

Rails 2

New features for your applications

by Ryan Daigle



CONTENTS

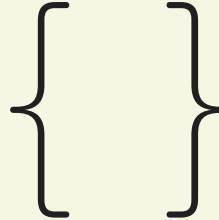
- 4 **ActiveRecord**
 - 4 Validations
 - 6 Query Caching
 - 9 Sexy Migrations
- 11 **ActionController & ActionView**
 - 11 Asset Servers
 - 14 Asset Caching
 - 17 Default Cookie-based Sessions
 - 18 Partial Layouts
 - 22 RESTful Routing Updates
 - 25 rescue_from Exception Handlers
 - 27 HTTP Authentication
 - 29 Convenient Access to Helpers
- 31 **Miscellaneous**
 - 31 De-Cruft Your Environment File
 - 32 Collection Fixtures
 - 33 Rake Tasks
 - 36 script/generate Update
 - 37 Debugging
- 42 **Deprecated Features**
 - 42 Deprecations from Previous Versions
 - 45 Pagination Removed
 - 46 acts_as Plugins
 - 47 Database Adapters Removed
 - 48 smtp_settings
 - 49 Renamed Routes
- 50 **Changelogs**
 - 50 ActionMailer
 - 51 ActionPack
 - 99 ActiveRecord
 - 130 ActiveSupport
- 138 **Revisions**

THE QUERY CACHE

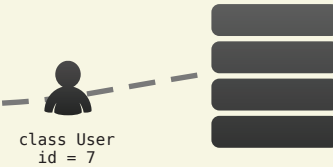
1 A query is created...

```
SELECT *  
FROM users  
WHERE id = 7
```

...the cache is empty...



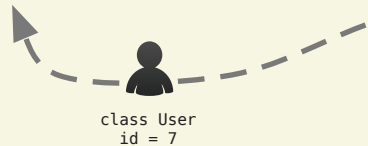
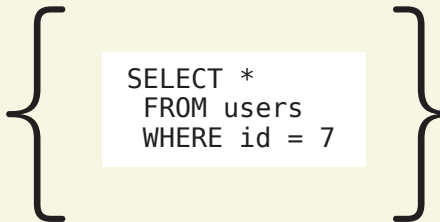
...so the database is searched and the record is cached.



2 Later during that request, the same query is created...

```
SELECT *  
FROM users  
WHERE id = 7
```

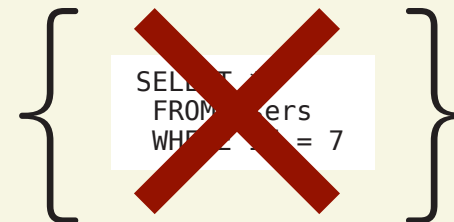
...it is found in the cache, so the User object is returned immediately.



3 Again during that same request, a record is updated...

```
UPDATE orders  
SET (amount = 100)  
WHERE id = 38273
```

...the entire cache is cleared.



Rendering this simple article partial produces this html:

```
<div id="article_1">
  <h2 id="article_1_title">
    Partial Layouts
  </h2>
  <div id="article_1_body">
    If you've ever taken the time to factor out the various
    commonalities that often exist in your Rails views into
    partials, you've probably noticed that there are still
    opportunities for DRYing up those partials. Well now you
    have a tool in your toolbox...
  </div>
</div>
```

However, when viewing an individual article it's appropriate to display more detail about the article. In that case you could use a more detailed article layout. Here's the `show.html.erb` view:

```
<%= render :partial => 'article',
          :layout => 'article_detailed',
          :locals => {:article => @post} %>
```

And the more detailed layout (`_article_detailed.html.erb`):

```
<div id="article_<%= article.id %>">
  <div class="ads"><%= get_ads %></div>
  <ul class="meta">
    <li>Posted by <%= article.author %></li>
    <li>on <%= article.published_at %></li>
  </ul>
  <%= yield %>
</div>
```

WHEN TO USE?

It's easy to get confused about how to use partial layouts as opposed to just several partials or even nested partials. I like to think of partial layouts as common view elements wrapped *around* the main partial content – they control the structure. Multiple or nested partials, on the other hand, just iterate through common view content. Ask yourself whether you have a common structure wrapped around your view bits or if you just have several common, independent bits.