Ruby On Rails

James Reynolds

Today

* Ruby on Rails introduction
* Run Enviornments
* MVC
* A little Ruby
* Exercises

Installation

* Mac OS X 10.5 will include Rails
* Mac OS X 10.4 includes Ruby
  * Most people reinstall it anyway
* From scratch
* Drag and drop Locomotive

Databases

* Mysql
* SQLite
* PostgreSQL
* DB2
* Oracle
* Firebird
* SQL Server
* more

Webservers

* Apache w/ FastCGI or Mongrel
* LightTPD
* WEBrick

"IDE's"

* TextMate and Terminal (preferred)
* RadRails (Eclipse)
* jEdit
* Komodo
* Arachno Ruby
* NetBeans IDE

Run Environments

* Production
* Development
* Testing

Run Environments

* Production
  * Cached
  * Freeze Rails
  * Ship Rails with your app
  * etc

Run Environments

* Development
  * Reloads source files every time
  * Scaffold

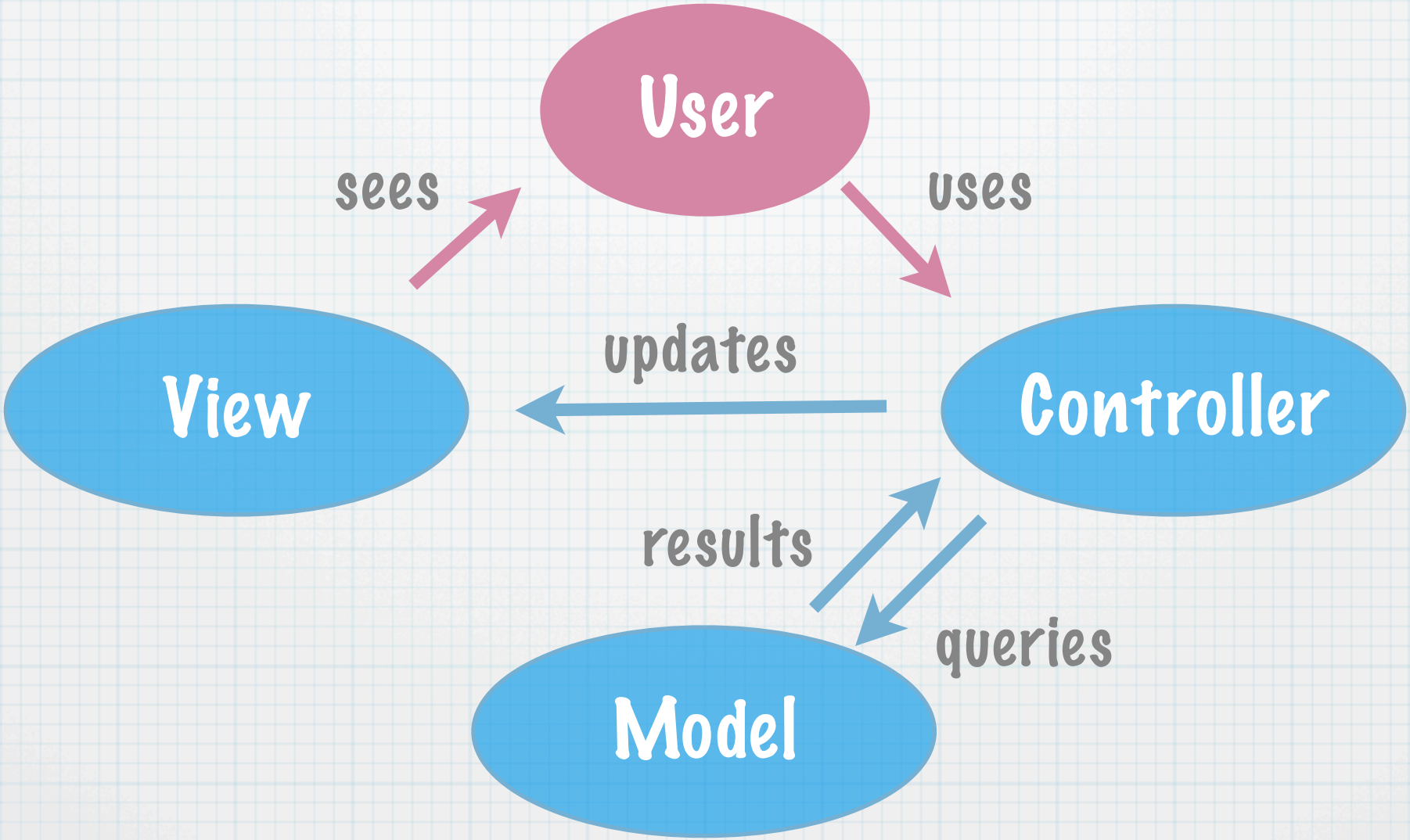
Run Environments

* Testing
  * Connect debugger to running webapp
    * Stop at breakpoints
  * Unit testing
    * Integration testing
    * Functional testing
    * DB is reloaded w/ each test
    * Mock and stub code

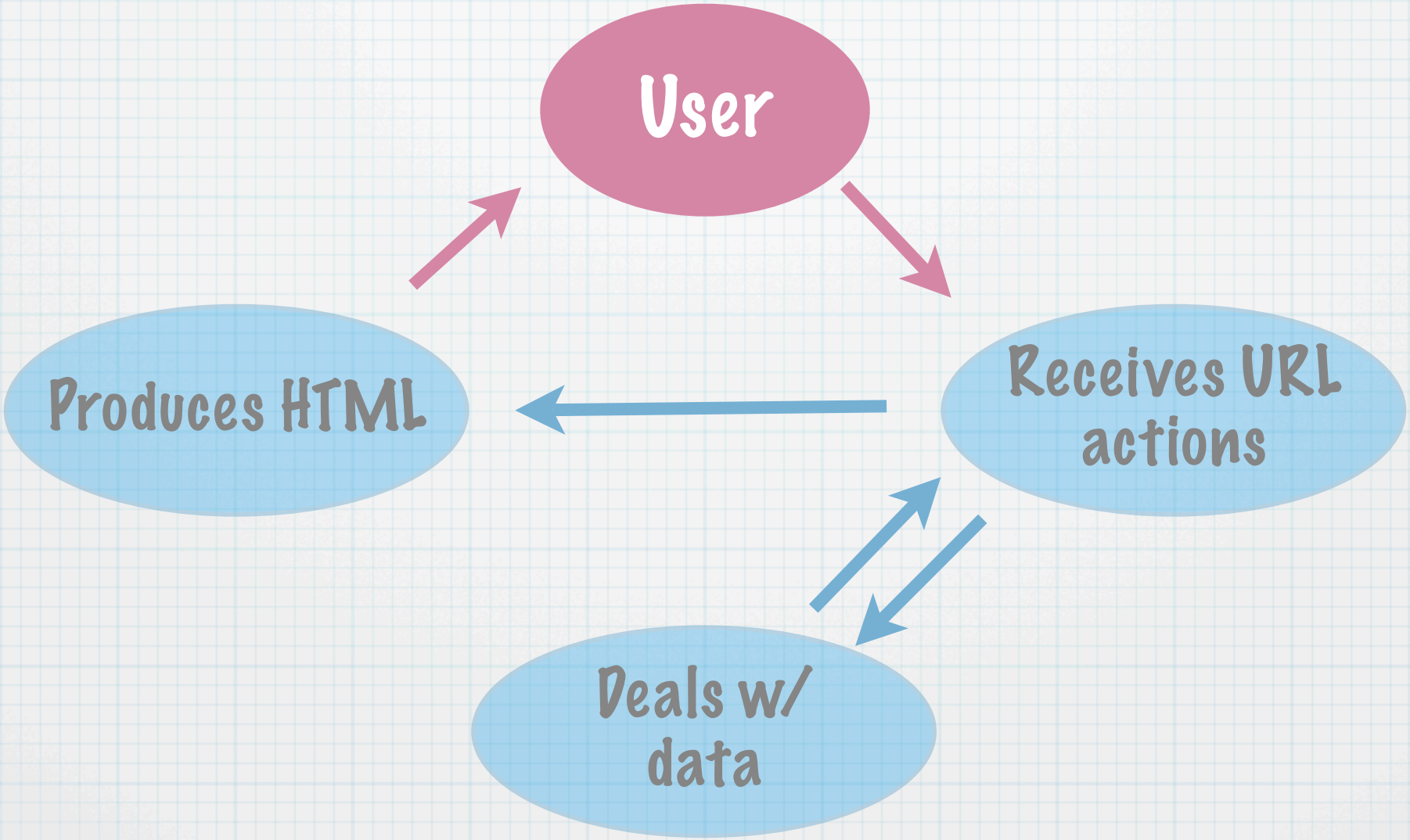
MVC

- First Described in 1979
- Totally ignored in web dev
  - Except
    - WebObjects
    - Struts
    - JavaServer Faces

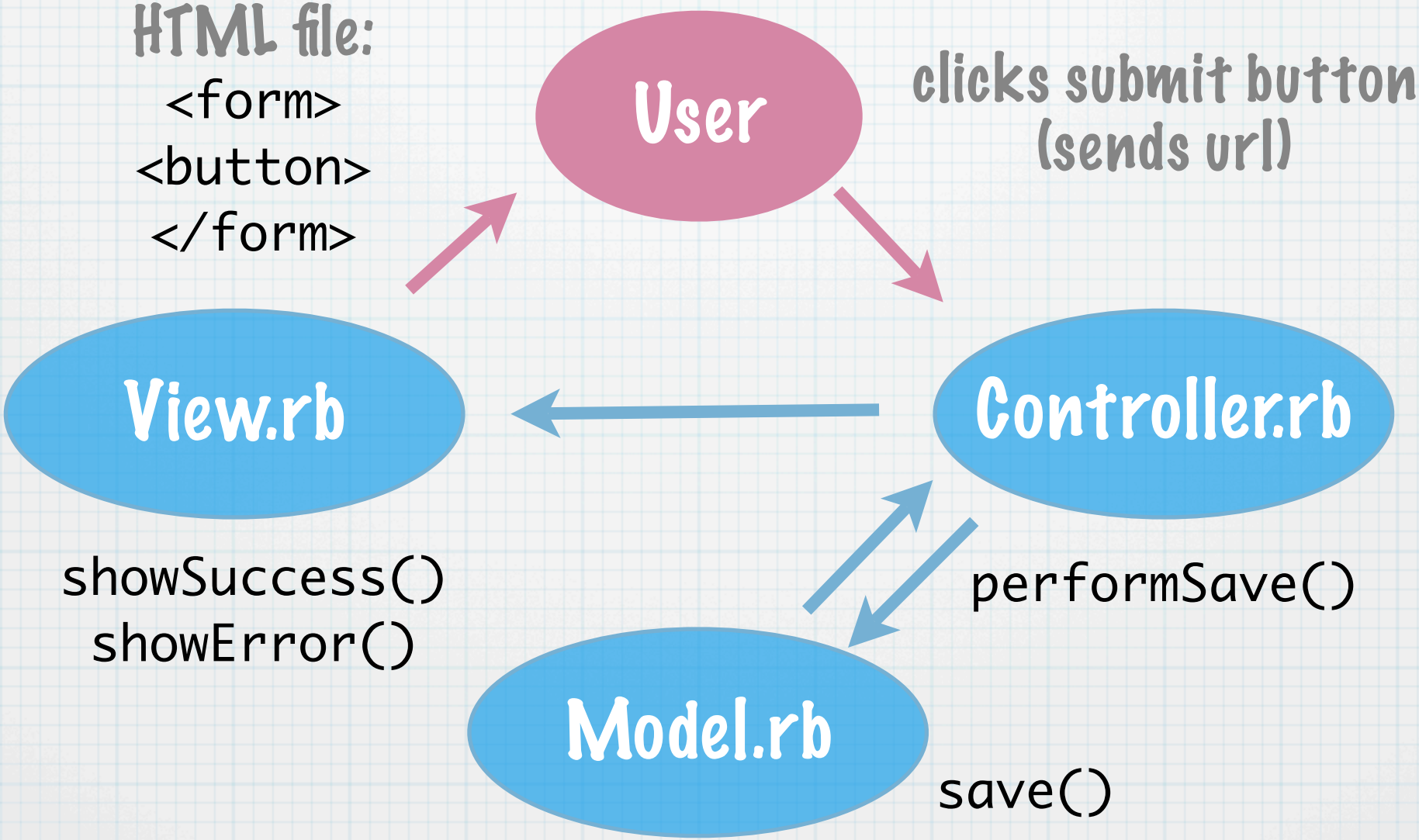
Model-View-Controller



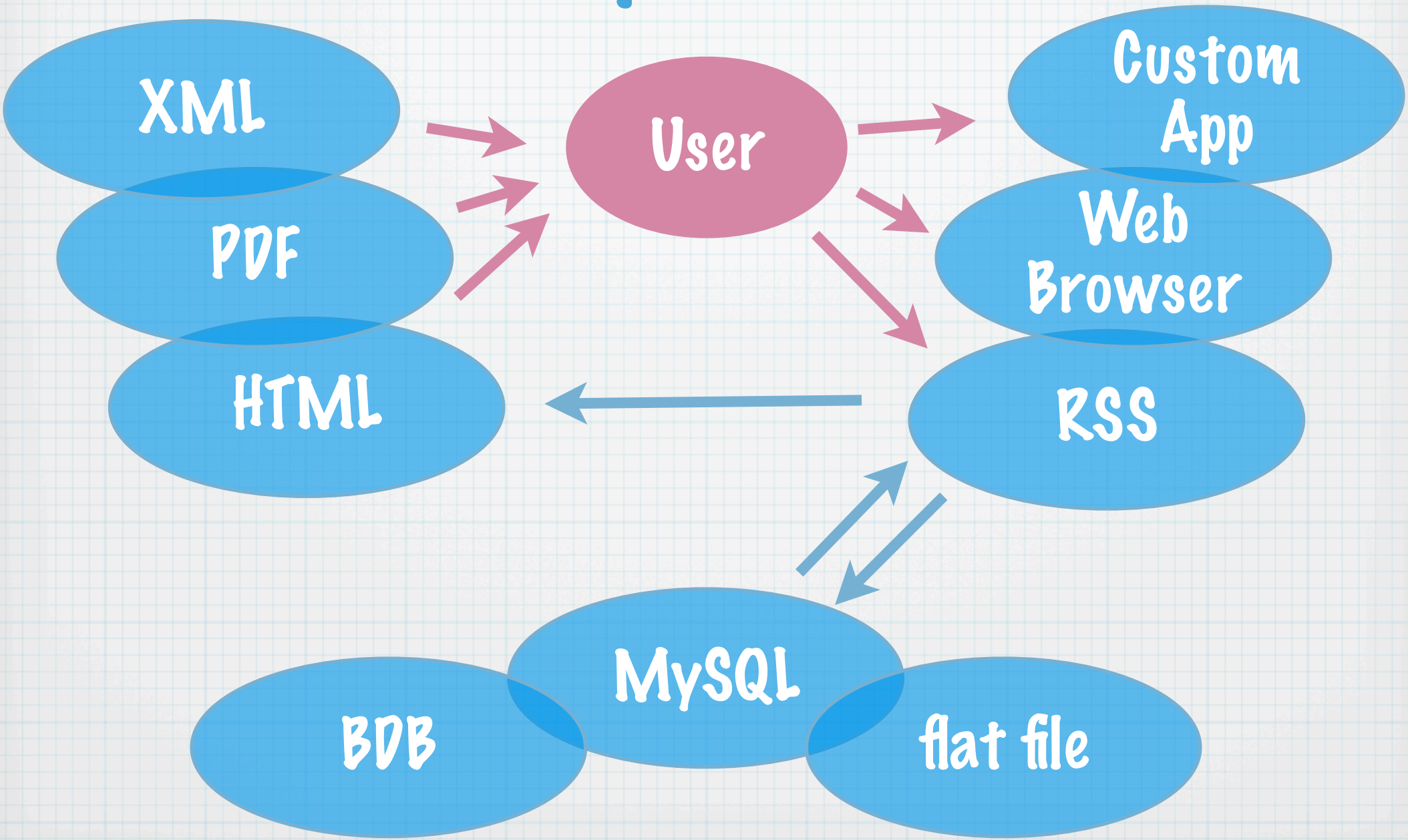
How it works



How it works


HTML file:
<form>
<button>
</form>
User
clicks submit button
(sends url)
View.rb
Controller.rb
showSuccess()
showError()
performSave()
Model.rb
save()

Why use it?


XML
PDF
HTML
User
Custom App
Web Browser
RSS
MySQL
BDB
flat file

Controllers

* Friendly urls
  * NO: http://example.com/?node=34
  * Yes: http://example.com/blog/view/34
* Customizing URL's easy
  * config/routes.rb

Controllers

- Default URL mapping
  - http://example.com/blog/view/34
    - controller = blog
    - action = view
    - id = 34

Views

* Can create html, xml, and js
* Easy handling of params, flash, cookies, etc
* Ajax built in and dumb simple

Views - RHTML

* Mixing up <% ... %> and <%= ... %> is a HUGE source of bugs
* Be sure to put spaces around the tags
  * Not required, but is easier to read IMO

A little Ruby

* Symbols
* Default parameter values
* Named parameters

Symbols

```
moods = { 'angry' => 'red',
          'sick'  => 'green',
          'sad'   => 'blue' }

puts "I'm feeling " + moods['sick']

moods = { :royal   => 'purple',
          :angelic => 'white',
          :guilty  => 'black' }

puts "I'm feeling " + moods[:royal]
```

Named parameters

```
def printParams lotsaParams = {}
    if lotsaParams[:one]
      puts lotsaParams[:one]
    end
    if lotsaParams[:two]
      puts lotsaParams[:two]
    end
end
printParams :one => '1', :two => '2'
printParams :two => '2', :one => '1'
```

Common Errors!

* Do not press return before "=>"
  * Wrong:

```
printParams :one
=> '1', :two => '2'
```

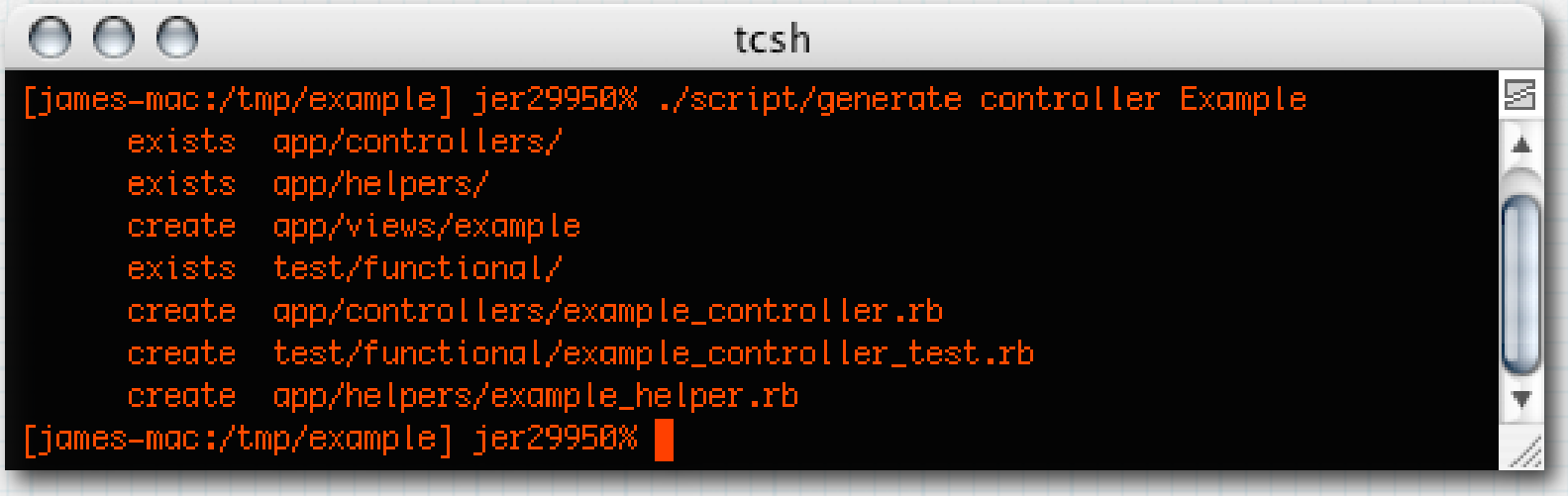
* Missing or misplaced ":" causes errors!
  * Wrong: `printParams one => '1',: two => '2'`
  * Right: `printParams :one => '1', :two => '2'`
* Space or return between hash/array and `[...]`
  * Wrong: `moods [:royal]` Right: `moods[:royal]`

Try it ourselves

* Make a webapp that says "Hello World" on the index page
* Create the app (in Locomotive)

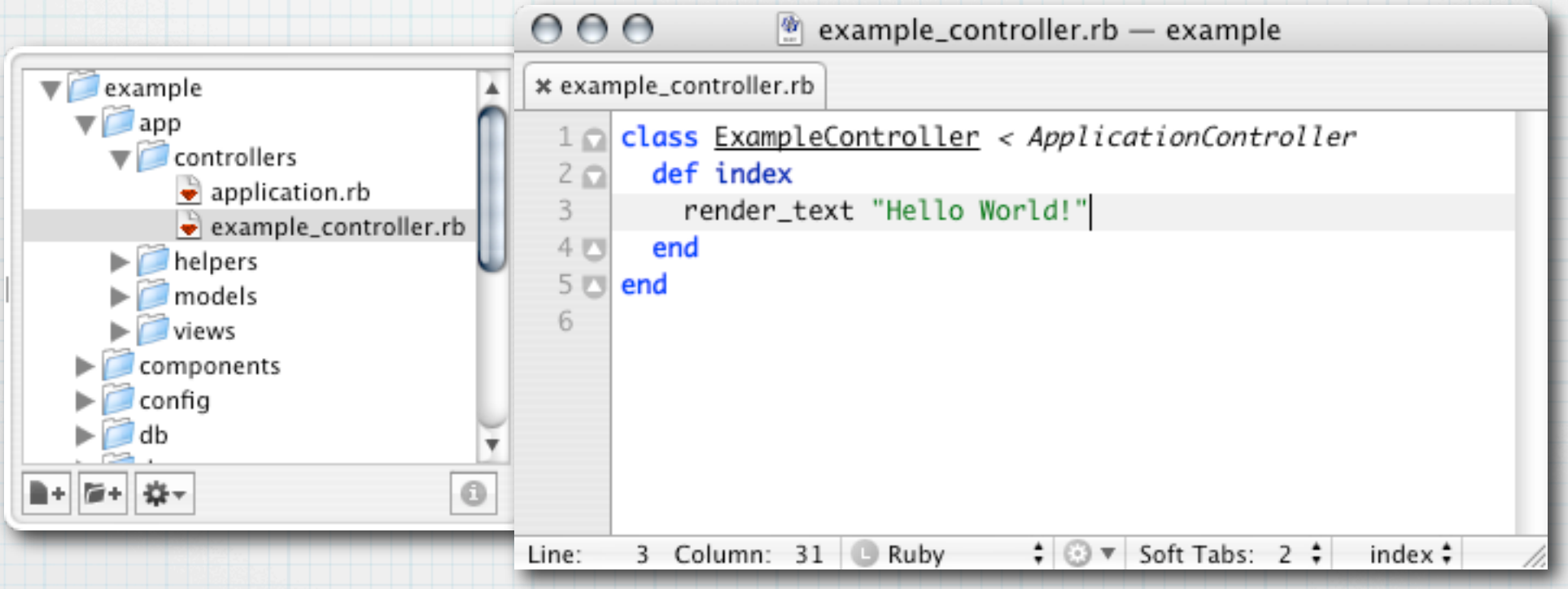
Hello World

* Create the controller (in Terminal)



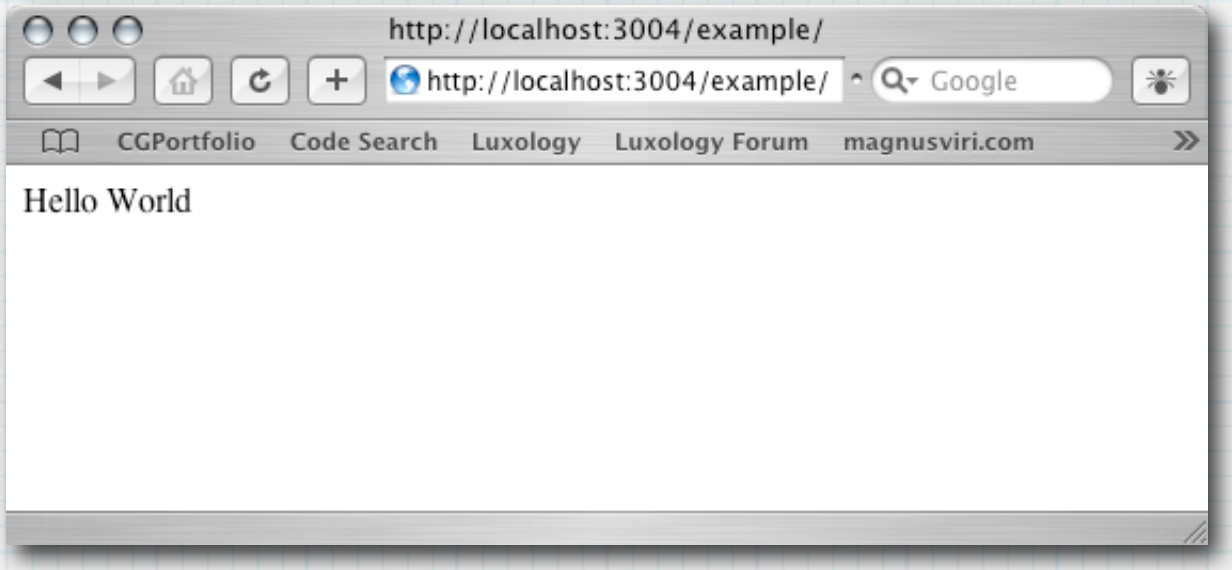
Hello World

* Open the file "example_controller.rb" (in TextMate)
* Create the index method



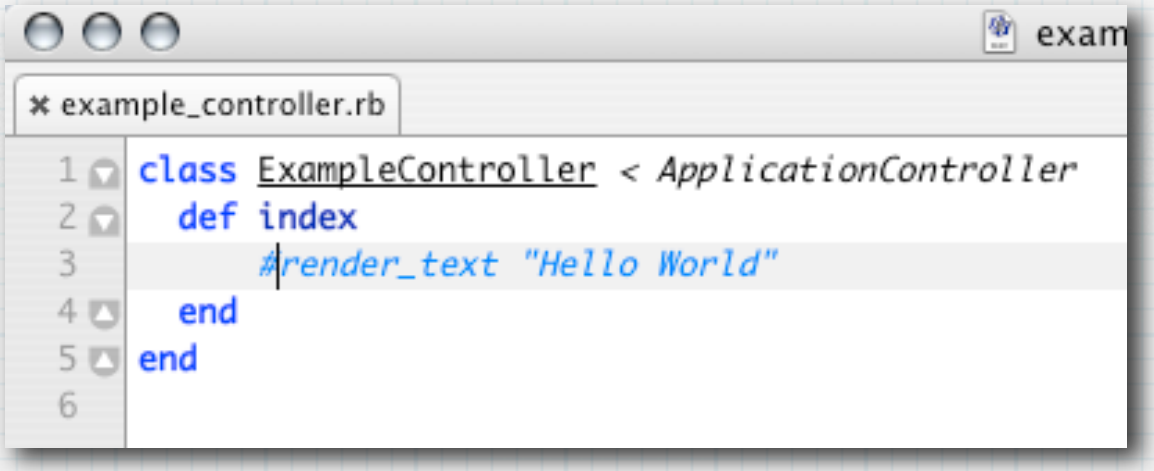
Hello World

* View in web browser!



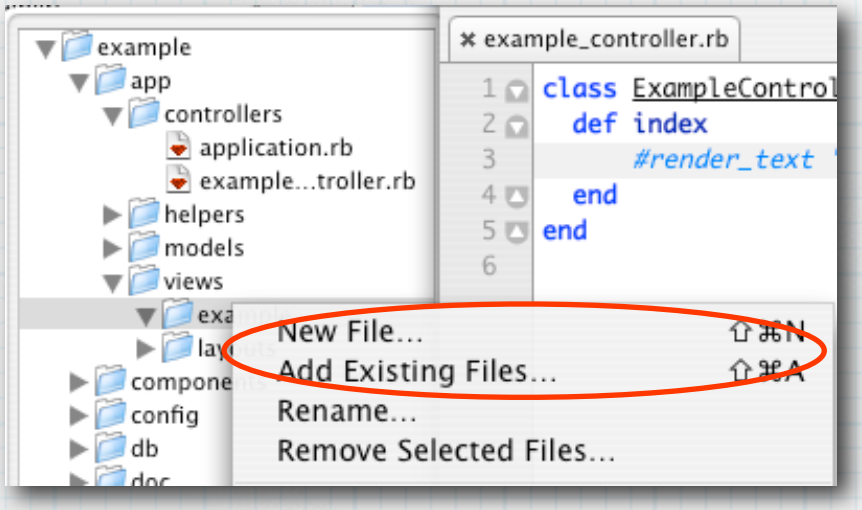
Hello World via template

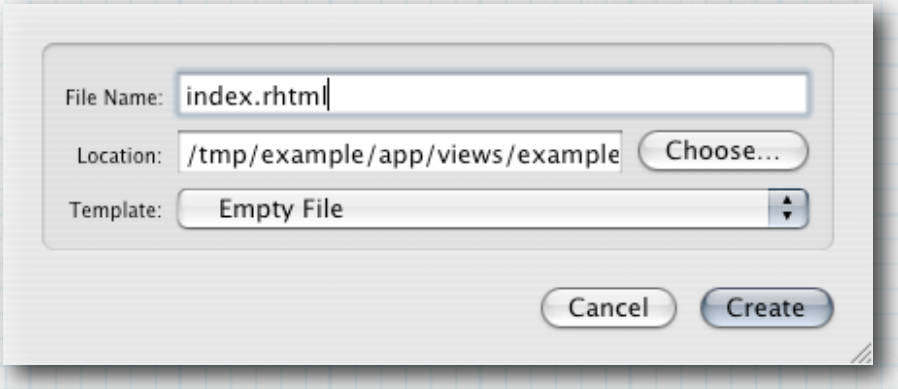
* Comment out "render_text" line



Hello World via template

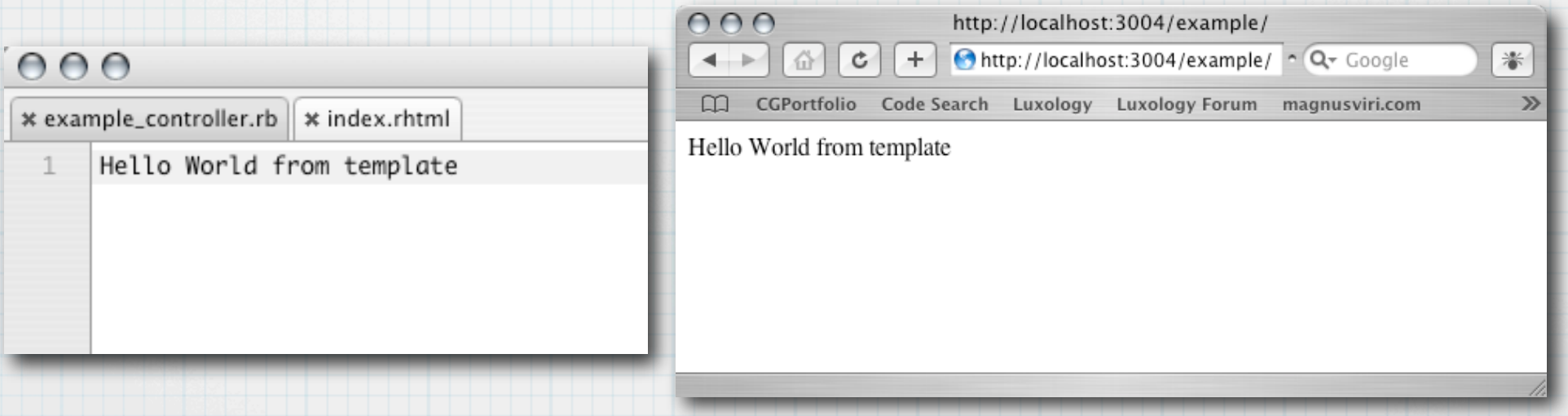
* Create new view file "index.rhtml"
  * Right click on "views/example" dir





Hello World via template

* Insert HTML into "index.rhtml"
* Preview in browser



Request Params

* Print out the request params
  * Add this to index.rhtml
  * Pay attention to <% vs <%=

```
<% params.each_pair do |key,value| %>
 <%= key.to_s %> =&gt; <%= value.to_s %><br/>
<% end %>
```

Try it ourselves

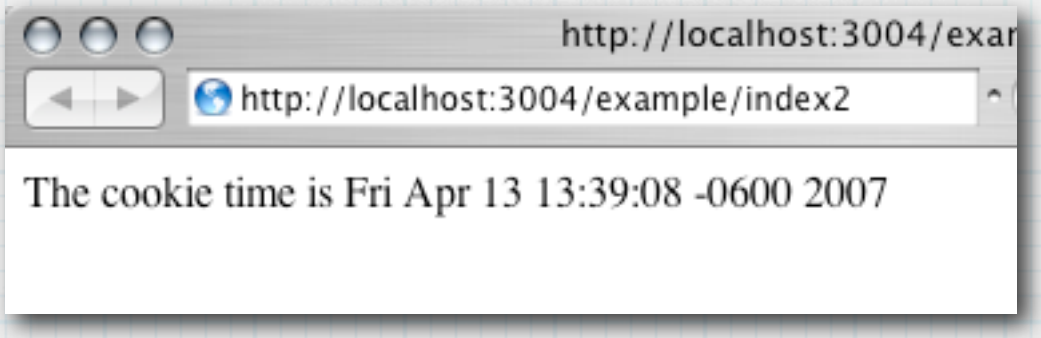
* Print out the request params
  * It looks like this
    * (action & controller always sent as params)

Cookies

* Change index_controller.rb to this:

```
class ExampleController < ApplicationController
  def index
    cookies[:the_time] = Time.now.to_s
    redirect_to :action => :index2
  end

  def index2
    render(:text => "The cookie time is #{cookies[:the_time]}")
  end
end
```



Make a Session

* Put this in example_controller.rb

```
class ExampleController < ApplicationController
  def index
    session[:counter] ||= 0
    session[:counter] += 1
  end
  def reset_counter
    session[:counter] = 0
    redirect_to :action => :index
  end
end
```

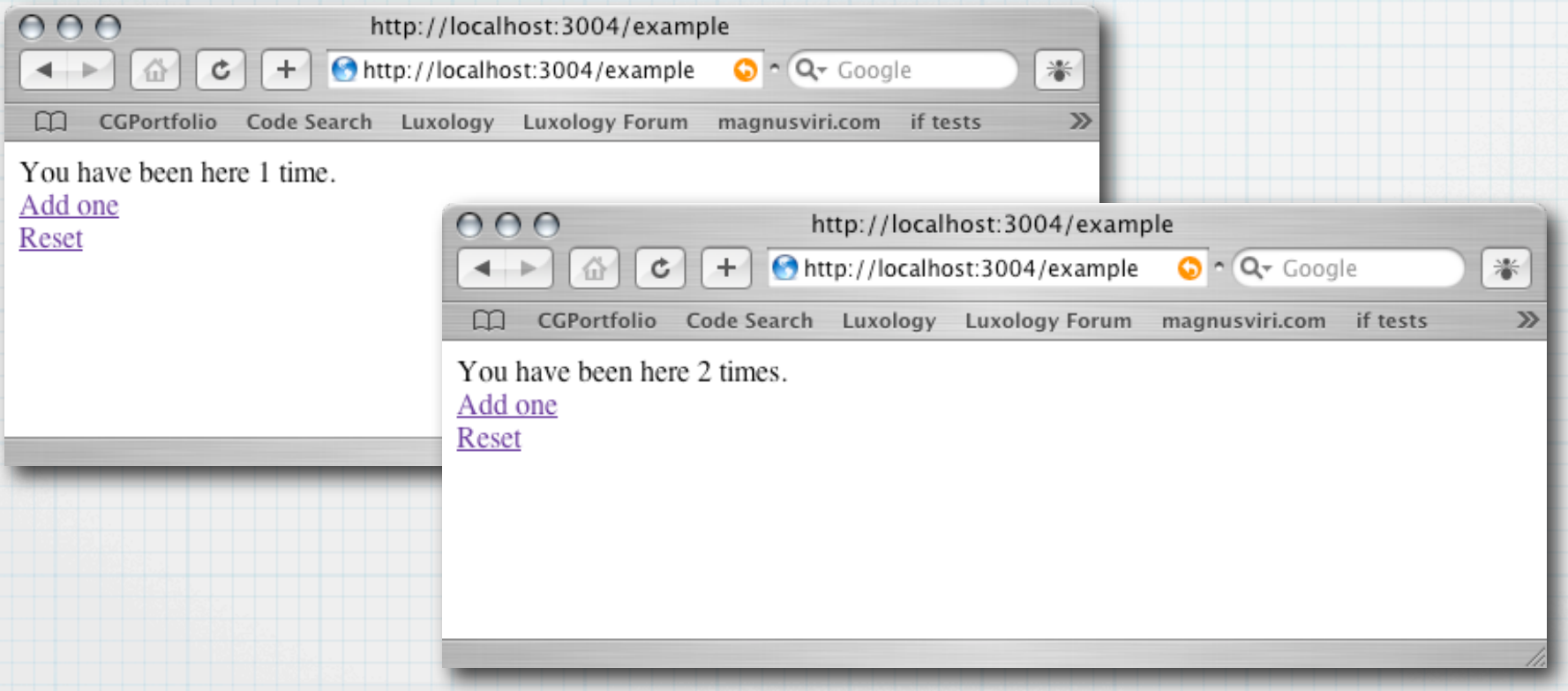
Make a Session

* Put this in index.rhtml

```
You have been here <%= session[:counter];
pluralize(session[:counter], "time") %>.
<br>
<%= link_to "Add one", :action=>:index %>
<br>
<%= link_to "Reset", :action=>:reset_counter %>
```

Make a Session

* Should look like this



Flash Message

* Put this in example_controller.rb

```
class ExampleController < ApplicationController
  def peek
    flash[:notice] = "A BOO!"
    redirect_to :action => :index
  end
end
```

Flash Message

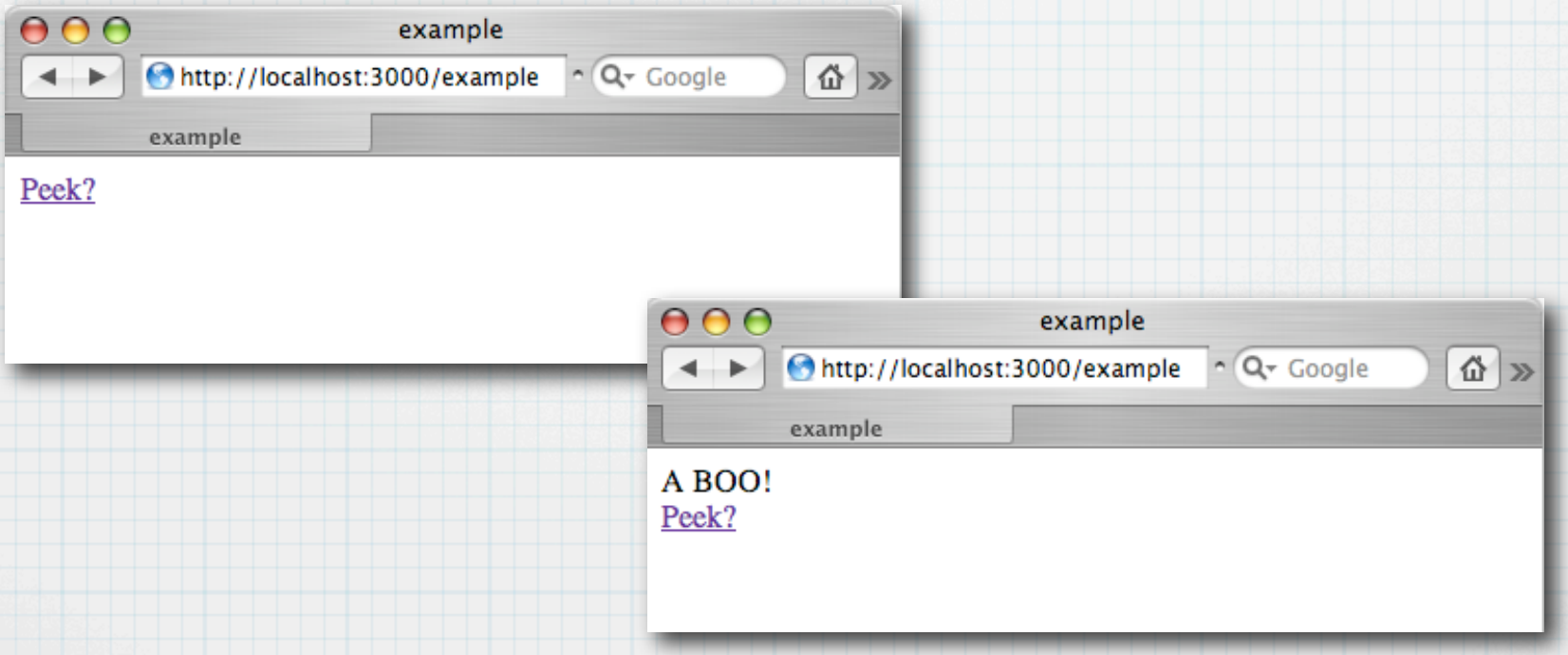
* Put this in index.rhtml

```
<% if flash[:notice] -%>
<div id="notice"><%= flash[:notice] %></div>
<% end -%>

<%= link_to "Peek?", :action => :peek %>
```

Flash Message

* Should look like this

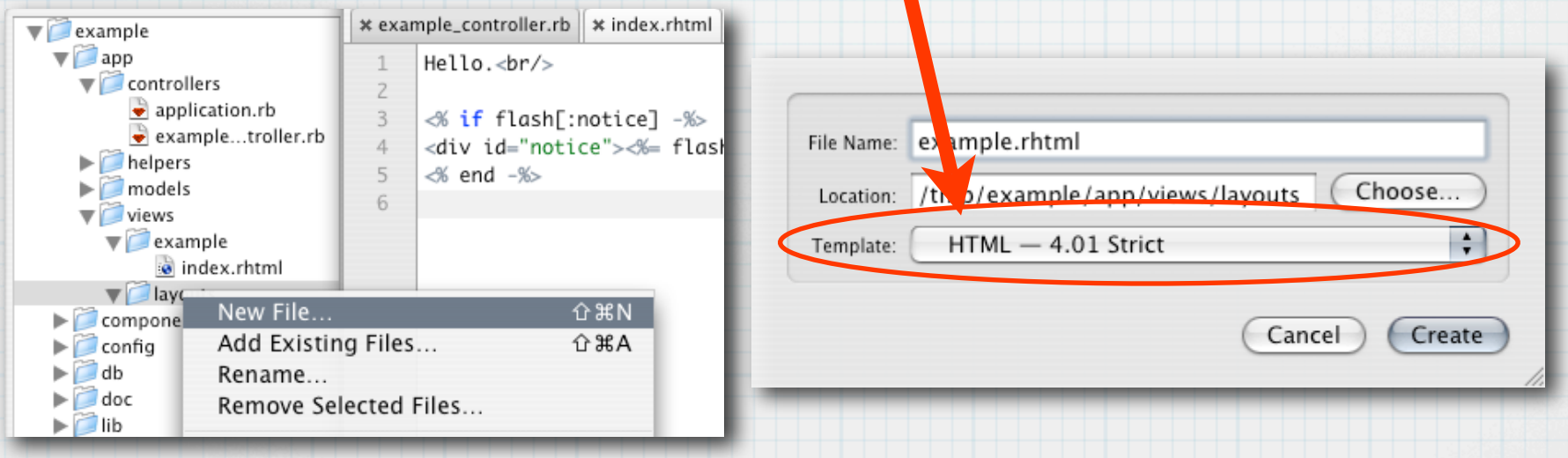


Use Layout

* Look at web browser source code
* No <head> or <body> tags

Use Layout

* Create new file "example.rhtml" in "layouts"
* Be sure to use HTML template



Use Layout

* Cut the flash code out of index.rhtml
* This is all that is left:

```
<%= link_to "Peek?", :action => :peek %>
```

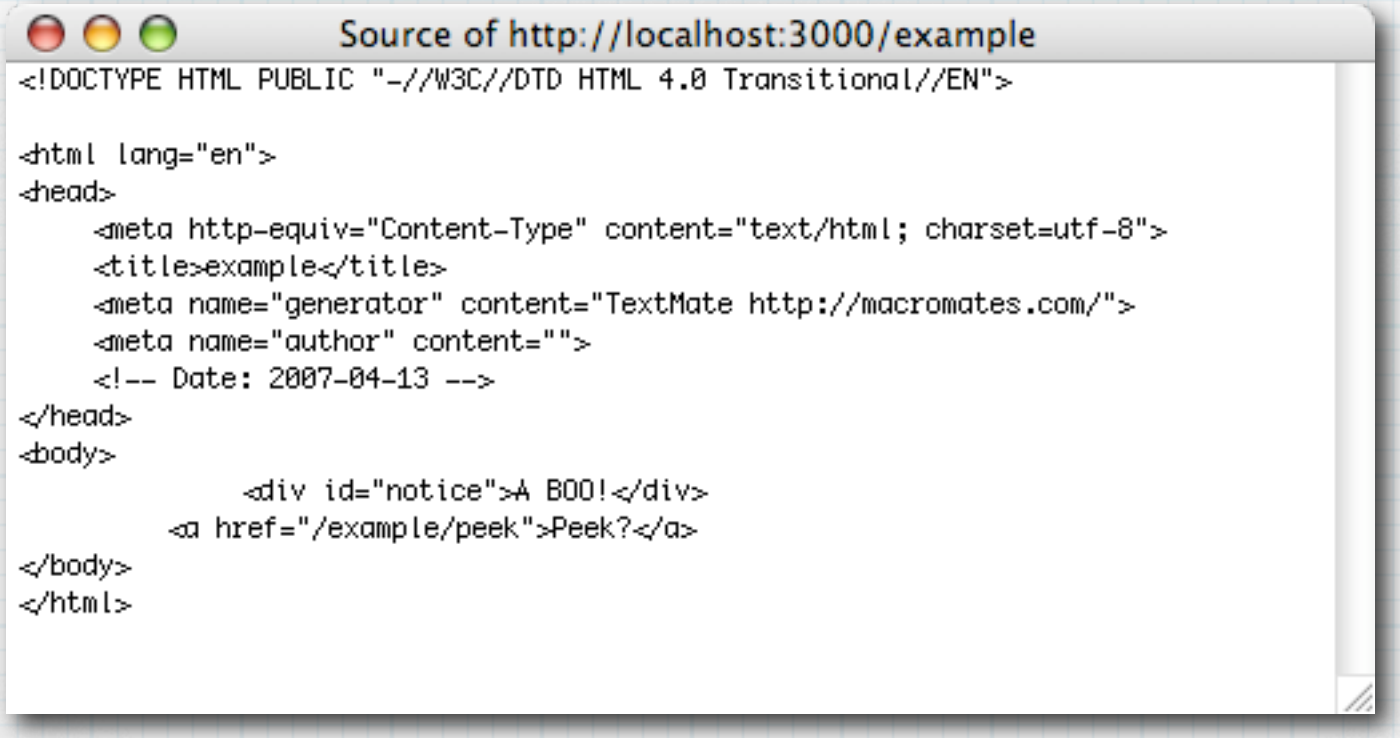
Use Layout

* Add code inside of example.rhtml's <body>
* Yield must have <%= !!!
* The - in -%> will remove the next \n

```
...
</head>
<body>
    <% if flash[:notice] -%>
        <div id="notice"><%= flash[:notice] %></div>
    <% end -%>
    <%= yield %>
</body>
</html>
```

Use Layout

* Should behave the same, except the source code!



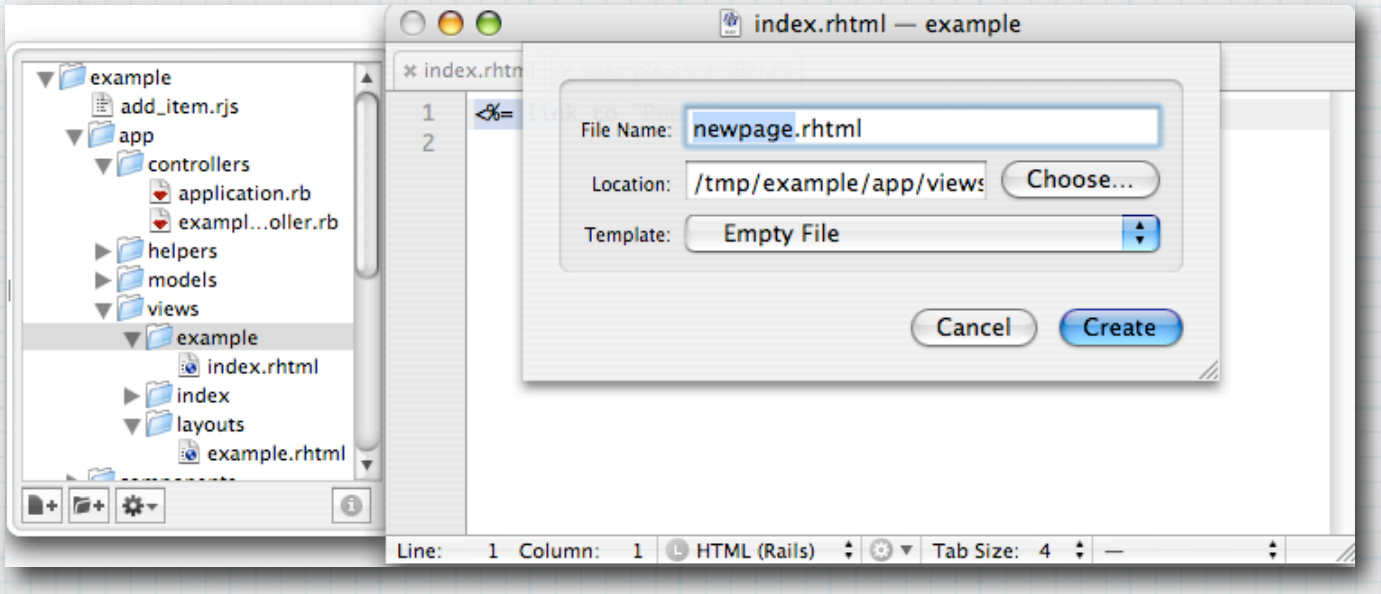
Use Layout

* ALL pages will show flash[:notice]
* Verify by adding a new page and change

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

To:

* `redirect_to :action => :newpage`



Save Form w/ Session

* Put in example_controller.rb

```
class ExampleController < ApplicationController
  def index
    session[:comment_list] ||= [ "Original item." ]
  end
  def add_item
    session[:comment_list].push params[:newitem]
    redirect_to( :action => :index )
  end
end
```

Save Form w/ Session

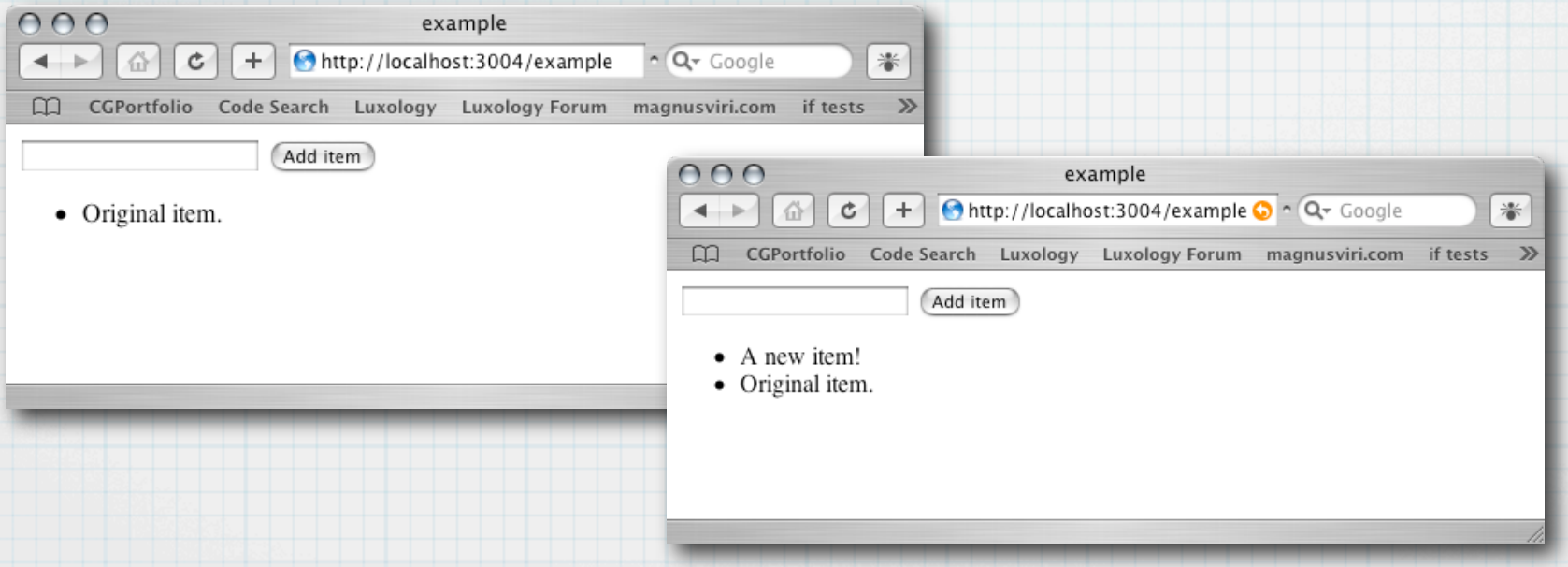
* Put in index.rhtml

```
<% form_tag( :action => :add_item ) do %>
    <%= text_field_tag :newitem %>
    <%= submit_tag "Add item" %>
<% end %>

<ul id="my_list">
    <% session[:comment_list].reverse.each do |line| -%>
      <li><%= line -%></li>
    <% end -%>
</ul>
```

Save Form w/ Session

* Should look like this
  * If it isn't, make sure your <%= is not <%



Convert to AJAX

* Change example_controller.rb

```
class ExampleController < ApplicationController
  def index
    session[:comment_list] ||= [ "Original item." ]
  end
  def add_item
    session[:comment_list].push params[:newitem]
    render_text "<li>#{params[:newitem]}</li>"
  end
end
```

Convert to AJAX

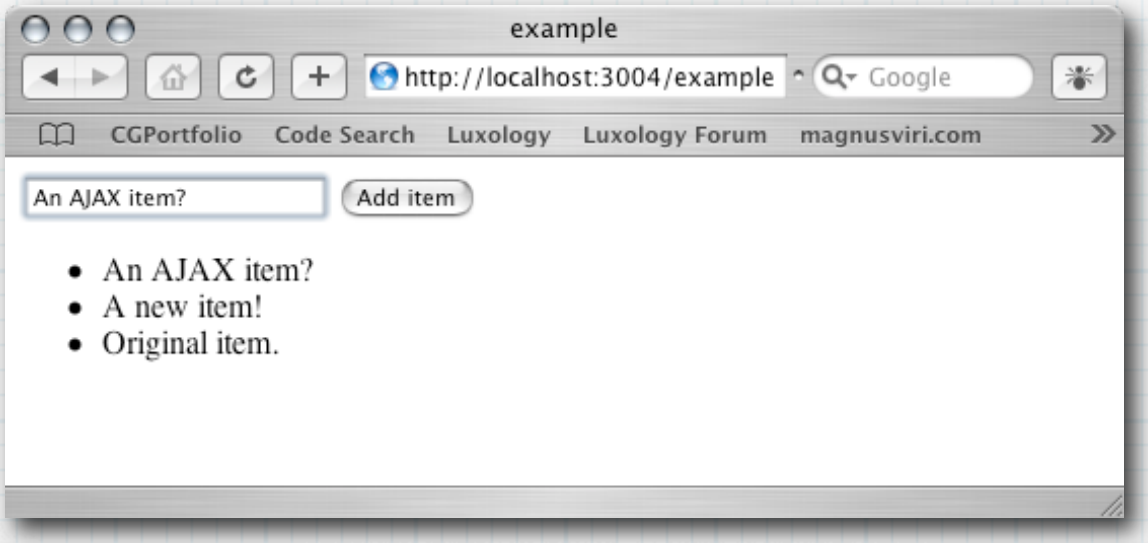
* Change index.rhtml

```
<%= javascript_include_tag :defaults %>
<% form_remote_tag( :update => "my_list", :url => { :action =>
 :add_item }, :position => "top" ) do %>
    <%= text_field_tag :newitem %>
    <%= submit_tag "Add item" %>
<% end %>

<ul id="my_list">
    <% session[:comment_list].reverse.each do |line| -%>
      <li><%= line -%></li>
    <% end -%>
</ul>
```

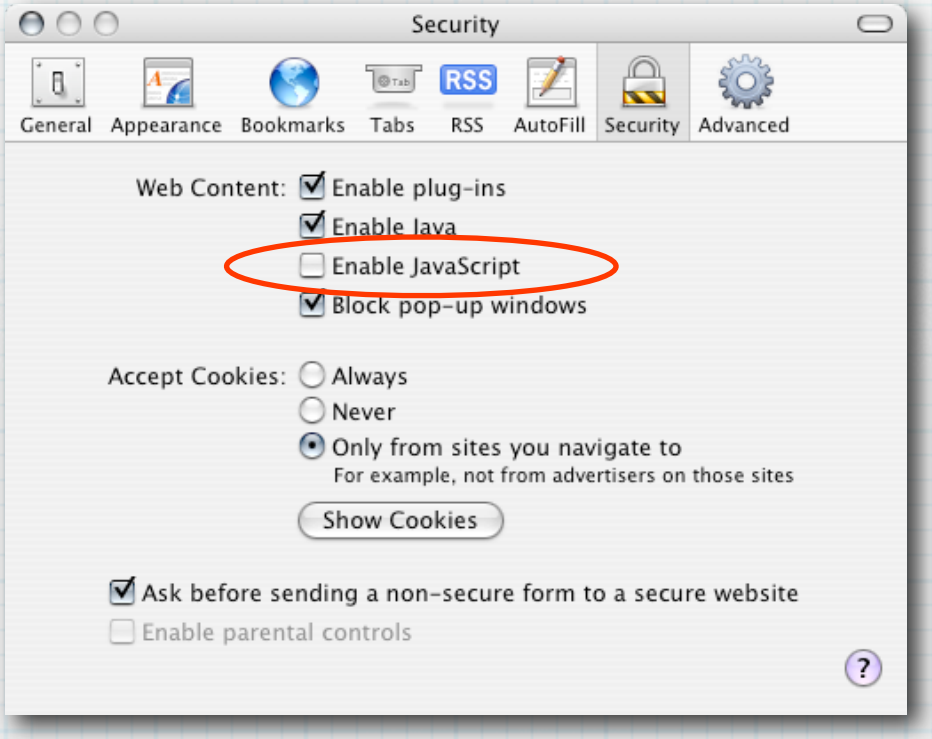
Convert to AJAX

* How do you know it is AJAX?
  * The text field didn't go blank!



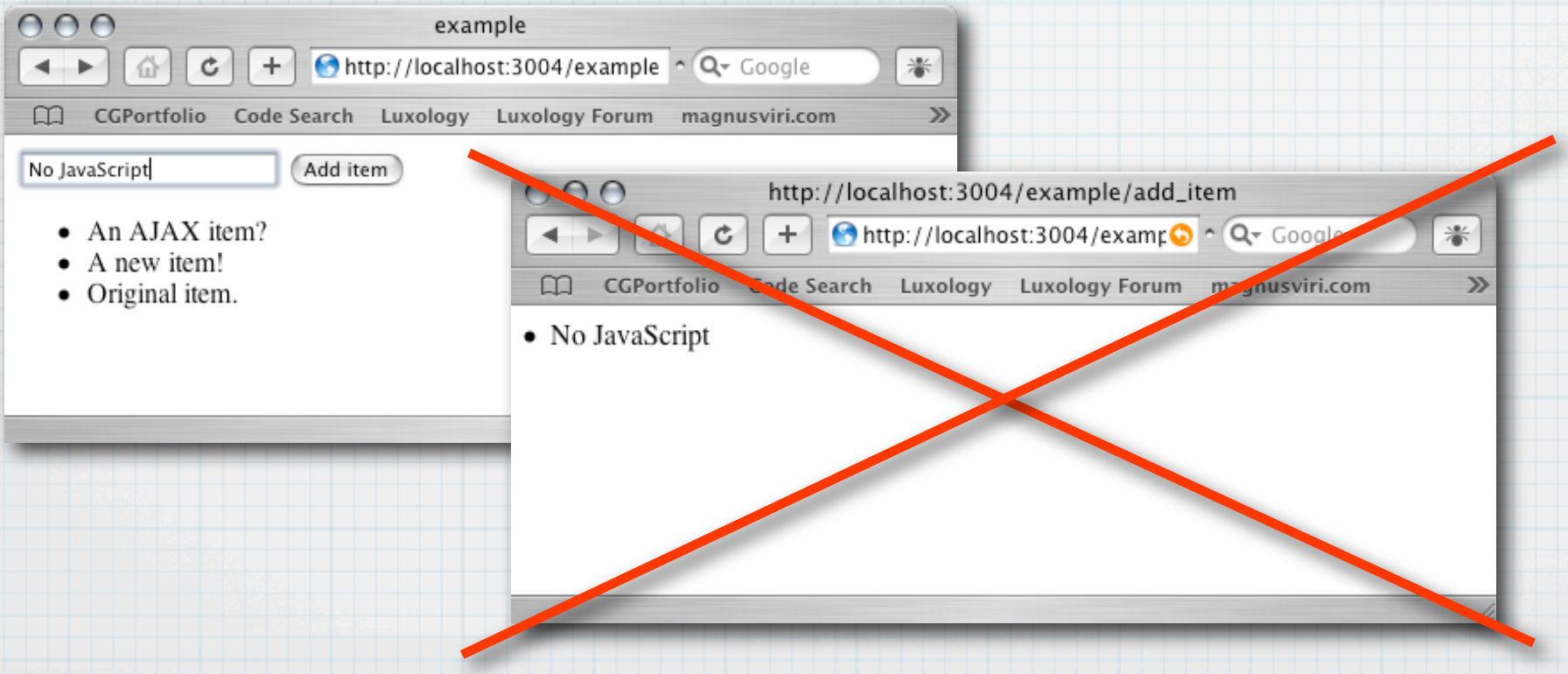
No JavaScript

* What happens if JavaScript is off?



No JavaScript

* That is unacceptable!



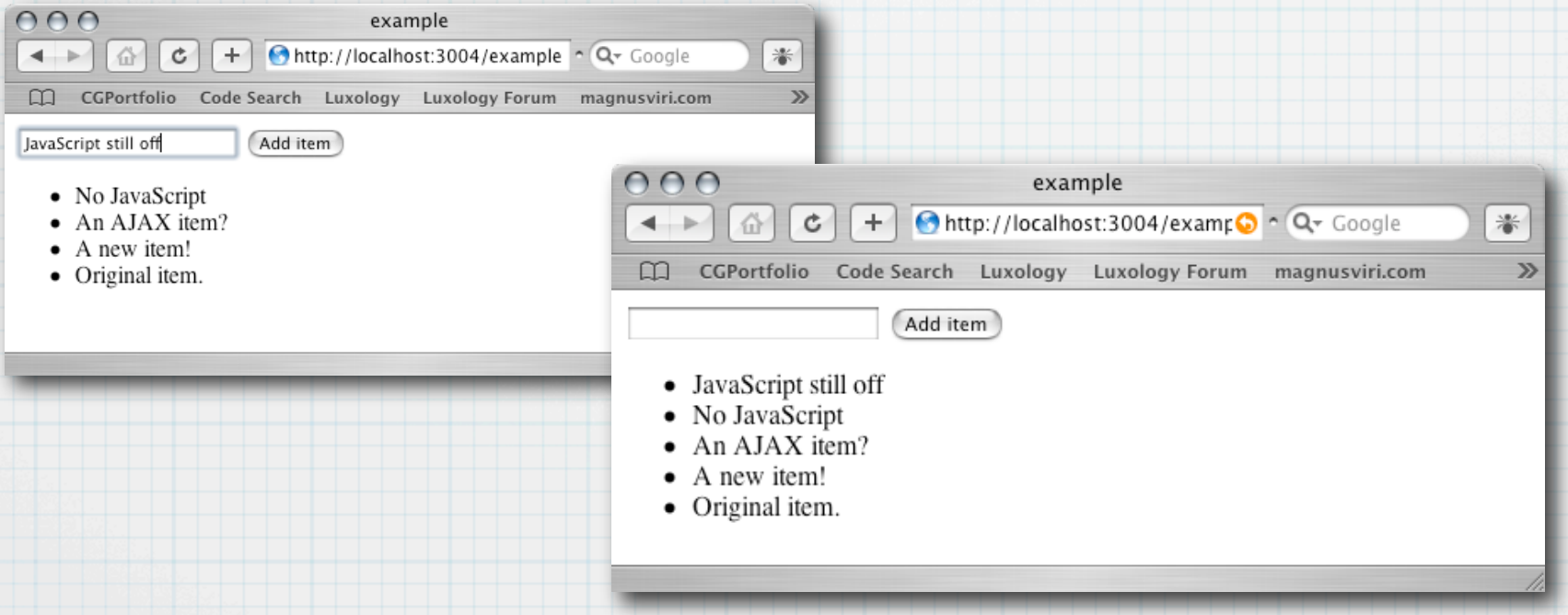
No JavaScript

* Change example_controller.rb

```
class ExampleController < ApplicationController
  def index
    session[:comment_list] ||= [ "Original item." ]
  end
  def add_item
    session[:comment_list].push params[:newitem]
    if request.xhr?
      render_text "<li>#{params[:newitem]}</li>"
    else
      redirect_to( :action => :index )
    end
  end
end
```

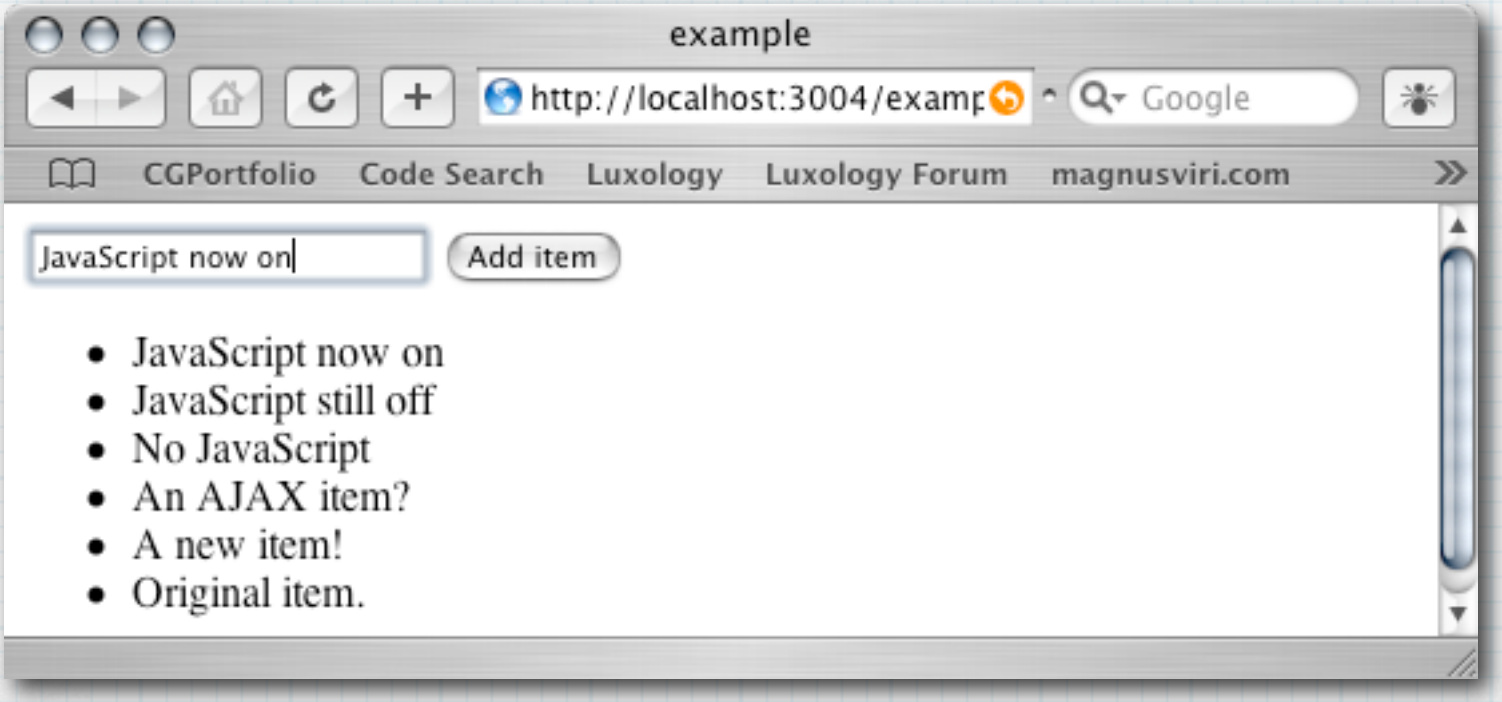
No JavaScript

* JavaScript off now works!



No JavaScript

* JavaScript on still uses AJAX!



Use RJS file

* Change example_controller.rb

```
class ExampleController < ApplicationController
  def index
    session[:comment_list] ||= [ "Original item." ]
  end
  def add_item
    session[:comment_list].push params[:newitem]
    redirect_to( :action => :index ) unless request.xhr?
  end
end
```

Use RJS file

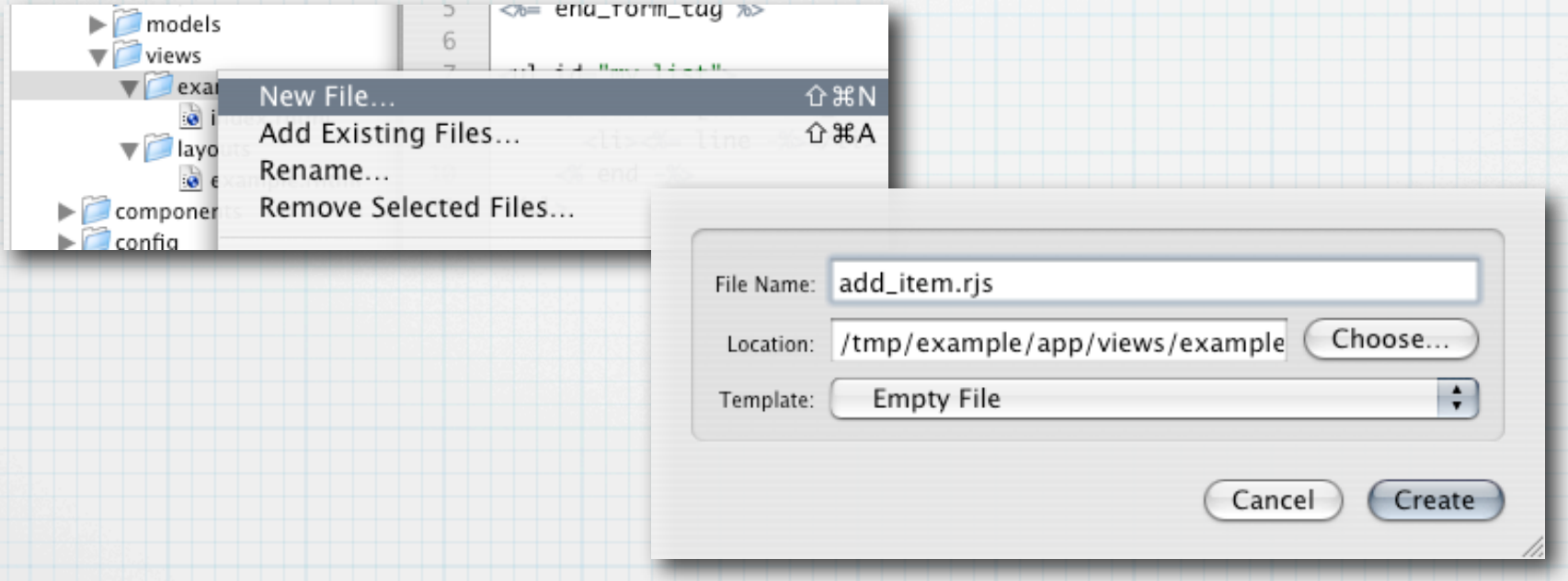
* Change index.rhtml

```
<%= javascript_include_tag :defaults %>
<% form_remote_tag( :update => "my_list", :url => { :action =>
 :add_item }, :position => "top" do ) %>
    <%= text_field_tag :newitem %>
    <%= submit_tag "Add item" %>
<% end %>

<ul id="my_list">
    <% session[:comment_list].reverse.each do |line| -%>
      <li><%= line -%></li>
    <% end -%>
</ul>
```

Use RJS file

* Create new file "add_item.rjs"



Use RJS file

* Put in add_item.rjs
  * This file is RUBY code, not JavaScript
  * It is converted to JavaScript by Rails

```
page.insert_html :top, "my_list", "<li>#{params[:newitem]}</li>"
```

Use RJS file

* Should behave exactly the same
* Well, why did we do that??
* Because we want "Magic"!
* To get the "Magic", all 's need to be numbered.

Add Magic

* Add id's to each item
  * Change index.rhtml

```
<%= javascript_include_tag :defaults %>
<% form_remote_tag( :url => { :action => :add_item }, :position =>
"top" ) do %>
    <%= text_field_tag :newitem %>
    <%= submit_tag "Add item" %>
<% end %>
<ul id="my_list">
  <% session[:comment_list].reverse.each_with_index do |line, index| -%>
    <li id='<%= session[:comment_list].length-index -%>'><%=line%></li>
  <% end -%>
</ul>
```

Add Magic

* Add id's to each item
  * Change add_item.rjs
  * As one line (no returns)

```
page.insert_html :top, "my_list", "<li
id='#{session[:comment_list].length}'>
#{params[:newitem]}</li>"
```

Add Magic

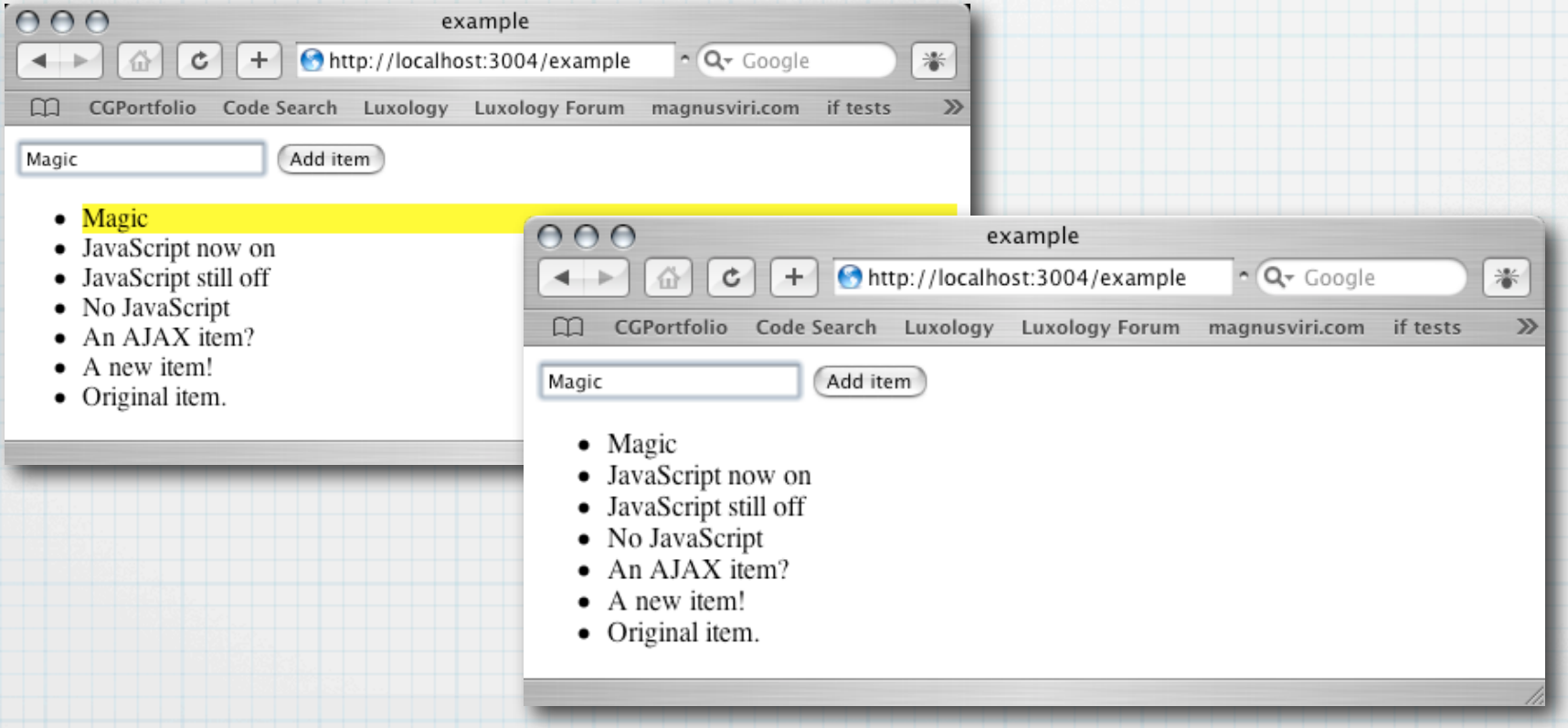
* Add a line to add_item.rjs
  * ALL AS ONE LINE (no return before :highlight)

```
page.insert_html :top, "my_list", "<li id='#{session
[:comment_list].length}'>#{params[:newitem]}</li>"
```

```
page[session[:comment_list].length.to_s].visual_effect
 :highlight, :startcolor => "#ffff00", :endcolor =>
"#ffffff"
```

Add Magic

* What have we got? Highlighting!



Done!

* Next class:
  * Connecting to a database
  * Using migrations
  * Scaffold
  * Model relationships
  * Using before_filter