“Component” is not a dirty word

Composing your Rails UI with stateful objects

(beginning of overview)

Our Mission

- Extend Rails because...
  - ...Rails provides shared partials
  - ...Rails provides view helpers
  - ...Rails wraps a few JS controls
- But these are largely just generators
- We want more
  - We want better toys to build our user expeiences!

Better toys!

- (Everything I need to know about programming I learned in kindergarten...)
- Not lame toys
  - Like just
    - check_box, file_field, label, text_area, etc.
- Awesome toys!
  - Grids, trees, slideshows, tabs, user-defined components, wrappers on JS lib widgets like YUI, etc.

“No toys for you!”

Yes, web UI's are drastically constrained, offer a paucity of controls, and enforce a brutally linear control flow; and these are good things.
– Tim Bray

Whatever!

- When left to their own devices, users create rich interfaces with sophisticated capabilities.
- See MySpace profiles for millions of examples. (Embedded games, music players, slideshows, etc.)
- After all, why should the web browser and the OS it runs in have powerful widgets, but the web page not?

What we want

- To create actions/ views that are easy to re-use
    - *Double rendering is not an error!*
- To have view helpers that use *templates*
    - Not bizarre pseudo-markup helper-speak
- To model our views in code as what they are: compositions of sub-views
    - Not just monolithic pages
- To bring the abstractions of OOP to our UI's
    - Not just to be dumped at <html>

But wait, we want even more!

- To have complex UI widgets that manage their own state
- To code user interactions via *events*
- To have widgets know how to pass messages to each other
- To use the abstractions commonly employed in GUI toolkits

The cool kids have it!

- Cocoa (including iPhone SDK)
- WebObjects and its descendants
  - Seaside (Smalltalk)
  - Iowa (Ruby)
  - Tapestry (Java) (marginally cool)
- Erlyweb (Erlang), Lift (Scala)
- Many, many, others

Make apps **and** widgets

- As we develop, we are building reusable parts for future apps...
- ...while still having Rails with all of its advantages (the Golden Path!)
- We build a collection of bigger, better, toys!

How do we get there?

- Rails plugins
  - *Cells*: Free your views from the tyranny of the monolithic render
  - *Apotomo*: (optionally) Adds automatic state management, message passing and events for UI widgets

Why not?

Aim to have the best of Seaside
and the best of Rails!

(end of overview)

So, who's the bald guy?

Mike Pence, Sarasota, FL

mike.pence@gmail.com

http://twitter.com/mikepence
http://mikepence.wordpress.com

15 years of cubicle incarceration for Big Name Corporations; Independent Rails developer since '05

Specializing in project direction, cloud computing, and rich web interfaces.

Now booking 1Q '09 projects :)

Meet Cells

Let's make a Cell together

```
rails hello_cells

...

cd hello_cells

script/plugin install
  git://github.com/apotonick/cells.git

# Update bootstrap section of environment.rb, add:

require File.join(File.dirname(__FILE__), '../vendor/
  plugins/cells/boot')
```

Let's make a Cell together

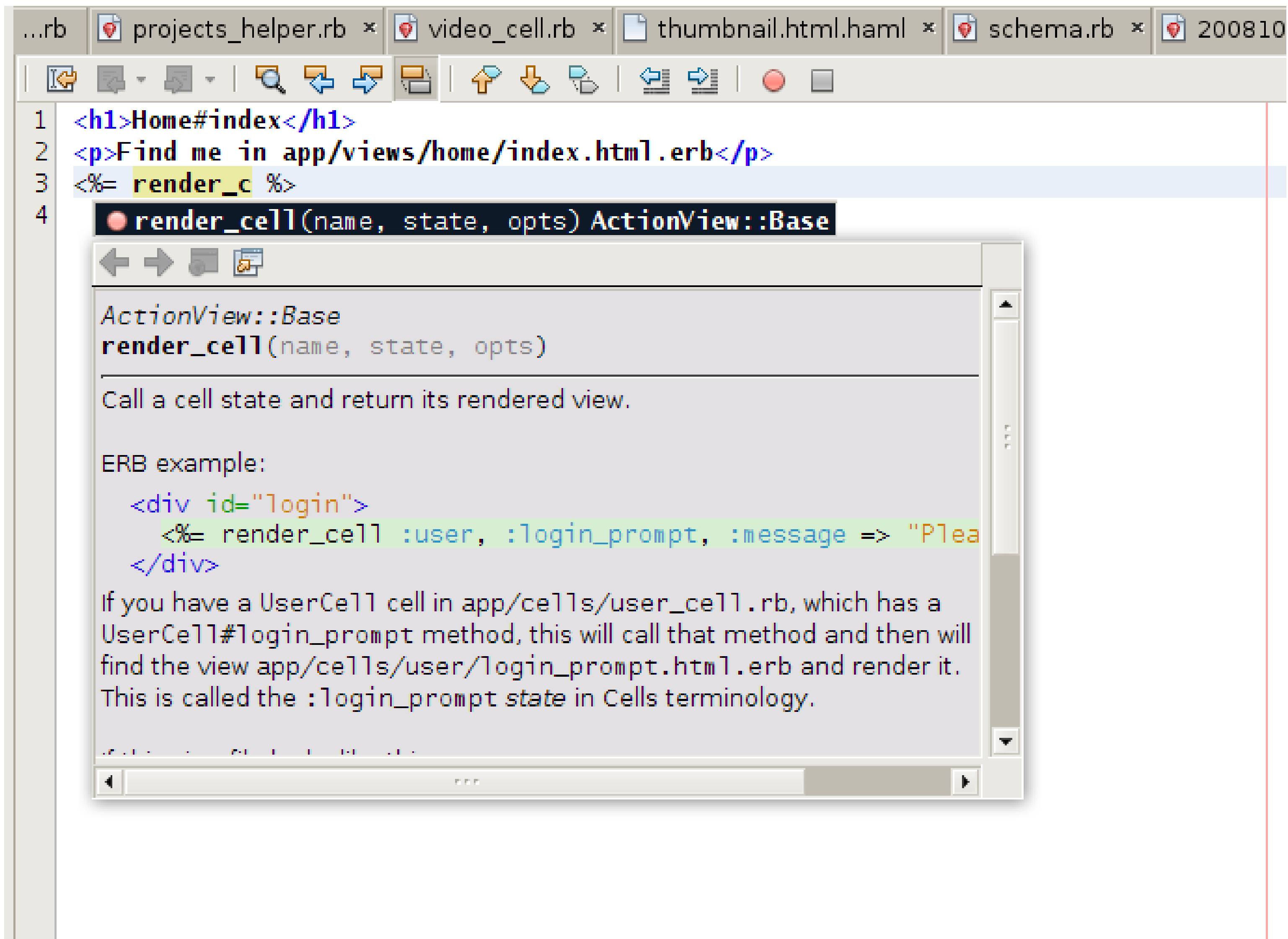
```
script/generate controller home index
...
script/generate cell hello rubyconf
      create  app/cells/
      create  app/cells/hello
      create  app/cells/hello_cell.rb
      create  app/cells/hello/rubyconf.html.erb
```

Let's make a Cell together

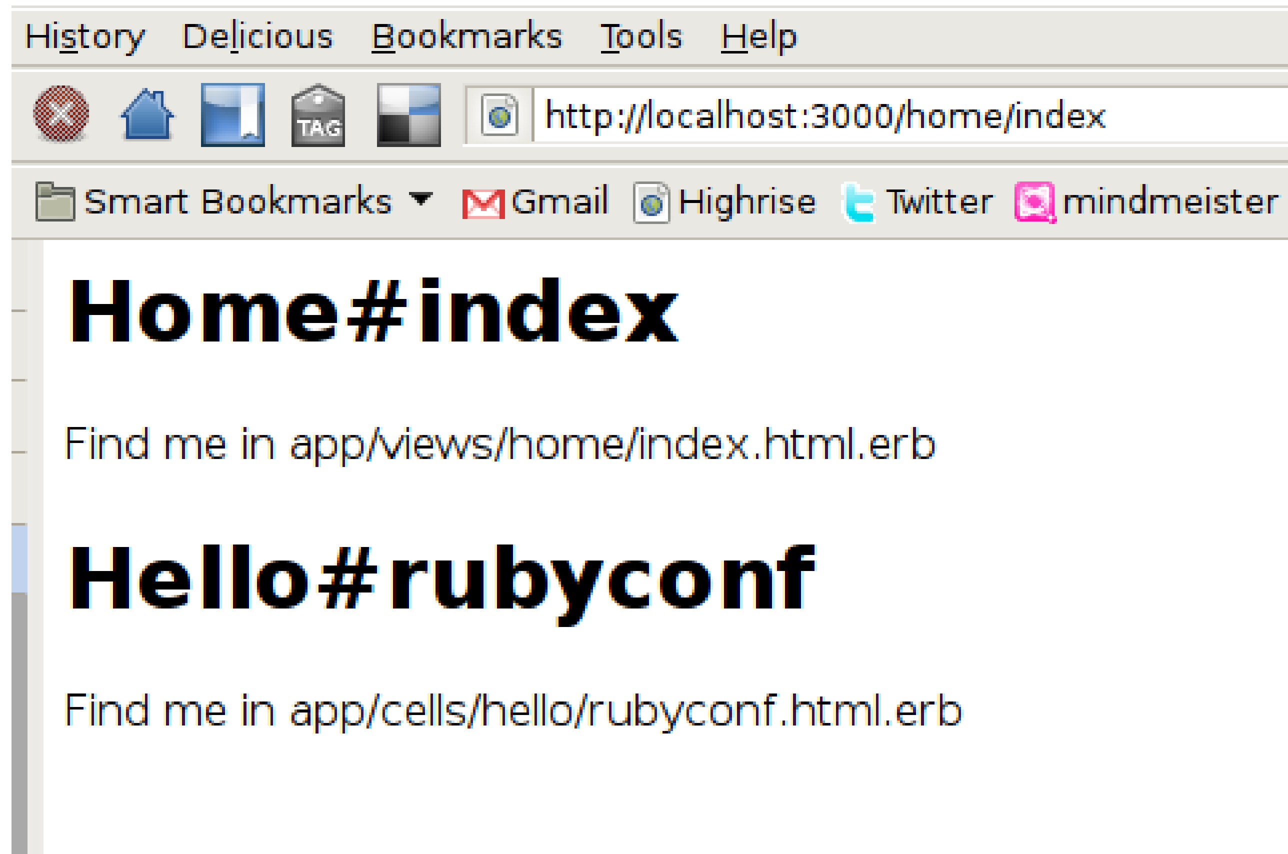
```
# Edit app/views/home/index.html.erb, add:

<%= render_cell :hello, :rubyconf %>
```

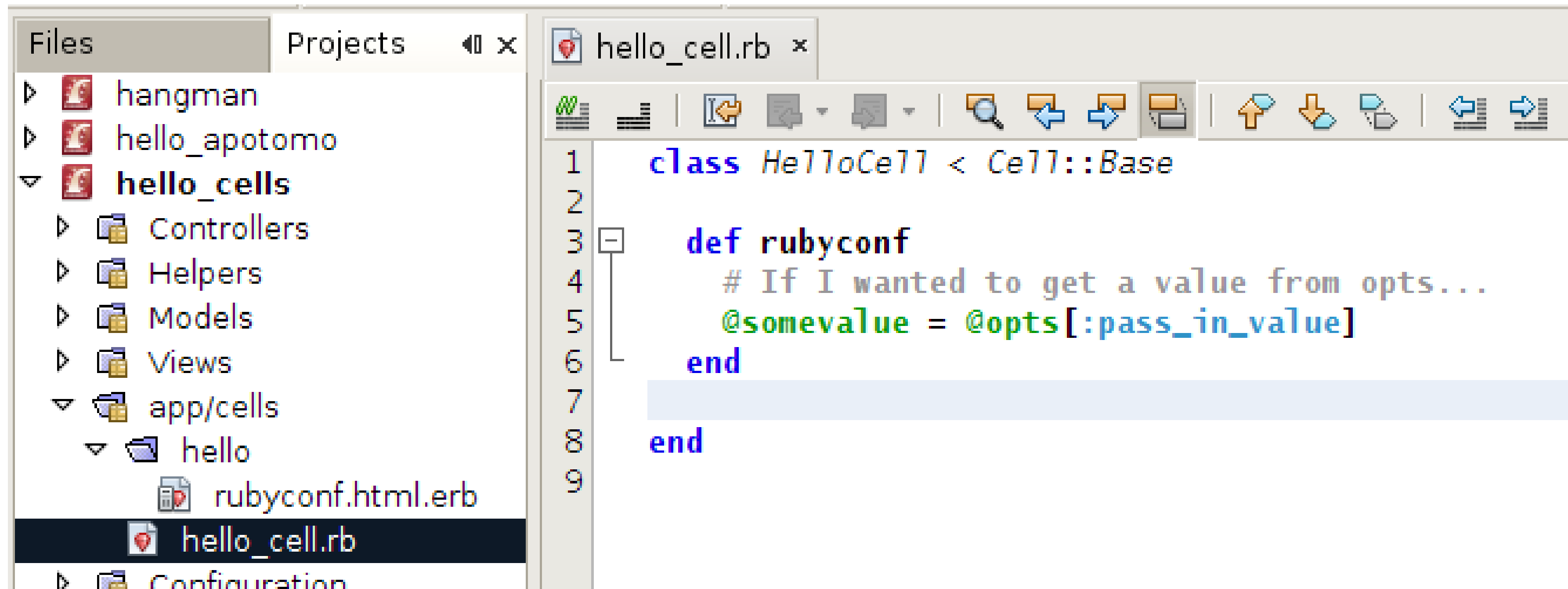
Let's make a Cell together



Let's make a Cell together



A little Cell-ular anatomy



What Cells give us

- Extra rendering cycles
- A great place to code re-usable UI elements

When to use Cells

- When you would use a shared partial
  - But you want better encapsulation
  - And/or you want actions, or states, for a sub-view
- When you would use a view helper
  - But with the advantage of having a template
- When you want easy view re-use within or across apps
  - Packagable via Engines

Cells: Free your views!

- http://cells.rubyforge.org/
- http://rubyforge.org/mail/?group_id=2978
- #cells on freenode
- http://github.com/apotonick/cells/tree/master

But...

- Cells, like Rails itself, get you through the initial rendering of the UI
- However, they don't maintain state or otherwise help you to build better Javascript apps

How can we do more?

- Meet Nick Sutterer, all the way from Freiburg, Germany, to tell us how to put *living objects* in our views...