


Iteration

When you have a list of things, you often want to do something with each and every item in the list.

Here is a typical section of code that puts each item of an array onto the screen.

```
teams = []  
teams.push "Hawks"  
teams.push "Bears"  
teams.push "Cubs"
```

```
teams.each do |nickname|  
  puts nickname  
end
```



nickname is known as a
"block variable"

Core Rails Project Structure

```
app/  
  assets/  
  controllers/  
  mailers/  
  models/  
  views/  
  layouts/  
  
config/  
  application.rb  
  database.yml  
  routes.rb  
  environments/  
  
log/  
  
public/  
  
Gemfile
```

Rails Routes and Actions

Rendering a Response

1. Define a **route** that maps a url path to a specific **controller** and **action method**.
2. Define a **controller class**.
3. Inside the controller class, define an **action method**.
4. Call the **render** method to respond with a **status** code along with a **text** or **inline** body; or other headers.

```
class MyController < ApplicationController

  def hello
    render :text => "Hello!", :status => 200
  end

  def goodbye
    render :status => 302,
          :location => "http://apple.com"
  end

end
```

MVC Basics

4-Step Web Page Recipe

1. Define a **route** that maps a url path to a specific **controller** and **action**.
2. Define a **controller class**.
3. Inside the controller class, define an **action method**. This method is the best place to create **instance variables** that contain the data you want on your page. (You can use your **models** to retrieve the data you want.)
4. Write a **view template** that generates the HTML for your web page. You can use any instance variables you defined in Step 3. .

How To Use Route Placeholders

```
# config/routes.rb
MyApp::Application.routes.draw do

  get "products/:id", :controller => "products",
    :action => "show"

end
```

Placeholder



```
# app/controllers/products_controller.rb
class ProductsController < ApplicationController

  def show
    product_id = params[:id]
    @the_product = Product.find(product_id)
  end

end
```

Helpful Command-Line Tasks

rake

rake about

rake db:create

rake db:drop

rake db:migrate

rake db:rollback

rake db:seed

rake db:version

rake notes

rake routes

rake stats

rails new

rails server

rails console

rails dbconsole

rails runner

Adding a New Model In 5 Easy Steps

When you want to add a new model to your application, these are the basic steps you can follow.

See the following pages for details, too.

MODEL NAMES USE THE SINGULAR (“Movie”, not “Movies”).

Use the Model Generator

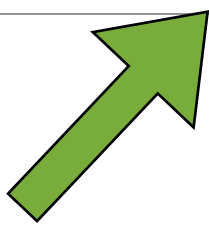
1. rails generate model Movie title:string year:integer
2. Open the model file that was generated (movie.rb). Notice how the generator took the liberty of whitelisting your columns for mass assignment.
3. Open the migration file (db/migrate/...create_movies.rb). Fix any typos, add columns that you forgot about when you did Step 1.
4. rake db:migrate
5. rails console. Check to make sure your class works as expected. Try to create new rows in your table by using your model class.

You could also handwrite the model class and only generate the migration with **rails generate migration...** instead.

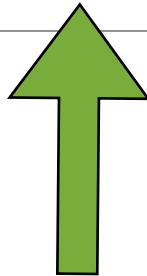
ActiveRecord

Generating a New Model

```
rails generate model product name:string color:string
```



Generator



Model
name



Model
Attributes

products		
id	integer	autoincrement
name	string	
color	string	