software
studio

data models in Rails

Daniel Jackson
schema management

schema
› like a type decl for a database
› lists tables and columns

writing a schema in SQL
› schema is created using CREATE TABLE operations

what if schema changes after deployment?
› make a new schema
› lose all data in database
› reimport data
class CreateProducts  ActiveRecord::Migration  
  def up
    create_table :products do
      t.string :name
      t.text :description
      t.timestamps
    end
  end
  
  def down
    drop_table :products
  end
end

idea
  › programmer doesn’t write schema
  › instead, writes incremental changes

migration
  › an incremental change to the schema
  › just a Ruby class with code to modify the schema
  › methods to make (up) and rollback (down) change
editing schemas

$ rails generate model Product name:string description:text

to create a table
› make a migration class (by hand, or with generator as above)
› run rake db:migrate

to change a table
› run rake db:rollback
› edit the migration
› run rake db:migrate

or
› create a migration for the edit
› run rake db:migrate
object relational mapping

class Product

name: “787”
description: “‘‘new ...batteries’’

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“787”</td>
<td>“new dreamliner with excitable batteries”</td>
</tr>
<tr>
<td>2</td>
<td>“747”</td>
<td>“long haul aircraft for 400 passengers”</td>
</tr>
</tbody>
</table>

table products
model declarations

class declaration
› created with migration by rails generate model
› need not mention columns

mass assignment
› a security vulnerability
› mark assignable columns with attr_accessible

http://www.example.com/user/signup?user[name]=ow3ned&user[admin]=1
params[:user]  # => {:name => “ow3ned”, :admin => true}

def signup
  @user = User new(params[:user])
end

class User  ActiveRecord::Base
  attr_accessible :name
end
using models

# Create a new product object
p = Product.new(:name => "787", :description => "dreamliner")

# Modify a product object
p.description = "bad dream liner"

# Save object as tuple in database table
p.save

# Find the product with primary key (id) 3
p = Product.find(3)
associations

class Customer

name: “Alice”
orders:

class Order

date: Feb 19, 2013
customer:

array

[0]:

table customers

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“Alice”</td>
</tr>
<tr>
<td>2</td>
<td>“Bob”</td>
</tr>
</tbody>
</table>

table orders

<table>
<thead>
<tr>
<th>id</th>
<th>date</th>
<th>customer_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Feb 20, 2013</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Feb 19, 2013</td>
<td>1</td>
</tr>
</tbody>
</table>
declaring associations

class Order < ActiveRecord::Base
  belongs_to :customer
end

class Customer < ActiveRecord::Base
  has_many :orders # note pluralization!
end

class CreateOrders < ActiveRecord::Migration
  def up
    create_table :orders do |t|
      t.references :customer
    end
  end
  def down
    drop_table :orders
  end
end
association types

**has_many + belongs_to**
- many-to-one relationship
- foreign key goes in table for class with belongs_to decl

**has_one + belongs_to**
- one-to-one relationship
- foreign key goes in table for class with belongs_to decl

**has_and_belongs_to_many x 2**
- many-to-many relationship
- need an extra ‘association’ table

Rails guide
- provides examples (but sadly lacks details and clarity)