

Patrick Crowley

the.railsi.st

Migrating Legacy Data





**How do you migrate
your old data?**

- **Dump old database**
- **Massage text with TextMate, BBEdit, grep, etc.**
- **Eventually give up**
- **Import into new database**

Here's a better way.

Step 1:
Add new db adapter

/config/database.yml

development:

adapter: mysql
encoding: utf8
database: cinema_development
username: root
password:

...

legacy:

adapter: mysql
encoding: utf8
database: cinema_legacy
username: root
password:

Step 2: Add Legacy models

/app/models/legacy_base.rb

```
class LegacyBase < ActiveRecord::Base
  self.abstract_class = true
  establish_connection "legacy"

  def migrate
    new_record = eval("#{self.class}".gsub(/Legacy/, ''')).new(map)
    new_record[:id] = self.id
    new_record.save
  end
end
```

/app/models/legacy_architect.rb

```
class LegacyArchitect < LegacyBase
  set_table_name "architect"

  def map
    {
      :first_name => self.name_first,
      :last_name  => self.name_last,
      :description => self.desc_long
    }
  end
end
```

Step 3: Add Migration helper

/lib/migration_helper.rb

```
def migrate(name, options={})
  # Grab custom entity label if present
  label = options.delete(:label) if options[:label]

  unless options[:helper]
    model = name.to_s.singularize.capitalize
    model.constantize.delete_all

    puts "Migrating #{number_of_records || "all"} #{label || name} #{"after #{offset_for_records}" if
offset_for_records}"
    "Legacy#{model}".constantize.find(:all, with(options)).each do |record|
      record.migrate
    end
  else
    eval options[:helper].to_s
  end
end

def with(options={})
  {:limit => number_of_records, :offset => offset_for_records}.merge(options)
end

def number_of_records
  nil || ENV['limit'].to_i if ENV['limit'].to_i > 0
end

def offset_for_records
  nil || ENV['offset'].to_i if ENV['offset'].to_i > 0
end
```

Step 4:

Add Rake task

/lib/tasks/migrate.rake

```
require 'migration_helper'
```

```
namespace :db do
```

```
  namespace :migrate do
```

```
    desc 'Migrates legacy content'
```

```
    task :legacy => ["architects", "styles"]
```

```
    desc 'Migrates architects'
```

```
    task :architects => :environment do
```

```
      migrate :architects
```

```
    end
```

```
    desc 'Migrates theaters'
```

```
    task :theaters => :environment do
```

```
      migrate :theaters
```

```
    end
```

```
  end
```

```
end
```

Step 5:
Start migrating

Demo

**[http://github.com/
mokolabs/legacy/](http://github.com/mokolabs/legacy/)**

Special thanks

PragDave

Dave Thomas—Pragmatic Programmer

« [The Water Fountain Methodology](#) | [Main](#) | [PyCon's Coming to Town](#) »

January 04, 2006

→ Sharing External ActiveRecord Connections

I had a problem with a deployed Rails application today that Mike Clark helped me track down.

It started after Mike and I migrated a Rails app to a new, zippier server. Everything looked great (and the app ran a lot faster). However, after a while, I started getting problems with a database. To explain what went wrong, I need to chat a little about the application.

This piece of code is an attempt to migrate functionality away from a particularly nasty set of old CGIs, web applications, bailing wire, chewing gum, and other historical crud. There's a lot of existing functionality, and I decided not to try to port it all across to Rails on day one. Instead, I gave the Rails application its own database to play with, and gave it access to the old legacy database for when it needed to look up stuff in the existing system. To do that, I created a set of ActiveRecord classes for the legacy tables, and explicitly connected them to the legacy database, rather than the default Rails application database. A typical legacy model class looked something like this:

```
class LegacyOrder < ActiveRecord::Base
  set_table_name "orders"
  set_primary_key "o_id"

  establish_connection "legacy_#{RAILS_ENV}"
  ...
end
```

(The "legacy_#{RAILS_ENV}" stuff lets me have development, test, and production versions of the legacy database in my database.yml file.)

Now in Beta



Third Edition, Covering Ruby 1.9, now [in beta](#)



Questions?

The End

