Subject: Is Radicore better than Ruby On Rails?
Posted by AJM on Sun, 28 May 2006 12:46:23 GMT
View Forum Message <> Reply to Message

This is the question asked in http://www.tonymarston.net/php-mysql/radicore-vs-ror.html

Post your comments, opinons or views here...

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by Aviator on Wed, 21 Jun 2006 06:49:32 GMT
View Forum Message <> Reply to Message

Ruby on Rails is COOL. Everybody loves it.

The truth is, its website provides numerous examples of applications showing how powerful and flexible Rails framework can be (Tada-list, Backpack, wiki) it provides screencasts, and much more.

I have logged-in to your demonstration page, and it dissappointed me. Much of it is kinda rubbish, and I don't see why Radicore has capabilities of building a powerful web app.

I'm I a little biased?

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Wed, 21 Jun 2006 09:08:37 GMT
View Forum Message <> Reply to Message

Some people find that RoR is better than what they are used to, but it certainly is not the best there is! I do not like it for the reasons which are stated in my article. Do you have any response to those criticisms, something more than "Ruby on Rails is COOL"?

If you don't see why Radicore has capabilities of building a powerful web application then you obviously haven't read much. Radicore comes with the following built in:
- a dynamic menu system
- a role based access control system
- an audit logging system
- an activity based workflow engine

What does RoR have in comparison?

Also, Radicore is a true RAD environment based on transaction patterns which make use of the following:
- all Views are provided as pre-built reusable XSL stylesheets
- all Controllers are provided as pre-built and reusable scripts
- all Models are custom made, but the initial code is automatically generated from the Data

Dictionary. This takes cae of any data validation and communication with the database. All the programmer has to do is write the code for all the business rules.

This means that it is possible to create working transactions without writing a single line of HTML or SQL.

Does RoR have that amount of reusable code?

When you say "Much of it (my demo application) is kinda rubbish" what exactly do you mean? Can you be more specific?

Do I think you are biased? Most definitely.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by martin on Wed, 21 Jun 2006 09:11:41 GMT
View Forum Message <> Reply to Message

Hi Aviator,

It would be easier to discuss if you provided some information about what you are dissappointed about in the Radicore framework or more specifically the demo application. Also, what is it that you think is Rubbish in the demo application?

Personally, I see the following Radicore features as superior to other frameworks when it comes to building an administrative web application to be used on a daily basis behind the firewall in a company. The features also applies when building, for example, a backend application that handles all the contents of a public web application:

User Transaction Templates: One of the major features of this framework is the user transaction/web page templates that the programmer can choose from instead of starting from scratch for each page. The developer chooses a user transaction that he or she whishes to implement, such as view employee data, and then finds a suitable template for that user transaction. The framework comes with 25+ templates that can be mixed together
in an endless number of combinations to create the desired
web page flow...

Web Page Flow Context Management: One of the major hurdles when developing web applications is how to manage the context when navigating between web pages. This is completely automated in Radicore and transparent to the developer. If a developer for example wants to list employees and then view one or more of those, appropriate transaction/web page templates can be combined to achieve that without any session data programming.

Role-based Access Control (RBAC): Radicore comes with a fully integrated role-based access control system. Access to any task (i.e. menu, page, operation) in the system can be controlled dynamically by assigning Task Access rights and Field Access rights to roles. A user is then assigned a role that control the access to the application for that user and menus will also contain only the accessible menu items...

Dynamic Menu System: The menu system and the navigation system in Radicore are build on top of the RBAC module. When a user logs in the menu will automatically be build to show only those menu items that the user has access to depending on his or her role.

RBAC Management: A web application to administer the access control comes with the framework. You can also integrate the pages of this application with your own security management application as all components build with Radicore are re-usable.

Audit Logging Management: Audit logging is becoming more and more important these days with the Sarbanes-Oxly act etc. Radicore is build with this in mind from the ground up. All data in the system are by default auditable and the framework comes with an Audit Log Management application where audit logs can be inspected.

Internationalization (I18N): When a new web application is developed with Radicore it is automatically supporting internationalization. All text in the new application is contained in property files and all standard texts for buttons and messages are contained in property files. The UI is generated via an XSLT process and the standard stylesheets for different web page templates are unaware of any specific entity data. All application specific texts are fed into the XSLT process at runtime. So it is quite easy to support I18N.

Integrated Help System: Every task in the system is described in the database and context sensitive help links are
available in the UI on each page to take you to the help page
for current task.

Print Preview: Each web page in a Radicore application can be converted into a print preview page by one simple click on a link at the top of the screen. No extra programming.

Validation: Primary validation is build in to the framework and it will be executed automatically by looking at the field specifications to determine if an error message need to be displayed. Secondary validation is also supported via custom programming to for example check two fields against each other.

Themes/Styles: The applications build with Radicore can be made to look just the way you want them too by modifying the CSS files that comes with the framework.

Code Generator: To develop with Radicore is already extremely fast, but to speed it up even more it comes with a code generator in combination with a data dictionary. With the code generator you connect to the database and read all metadata for tables you which to create web pages for. After that you work in steps to configure field specifications, setup table relationships etc and in the final step you generate code, that is ready to run.

I think these are some really nice features I have not come across in any other framework.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by Aviator on Thu, 22 Jun 2006 05:42:50 GMT
View Forum Message <> Reply to Message

Thanks for the quick reply, guys.

No, it's not the framework I'm dissappointed with. But it's how you failed to tell anyone that Radicore is much better than Ruby on Rails. I've mastered quite a bit of PHP myself, but I'm really impressed by the potentials of RoR. It's codes are very, very simple, and by the product that comes from it (Tada List, Backpack, Campfire, forum system, wiki) I can tell that it succeeded in telling everyone how COOL the framework is.

The demostration on your site seems more like an online administration system than a web application. Every page in it is presented in lists! It does not convince me of Radicore's capabilities. I'm sorry to say this, but it's pretty dull, and I could not understand it's motive. I think you should provide some more web applications made with Radicore, some code samples or maybe some screencasts to prove Radicore's capabilities to the audience.

Or maybe it's already intended to make an online administration system? If that's the case, than excuse me for being dissappointed in Radicore. I'm just looking for a powerful PHP framework that could replace RoR in terms of simplicity, or maybe 'coolness'. OK, forget the word 'cool'.

By the way, I think they can make some RoR plugin which provide all of the features included in Radicore.

Don't get me wrong, I still love PHP anyway.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Thu, 22 Jun 2006 08:56:41 GMT
View Forum Message <> Reply to Message

If you read the description of Radicore on http://www.radicore.org/ it most definitely says that it is designed for administrative web applications which have restricted user access instead of the more common web sites which are open to anyone. This is more like a traditional desktop application, but converted to a web application.

Radicore may act as the "back office" to administer the contents of a web site, but it is not designed to create the "front end" which is open to anyone. There is a subtle difference.

Programmers who build nothing but web sites often have great difficulty in building more complex administrative applications, but as I have been building such applications for more than 20 years I have a great deal of experience to call on. In fact the Radicore framework is the direct descendant of a COBOL framework I designed and wrote in 1985.

Remember also that the "RAD" in Radicore stands for Rapid Application Development, so the aim of the game is to allow programmers to create working transactions as quickly as possible. This means making use of a powerful framework which contains as much pre-written and reusable code as possible, and to make as many features as possible, such as pagination, column sorting, navigation, etc, available with as little effort as possible.

Radicore is aimed at a particular market, and if you are not in that market then Radicore is not for you.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by martin on Fri, 23 Jun 2006 16:54:11 GMT
View Forum Message <> Reply to Message

Hi all,

I have done an implementation of the Ruby on Rails Cookbook example to better give us a way to understand the differences between these frameworks.

My article can be found here: http://www.marversolutions.com/examples/cookbook.html
The Rails article can be found here:
 http://www.onlamp.com/pub/a/onlamp/2005/01/20/rails.html?pag e=1.

This implementation is done with Radicore for Java but it will give the PHP reader a clear view of what the differences are between Ruby on Rails and Radicore when it comes to what features and functionality that are provided for free by respective framework.

I suggest reading the Ruby on Rails article first and then my article.

Cheers,
Martin

PS. The Radicore for Java product is scheduled for public release in July-2006.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Fri, 23 Jun 2006 17:24:46 GMT
View Forum Message <> Reply to Message

Nice one, Martin. Best of luck with your up-coming release.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by darkmatter on Wed, 29 Nov 2006 23:19:24 GMT
View Forum Message <> Reply to Message

> 1. Table names must be plural

Not true. This is the default behavior, but you can override it (like almost *anything* in Rails):

```
class MyModel
  set_table_name "my_model"
end
```

> 2. Primary keys must be auto_increment and named 'id'

Not true. Again, this is just the (overrideable) default:

```
class MyModel
  set_primary_key "ModelId"
end
```

As for the "auto_increment" part, you are free to manage your
primary keys as you want, if you really want. There are even
plugins to handle composite primary keys.

> 3. Candidate keys must be ... ?

This is not very clear. What do you need candidate keys for when
you have a primary key? Maybe you should explain yourself better.

> 4. Relationships are inferred from foreign key names

Not true. Again, you seem to have a hard time distinguishing
the (very reasonable) Rails defaults from hard-coded conventions.

```
class MyModel
  has_may :other_models, :foreign_key => "an_elephant"
end
```

> 5. Intersection tables in Many-to-Many relationships

"...RoR cannot recognise intersection tables unless they are named according to the rule
alphabeticallyFirstTablePlural_alphabeticallySecondTablePlur al. "

Not true. Same old story.

```
class MyModel
  has_and_belongs_to_many :elephants, :join_table => "pink_elephants_for_my_model"
end
```

> 6. Primary data validation

True, but nothing prevents you from writing a plugin that adds the necessary validation methods
inferring them from the database schema. But the general philosophy behind Rails is that most of
the business logic should be *visible* in the code. And anyway,
if you try e.g. to nullify a not-nullable column, you get a nice
exception from the database layer, which thankfully is very

easily handled in Ruby (unlike, say, in PHP4).

I would suggest that you try spending more than 1 minute on the Rails documentation and make some more amends to your document,
which at the moment relies almost entirely on wrong assumptions.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Thu, 30 Nov 2006 00:04:14 GMT
View Forum Message <> Reply to Message

All those comments I made about RoR were based on what I read on the RoR web site. If my comments are incorrect then the documentation is incorrect .

You are missing the point of my arguments. Even though these conventions in RoR can be overridden it means that RoR cannot be used "out of the box" with databases that do not follow these conventions. Before it will work people have to know the following:

that their database does not follow the RoR conventions.
what these conventions are
how and where to write the code which forces RoR to deal with these "unconventional" names.

Radicore, on the other hand, does not have any restrictive conventions, so will work "out of the box" with any database design. Just import the database schema into the data dictionary and off you go.

You also do not understand candidate keys. These are unique keys which may exist IN ADDITION to the primary key. The normal rule is that primary keys cannot be changed whereas candidate keys can. The Radicore framework knows when a candidate key is being changed, so it can verify that the new value is still unique. This does not require any custom code or any plugin, the necessary code is just there. Are you saying that RoR cannot deal with candidate keys? If so that is a glaring omission.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by iamaconvert on Wed, 17 Jan 2007 02:04:54 GMT
View Forum Message <> Reply to Message

Man, I can't help but agree with darkmatter and aviator.

I feel the demo/tutorial was a turn off as well.

Maybe I am a command-line person. I feel I have more control over RoR and Cake. I have to be honest, I haven't really go in depth with Radicore.

But if people get turn off right from the start, boy, something just not right. Be it the documentation, demo, framework etc.

I like the concept of RoR, yes, probably they have bind certain stuff which is not quite relevant in database theory but the whole idea is to make web application much more simple.

When I use radicore, I feel like I have just take a step back to last millennium. I think a great improvement for radicore is to dump the toolkit GUI (Redesign)...thats for last millennium. Really messy, too many things and not goal driven in the process. I believe that was the turn off. Oh, the listing of "everything", just give me the headaches.

Have a screencast? Seeing is believing.

Sorry guys, I'm not impressed by radicore, is just not easier (How can it be rapid when it is not easy to use?). I think I have to give it a pass.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Wed, 17 Jan 2007 09:30:45 GMT
View Forum Message <> Reply to Message

For people who want to develop administrative web applications instead of web sites (and if you don't know the difference then Radicore is NOT for you!) then the Radicore toolkit is a step in the right direction.

A web application is about using forms to get data into and out of a database, and Radicore provides the means to build those forms very quickly. It also provides such useful facilities as a dynamic menu system, role based access control, audit logging, workflow and internationalisation right "out of the box". If you do not have a need for any of those facilities then you are not writing a web application, and so you are wasting your time by looking at Radicore.

If you don't like the look of the interface then you have the opportunity to change the CSS files and produce whatever effect you want.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by iamaconvert on Wed, 17 Jan 2007 17:24:35 GMT
View Forum Message <> Reply to Message

Hmm...why would I need a framework for websites...

1. You need to read without being bias.

2. Stop assuming.

3. Relax. Take some feedbacks, go the extra mile to make the development better. You're the one that ask the public for comments and opinions. I gave you mine.

4. web applications is web applications...ACL, audit logging, workflow, i18n are all part of the project requirements if it is needed. ACL in RoR is pretty easy, maybe you should read up on it. Doesn't hurt, just like I read radicore and I have different opinions. Too bad if it doesn't pleases you.

5. Yup I probably could just change the css but you missing my point. Redesigning means reevaluate its hci, leads to better usability and might just improve overall development for the toolkit.

6. I have always believe no matter how large your web application maybe, you have to take in consideration from all sides. From framework, constraints, requirements, security, hci, design...everything plays apart.

Thanks for your sarcastic remarks.

You are right, I am wasting my time by looking at Radicore.

Subject: Re: Is Radicore better than Ruby On Rails?
Posted by AJM on Wed, 17 Jan 2007 18:30:38 GMT
View Forum Message <> Reply to Message

Perhaps if you gave more constructive criticism I would be abe to provide a better response. Statements like "I feel I have more control over RoR and Cake" leads me to believe that you like to do more things yourself rather than have the framework do them for you. This "DIY" approach means that you actually spend more time designing and writing code whereas the Radicore approach is to make use of as many pre-written and reusable components as possible, thus cutting down the development effort without cutting down on the features. The only problem arises if you don't like the way these features are implemented, but I cannot do anything about that. You either like it or you don't, and unless you can describe a better method you simply aren't saying anything worth listening to.

You suggest I should redesign the GUI without identifying how it could be improved. Statements like "I feel like I have just taken a step back to last millennium" do not contain anything of substance and are therefore worthless. Perhaps the fact that you say "I am a command-line person" means that you feel more at home with a command line interface instead of a GUI, in which case you are totally out of luck.

You say "it's really messy, too many things" without quantifying what exactly you mean. How is it messy? What could be removed where to clean it up?

When you say that Radicore is "not goal driven in the process" that just tells me that you have totally missed the point. The purpose of Radicore is to build transactions that maintain the contents of database tables, and these transactions can be built very quickly from the library of transaction patterns. These patterns include lots of standard behaviour that is instantly available without any further effort.

"The listing of 'everything', just give me the headaches." indicates to me that you have not worked on an administrative application, web or otherwise. I have been working with such applications for 25+ years and this is how they all work. Take a typical scenario in an accounting system for example. You want to look at a customer's details, so where do you start? Answer: at the 'List Customer' screen. Rather than having to scroll through all the records until you find the one you want you press the 'Search' button so you can enter whatever search criteria you have available. When you press the 'Submit' button on the 'Search' screen it instantly returns to the 'List' screen and rebuilds the display according to your search criteria. Having located the record you want you then have a variety of navigation buttons at your disposal to perform other actions on the selected customer. This could be viewing that customer's invoices, which again is presented as another list. The contents of this list can be refined with another 'Search' button, such as "overdue invoices only", or you can use any of the available hyperlinks to change the sort order, such as by value instead of date.

If this is not the type of application you are writing, then Radicore is not for you. If this is not how you want your application to function, then Radicore is not for you.

But if this *IS* the type of application you want, and *IS* the type of behaviour that you want, then you will have to go a long way to find anything better than Radicore.