

CRUD

Now This App Is Totally F'UCD

CRUD is a way of
thinking.

Create
Read
Update
Delete

**CRUD is a simple
architecture that frees
your mind to think.**

The Seven CRUD Methods

```
def list; end      # Show all items
def show; end     # Show a specific item
def new; end      # Form for creating new item
def create; end   # Create new item
def edit; end     # Form for editing new item
def update; end   # Update an existing item
def destroy; end  # Destroy an existing item
```

Those are your verbs.
Welcome to a world of
nouns.

Get every controller to
match those methods,
and you're done.

OK, Maybe not all will match. Some things just can't be CRUDy well.

Making CRUD

From the Ground On Up

Show and Make Things

```
def index
```

```
  @things = Thing.find_all
```

```
end
```

```
def new
```

```
  @thing = Thing.new
```

```
end
```

```
def create
```

```
  @thing = Thing.create(params[:thing])
```

```
  redirect_to :action => :index
```

```
end
```

index.rhtml

```
<%= render :partial => 'thing', :collection => @things %>
```

new.rhtml

```
<% form_for :thing, @thing, :url => { :action => :create } do %>  
  Name: <%= f.text_field :name %>  
<% end %>
```

Modify Things

```
def edit
```

```
  @thing = Thing.find(params[:id])
```

```
end
```

```
def update
```

```
  @thing = Thing.find(params[:id])
```

```
  @thing.update_attributes(params[:thing])
```

```
  redirect_to :action => :index
```

```
end
```

edit.rhtml

```
<% form_for :thing, @thing, :url => { :action => :update } do %>  
  Name: <%= f.text_field :name %>  
<% end %>
```

Get Rid of Things

```
def destroy
  @thing = Thing.find(params[:id])
  @thing.destroy
  redirect_to :action => :index
end
```

This is Basic Rails

Next is Where Things
Start To Get Messy

Adding Categories

```
class Thing < ActiveRecord::Base  
  has_and_belongs_to_many :categories  
end
```

```
class Category < ActiveRecord::Base  
  has_and_belongs_to_many :things  
end
```

How Do We Add Things
to Categories?

Put It In ThingsController!

```
def add_to_category
```

```
  @thing = Thing.find(params[:thing_id])
```

```
  @category = Category.find(params[:category_id])
```

```
  @thing.categories << @category
```

```
  redirect_to :action => :index
```

```
end
```

```
def remove_from_category
```

```
  @thing = Thing.find(params[:id])
```

```
  @category = Category.find(params[:category_id])
```

```
  @thing.delete(@category)
```

```
  redirect_to :action => :index
```

```
end
```

This Is Not CRUD!

Instead of shoehorning
this in, try to ensure
your app is CRUD.

CRUD Tip #1

Many-to-Many Associations need CRUD and new models.

Again, with CRUD

```
class Thing < ActiveRecord::Base
  has_many :categorizations
  has_many :categories, :through => :categorizations
end
```

```
class Category < ActiveRecord::Base
  has_many :categorizations
  has_many :things, :through => :categorizations
end
```

```
class Categorization < ActiveRecord::Base
  belongs_to :thing
  belongs_to :category
end
```

The Controller

```
class CategorizationsController < ApplicationController
  def create
    @categorization = Categorization.create(
      params[:categorization] )
    redirect_to :controller => 'things', :action => 'index'
  end

  def destroy
    @categorization = Categorization.find(params[:id])
    @categorization.destroy
    redirect_to :controller => 'things', :action => 'index'
  end
end
```

What's the big deal?
These are basically the
same methods in a
different place!

FALSE!

The new methods are
cleaner.

The architecture is
easier to read.

The association can
now have meaningful
properties.

OK, associations are a
prime place for CRUD.
Is that all it helps with?

Take a look at AJAX.

Creating with AJAX

```
def create_with_ajax
  @categorization = Categorization.create(
    params[:categorization] )
end
```

We've Seen This Before

```
def create
```

```
  @categorization = Categorization.create(  
                                params[:categorization] )
```

```
  redirect_to :controller => 'things', :action => 'index'  
end
```

```
def create_with_ajax
```

```
  @categorization = Categorization.create(  
                                params[:categorization] )
```

```
end
```

Duplication is bad, mkay.

So, let's get back to
CRUD.

Now With respond_to

```
def create
  @categorization = Categorization.create(params[:categorization] )
  respond_to do |type|
    type.html { redirect_to :controller => 'things' , :action => 'index' }
    type.js { render }
  end
end
```

CRUD Tip #2

Use `respond_to` for controlling view selection.

CRUD help anywhere
else?

Let's add users and then
see.

AccountController

```
def login; end      # CREATE a login  
def logout; end    # DESTROY a login  
def signup; end    # CREATE a user  
def edit; end      # UPDATE a user
```

The comments say it all.

CRUD Tip #3

Events are a great place to insert CRUD.

LoginController

```
def create
```

```
  @login = Login.create(params[:login])
```

```
end
```

```
def destroy
```

```
  @login = Login.find(params[:id])
```

```
  @login.destroy
```

```
end
```

The Models

```
class Event < ActiveRecord::Base  
end
```

```
class Login < Event  
  belongs_to :user  
end
```

```
class User < ActiveRecord::Base  
  has_many :logins  
end
```

Hey Idiot, I just made a
model for Logins. Argh,
CRUD wastes time!

Actually, look at what
you're now able to do.

LoginController
separates concept of
logging in from User
management.

Login model allows us
to track the user's logins
over time.

Modeling events gives
you information about
user behavior.

**Knowing your users
makes you money!**

Going from RPC to REST, “I cut the total number of actions by almost twenty.”

- Scott Raymond, *IconBuffer*

CRUD saves you
money!

It's got other benefits,
too.

simply_restful

A plugin for implementing verb oriented controllers.

Current URLs

```
def list; end      # GET things/list
def show; end     # GET things/show/ |
def new; end      # GET things/new
def create; end   # POST things/create
def edit; end     # GET things/edit/ |
def update; end   # POST things/update/ |
def destroy; end  # POST things/destroy/ |
```

simply_restful URLs

```
def list; end      # GET things
def show; end     # GET things/ |
def new; end      # GET things;new
def create; end   # POST things
def edit; end     # GET things/ |;edit
def update; end   # PUT things/ |
def destroy; end  # DELETE things/ |
```

Who cares? It's just a
URL.

It's prettier.
(And it makes better
use of HTTP)

It gives you
ActiveResource!

ActiveResource is an
easy to use API in
development by DHH.

Look at the source code or DHH's presentation to
learn more.

Recap

CRUD is an
architecture that helps
you organize your code
and remove obfuscation.

CRUD helps you make
money!

Other Resources

- DHH's presentation: www.loudthinking.com/lt-files/worldofresources.pdf
- Scott Raymond's blog: scottraymond.net/
- Ryan Daigle's article: www.ryandaigle.com/articles/2006/06/30/whats-new-in-edge-rails-activeresource-is-here