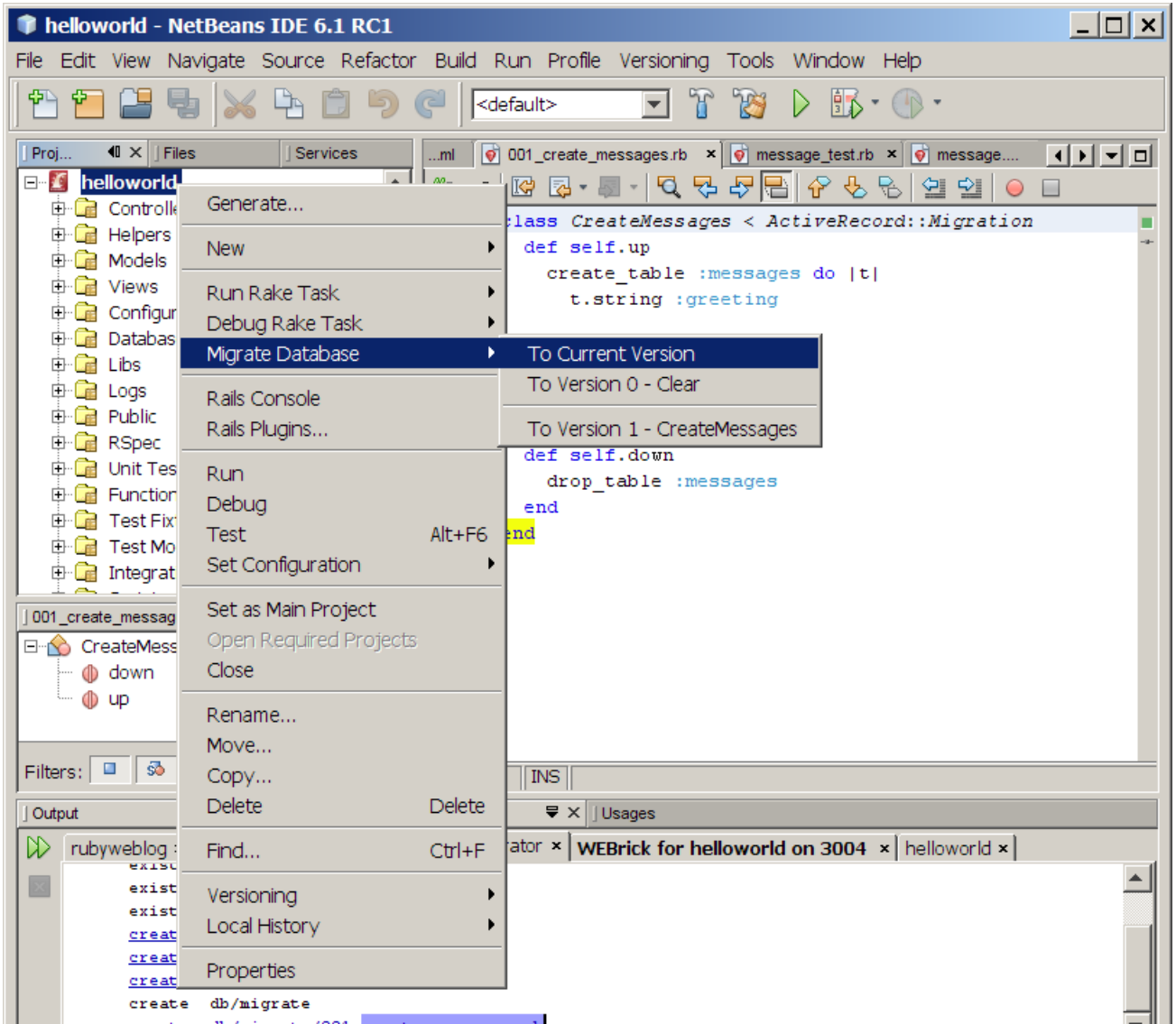
4. Create Database Tables using Migration

Create Database Table using Migration

- You are going to create a database table (in a previously created database) through migration
  - You also use migration for any change you are going to make in the schema - adding a new column, for example
- When you create a Model, the first version of the migration file is automatically created
  - *db/migrate/migrate/001_create_messages.rb*, which defines initial structure of the table

```
class CreateMessages < ActiveRecord::Migration
  def self.up
    create_table :messages do |t|
      t.string :greeting
      t.timestamps
```

Performing Migration



learning point:
What is Migration?

Issues with Schema Changes

- Database schema's keep changing (along with applications that use them)
  - Example: You need to add “email” column to the “customer” table
- Issues with schema changes
  - How do you version control schema changes?
  - How do you go back to previous version of the schema?
  - How do people work wth different versions of the schema?
  - How do you convey schema changes to other developers and the production server?

Migration To the Rescue

- Migration can manage the evolution of a schema
- With migrations, you can describe schema changes in self-contained Ruby classes - migration files
- You can check these migration files into a version control system
- Migration files are part of an application structure
- You (and others) can choose a schema version of choice, for example, several versions back from the current one

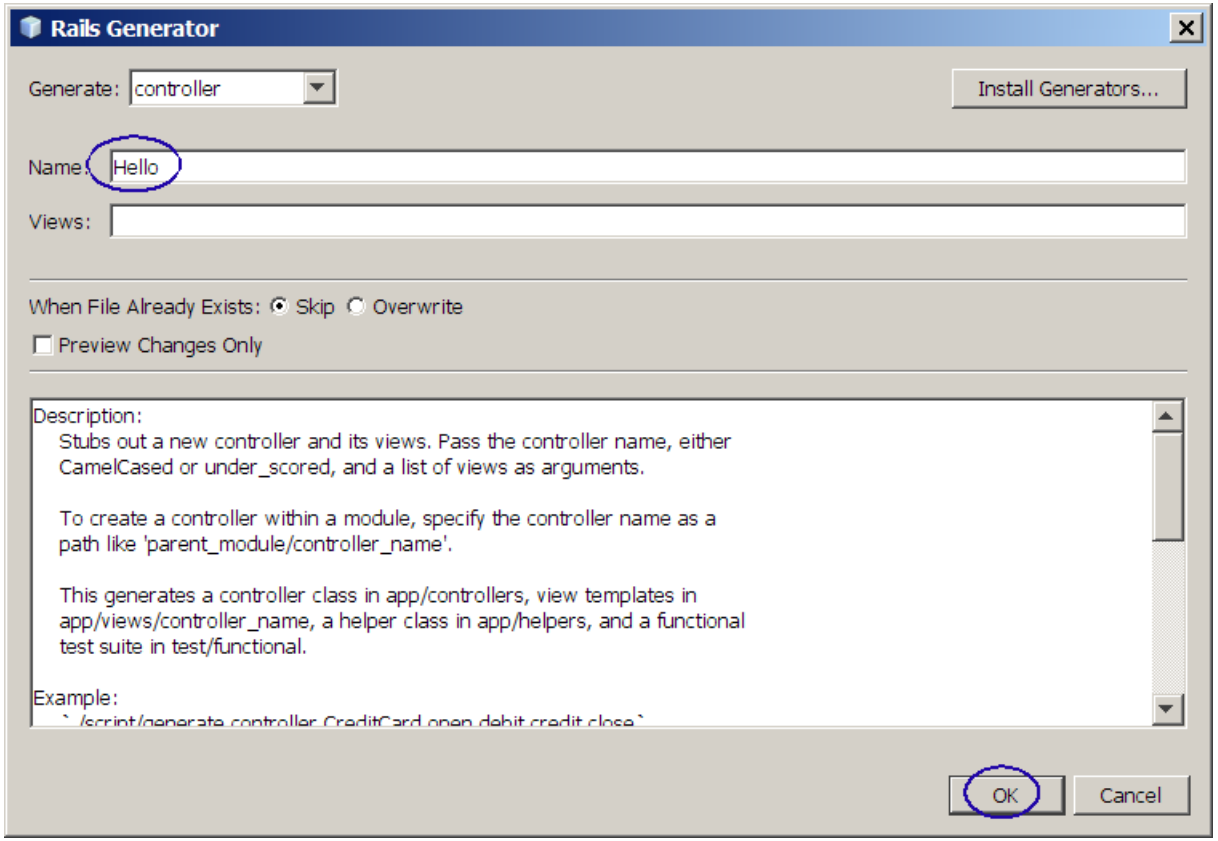
5. Create a Controller

What is a Controller?

- Action Controllers handle incoming Web requests
- A controller is made up of one or more actions
- Actions are executed to handle the incoming requests and then either render a template or redirect to another action.
- An action is defined as a public method of a controller
- Mapping between a request's URL and an action is specified in the Rails routing map (configuration/routes.rb)

Create a Controller using Generator

- You are going to create a controller using Generator



Example: HelloController

- Controller contains actions, which are defined with *def*

```
class HelloController < ApplicationController
  def say_hello
    @hello = Message.new(:greeting => "Hello
   World!")
  end
end
```

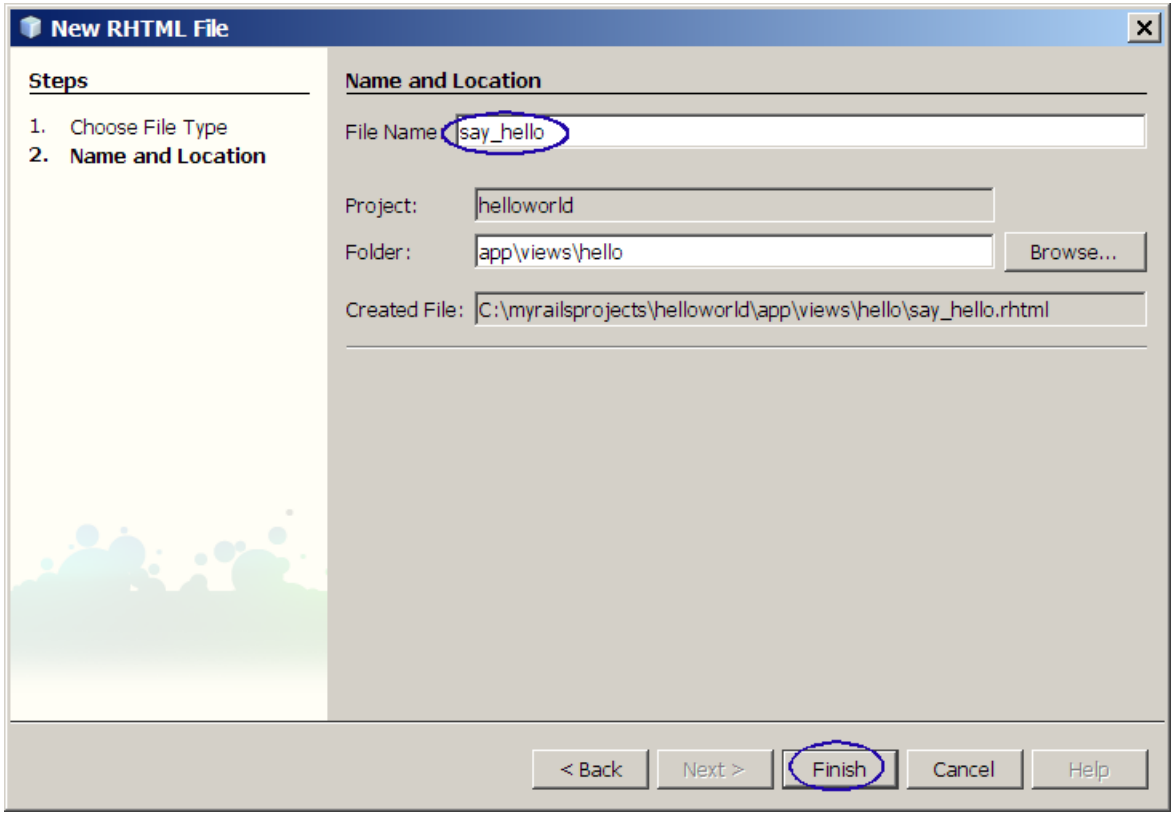
6. Write a View

What is a View?

- View is represented by a set of templates that get displayed.
- Templates share data with controllers through mutually accessible variables.
- A template can be either in the form of *.rhtml or *.erb file.
  - The *.erb file is searched first by Rails. If there is no *.erb file, then *.rhtml file is used.

Creating *.rhmtl file (or *.erb file)

- *.rhtml or *.erb file is created under the directory of /app/views/<controller>



Example *.rhtml file

- say_hello.rhtml

  My greeting message is <%= @hello.greeting %>
  *
*
  The current time is <%= Time.now %>

7. Set URL Routing

URL Routing

- The Rails routing facility is pure Ruby code that even allows you to use regular expressions.
  - Because Rails does not use the web server's URL mapping, your custom URL mapping will work the same on every web server.
  - *configuration/routes.rb* file contains the routing setting

routes.rb

```
ActionController::Routing::Routes.draw do |map|

  map.root :controller => "hello"

  # Install the default routes as the lowest
   priority.
  map.connect ':controller/:action/:id'
  map.connect ':controller/:action/:id.:format'
end
```

Thank you!

Sang Shin
http://www.javapassion.com
“Learning is fun!”