Chado on Rails

a framework to simplify development on the Chado schema

Justin Reese / Chris Childers
Some links:

These slides:

http://tinyurl.com/chadoonrails

Source code, have a look:

svn co http://chadoonrails.rubyforge.org/svn/trunk
The Big Idea

Chado database
(de facto standard in bioinformatics)

Chado on Rails

Your application
(gene pages, GFF loader, anything that needs to talk to Chado)
This time last year...

Bovine Genome Database status talk:

no gene pages!
How to implement gene pages?

- couldn't find existing tool that fit our needs

- why not roll our own?

- why not use Ruby on Rails?
The case for Ruby on Rails

- implement object-relational mapping for Chado database schema as legacy schema

- what we then get for free:

  Mature MVC framework for fast web development on top of Chado
  Rake tasks (a powerful tool for automating tasks)
  Capistrano (another powerful tool for deployment, automating tasks)
  Powerful debugging (stop request in stack wherever you want)
  Sophisticated testing suite
  Much more...

- BGD, Beebase, HymenopteraBase, other groups can build on this
Ruby

- "Perl's younger prettier sister"

- syntactic sugar, easy metaprogramming, pure object-oriented language, other sundry awesomeness

- A great 15 minute tutorial: http://tryruby.org/

Rails legacy database support

- Rails paradigm: "convention over configuration"
  e.g. Model name "Feature" == table name "features"

- Chado doesn't follow Rails convention
  e.g. Feature model == Chado table name "feature"

- Legacy support for Chado means overriding these conventions, explaining to Rails how to talk to Chado
**MVC paradigm**

- **Model** - the representation of your data (doesn't change much after development)

- **Controller** - "business logic", i.e. the logic to retrieve and process information coming and going from view (changes more seldom)

- **View** - presentation of information to user (changes every 3 seconds)
Quick rails anatomy lesson

Model files live in:
trunk/app/models/[model_name].rb

Controller files live in:
trunk/app/controllers/[model_name]_controller.rb

View files live in:
trunk/app/views/[model_name]/

One important config file:
trunk/config/database.yml - describes db conn.
Legacy Chado support - things to change

"Feature" model as an example:

In `trunk/app/models/feature.rb`:

```ruby
set_table_name "feature"  # default is "features"
```

This line tells Chado to look for features in table "feature", not table "features":

```ruby
set_primary_key "feature_id"   # default is "id"
```

This line tells Chado to use "feature_id" not "id" as primary key:

Repeat for each Chado tables we will support
Hooking up tables to each other ("associations")

Feature -- belongs to --> Cvterm

Cvterm <-- has many -- features

Easy:

in trunk/app/models/feature.rb, add this:

```ruby
belongs_to :cvterm
```

in trunk/app/models/cvterm.rb, add this:

```ruby
has_many :features, :foreign_key => "type_id"
```
We now have a powerful Chado console

Example: show me the name of the CV for the last thing loaded into Chado feature table

(try out by starting up trunk/script/console)

Before, using SQL:

```
SELECT cv.name FROM cv, cvterm, feature WHERE cv.cv_id = cvterm.cv_id AND cvterm.cvterm_id = feature.type_id ORDER BY feature_id DESC LIMIT 1;
```

After, using Rails:

```
Feature.find(:last).cvterm.cv.name
```
Example app: 5 min. Chado gene pages

**model:** done already

**controller:** make controller, write method to load up gene of interest into variable:

```
[~/.chado_on_rails/trunk] jtr4v$ ./script/generate controller gene
  exists app/controllers/
  exists app/helpers/
  create app/views/gene
  exists test/functional/
  exists test/unit/helpers/
  create app/controllers/gene_controller.rb
  create test/functional/gene_controller_test.rb
  create app/helpers/gene_helper.rb
  create test/unit/helpers/gene_helper_test.rb
```

then add this to trunk/app/controllers/gene_controller.rb:

```
def show
  @gene = Feature.find(params[:id])
end
```

**view:** add trunk/app/views/gene/show.html.erb, present some gene info to user (next slide)
Simple view for gene page

<%= render :partial => 'header' %>

<% if @gene.nil? %>
There is no record for feature id <%= params[:id] %> 
<% else %>
<% end %>

<% else %>
<ul>
<li>Name: <%= @gene.name %></li>
<li>Uniquename: <%= @gene.uniquename %></li>
<li>Feature id: <%= @gene.feature_id %></li>
<li>Cvterm: <%= @gene.cvterm.name %></li>
<li>Organism:.googleapis ogspecies"%></li>
<li>Time last modified: <%= @gene.timelastmodified %></li>
</ul>

<% end %>

<br/>
<%= render :partial => 'footer' %>
Chado on Rails + Chado db + 19 lines of code = gene pages!

http://0.0.0.0:3000/gene/show/17030045
Mature BGD genes pages

http://genomes.arc.georgetown.edu/bovine/genepages/genes/BT10609
Testing

- Rails ships with sophisticated testing suite (a whole other talk)
- Chado on Rails contains unit tests for associations, other reality checks
- Easy to add more sophisticated tests (functional, etc.)
- `rake test`

Caching

- Add one line in controller to cache gene pages:
  ```ruby
  caches_page :show
  ```
- More sophisticated caching (expire cache after x days, caching expensive queries, etc.)
Deployment - a non-issue

- Used to be a major weakness of Rails

- Passenger Phusion + Capistrano provide dead-easy deployment, plays nicely with Apache and Nginx

- Fairly easy to set up staging environments to try out new releases
Other applications

- Powerful framework for the development of applications on top of Chado

- Ideas for possible applications?

- Help available from me, Chris, Rails community

- http://rubyforge.org/projects/chadoonrails/ (there's another dead project by the same name)

- Justin Reese: justaddcoffee@gmail.com
- Chris Childers: genetics.guy@gmail.com
Thanks

Chris Childers
Chris Elsik
Jay Sundaram
Darren Hagen
Monica Muñoz-Torres
Steve Shen
Anna Bennett
Shu Tao
Don Vile