Subject: Tutorial
Posted by DannyBoyPoker on Mon, 08 Apr 2013 03:56:24 GMT
View Forum Message <> Reply to Message

Tony,
This is to document a few issues with the tutorial being slightly out of date, and I think that the relationship between subsystem id, subsystem name, subsystem description, task name, and menu button text, as well as subsystem directory, might be scrutinized, as in, are these relationships correct (or maybe, strictly correct). Just some grist for the mill, from fresh eyes.

From here:
http://www.radicore.org/radicore-for-php.php?PHPSESSID=9f705944510313c4e0c1a94b6495e278

'The following database drivers are also included:

MySQL version 4.0.6 to 4.0.25, which uses the original MySQL extension. Note that MySQL stopped active support for version 4.0 on 30th September 2006.
MySQL version 4.1 and above, which uses the improved MySQL extension. Note that MySQL stopped active support for version 4.1 on 31st December 2006.
PostgreSQL version 8 and above.
Oracle 10g
SQL Server 2008 R2'

This might be read, as offering that drivers are distributed with Radicore, which is not true? I think it's only intended to mean that Radicore has been *tested* with these drivers.

In which case, --I'm engaging with the tutorial, to be found here:
http://www.tonymarston.net/php-mysql/radicore-tutorial.html

And, this is an example of eight CREATE TABLE queries,
CREATE TABLE IF NOT EXISTS `x_option` (
[etc.]
) TYPE=MyISAM;

Now, normally, it is unnecessary to use ENGINE to specify the MyISAM storage engine. Also, the older term TYPE used to be supported as a synonym for ENGINE only for backward compatibility, though ENGINE has been the preferred term and TYPE was deprecated. And now, the keyword TYPE is removed since MySQL 5.1. Instead, use (for example):

) ENGINE = MYISAM;

But also, the update has already been incorporated, here:
'The database schema can also be found at radicore/xample/sql/mysql/xample-schema.sql.' So, it's only the schema as given on that page, at
http://www.tonymarston.net/php-mysql/radicore-tutorial.html, which is out of date.

Also on this page, is this:
'1.1 Create Subsystem

Log on to the framework and navigate to [etc.]'

Not given here, is how to log on to the framework. Which, I suppose, is given in the installation instructions..

Also, going through this 1.1. Create Subsystem, and 1.2 Build Directory, there is this:
'..press the Build Directory button in the navigation bar..This will cause the following actions to be performed:..Create a task of type MENU on the MNU_TASK table with the same name as the subsystem id. This will act as the initial menu for this subsystem.'

Here, first, I'll note that the MNU_TASK table can be found in the menu database.

And then also, I read here that a task of type MENU is created, with the same name as the subsystem id. The subsystem id is 'TEST'. But, the name of the task is 'test'. I'm of course merely pointing to the case of the letters, but this is not the same name, TEST /= test. It even appears, that while the name of the task matches the name of the directory of the test subsystem, the ID does not match the name of that directory, and in fact is not supposed to match (there are two different fields on the form).
Which, is in contradiction to this:
'..press the Build Directory button in the navigation bar..This will cause the following actions to be performed:..Create the directory structure in the file system with the same name as the subsystem id.'

Also, '(T)his menu item', which is to say, a task of type MENU, with the same name as the subsystem id, is added automatically to the main menu. So, what is added there, is called 'TEST'. And so, the button text of the menu item that is added automatically to the main menu, does not match the name of the task, which means that it also does not match the directory of the subsystem. In our case,
they are similar, TEST /= test, but the name of the task can be totally unrelated to the subsystem id.

Then, for the 'Import Database' step, a Database ID and Subsystem is chosen, where the correct choice of subsystem is 'test subsystem'. This is not the subsystem id, but the subsystem description. In fact, also, when you 'Enquire Task', the subsystem is given as 'test subsystem', which is the description, not the id. Which is, I take it, incorrect. The id is the name of the subsystem, though one might, here, optionally also list the description. In addition, the tutorial says: "(C)hoose 'test' from the dropdown list." Which, is not an option. But also, 'test' is not the subsystem id anyways ('TEST').

And then, '(W)hen the 'Submit' button is pressed this function will create an entry for this database on the DICT_DATABASE table, and create entries on the DICT_TABLE table for every table which currently exists within that database.'

Okay, a note, that the DICT_TABLE table is in the 'dict' database. And, '(I)t will then automatically display the results as shown in Figure 7:' However, this figure does not exactly match. The actual

page, shows an upper row of buttons: 'Search', 'Import Tables', 'Child Relationships (DB)', 'Parent Relationships (DB)', and 'Output to PDF (L)'. And, there is a lower row of buttons: 'Read', 'Update', 'Delete', 'Export to PHP', 'Columns', 'Keys', 'Child Relationships (Table)', Parent Relationships (Table), 'Audit Trail', 'Audit Trail (exact)'. The figure shown, only gives a 'Relationships (DB)' button, not 'Child Relationships (DB)' and 'Parent Relationships (DB)'. Similarly, 'Relationships (Table)' has become 'Child Relationships (Table)' and 'Parent Relationships (Table)'.

In addition, the columns of the table, 'Select', 'Table Id', 'Description', and 'Columns'. Missing, here, in the figure, is 'Child Relationships', and 'Parent Relationships'. There are a few other reasonably trivial-looking differences, the figure is just a bit out-of-date, I guess. Of those other differences, I maybe do want to mention this one, that it's not true that '(T)he number in the 'columns' column will show the value zero if the import column procedure has not yet been run on any table.'

When I get to this: 'Relationships are not defined in the database (no, foreign key constraints are NOT the same as relationships), so they must be entered into the dictionary manually.'

I do have a question, as to whether it is/is not true, that foreign key constraints are NOT the same as relationships. In particular, are we in fact going in and setting up parent tables, child tables, parent keys, and child keys. Which, would be the sort of thing that foreign keys enforce.

Subject: Re: Tutorial
Posted by jjtoranzo2004 on Mon, 08 Apr 2013 12:57:06 GMT
View Forum Message <> Reply to Message

Danny,

Could you number each issue separately?
Thay way it would be easier to follow (and announce which has been fixed).

Thanks in advance.

Subject: Re: Tutorial
Posted by AJM on Sat, 20 Apr 2013 10:33:20 GMT
View Forum Message <> Reply to Message

You are just nitpicking over trivialities. Their may be cosmetic differences between the tutorial when it was written and the current version software, but the functionality is the same. However, I shall answer your points as best I can.

1) By "database drivers" I do not mean "database software" but "database connectors" in the form of classes from which I create Data Access Objects (DAOs). I have changed the text to reflect this.

2) I have changed the code samples to read "ENGINE=MyISAM" instead of "TYPE=MyISAM".

3) I have updated the instructions in section 1.1 to explain what "logon to the framework" means.

4) I dislike case sensitive software immensely, which is why I wrote Case Sensitive Software is Evil. Consequently I consider "TEST" to be the same as "test".

5) It should not be difficult to understand that the "subsystem with the id of 'test'" is exactly the same as the "subsystem with the description 'test subsystem'".

6) The fact that some of the navigation buttons on the screens has been updated since the screenshots were made does not invalidate the functionality which is being described.

7) You do not understand the difference between a "foreign key" and a "foreign key constraint". A one-to-many relationship between two tables requires that the "many" table contains a foreign key which links to the primary key on the "one" table. It is possible to define a foreign key without any constraints, which is exactly what I do. A great deal of my database development has been with databases that did not have foreign key constraints (such as the MyISAM engine in MySQL)y I don't use them but instead have the relevant functionality built into my framework.

Subject: Re: Tutorial
Posted by DannyBoyPoker on Sun, 21 Apr 2013 07:57:17 GMT
View Forum Message <> Reply to Message

AJM wrote on Sat, 20 April 2013 06:33You are just nitpicking over trivialities.
Some of putative anomalies that I listed were pretty trivial.

AJM 4) I dislike case sensitive software immensely, which is why I wrote Case Sensitive Software is Evil. Consequently I consider "TEST" to be the same as "test".

The status quo is that some things are case sensitive. In that sense, I don't have any problems with casing. I mean, I might sympathize--case sensitivity, shall we say, has got to go. But, the world happens to be case sensitive, and this just happens to be one of the many ways that programmers can generate run-time errors. Requiring exact case matching is much easier to support. One might range widely, in considering this as a practical issue--for example, case insensitive comparisons are definitely much harder to do when you are dealing with more than just the standard English character set. Casing is an arbitrary distinction, but we are dealing with machines.

AJM5) It should not be difficult to understand that the "subsystem with the id of 'test'" is exactly the same as the "subsystem with the description 'test subsystem'".

Well, my response is going to resemble my previous response--what is easy to understand, is a rather subjective category, even if we were not also dealing with machines.

AJM6) The fact that some of the navigation buttons on the screens has been updated since the screenshots were made does not invalidate the functionality which is being described.

This is good to know. And also I'm sorry if I came off abrasive, chalk it up to enthusiasm.

AJM7) You do not understand the difference between a "foreign key" and a "foreign key constraint". A one-to-many relationship between two tables requires that the "many" table contains a foreign key which links to the primary key on the "one" table. It is possible to define a foreign key without any constraints, which is exactly what I do. A great deal of my database development has been with databases that did not have foreign key constraints (such as the MyISAM engine in MySQL)y I don't use them but instead have the relevant functionality built into my framework.

I do not understand, that it is possible to define a foreign key without any constraints? I understand that MySQL tables do not enforce referential integrity, because some MySQL engines (MyISAM, NDB) do not enforce referential integrity. I guess that I might attempt a list of some of the advantages of foreign key enforcement, but on the other hand, you have the relevant functionality built into your framework. So, you are leveraging some types of constraints, and the key advantage of this, is to increase the integrity of the data.

Subject: Re: Tutorial
Posted by AJM on Sun, 21 Apr 2013 09:59:30 GMT
View Forum Message <> Reply to Message

DannyBoyPoker wrote on Sun, 21 April 2013 08:57
4) The status quo is that some things are case sensitive.
Fortunately only *SOME* operating systems and *SOME* programming languages are case sensitive. I have been in software development for over 30 years, and nothing which I encountered prior to Unix/Linux was case sensitive. If you think that case sensitivity is a good thing just image what chaos would be caused if email addresses and URLs were case sensitive.

DannyBoyPoker
7) I do not understand, that it is possible to define a foreign key without any constraints?
To me a relationship between two tables allows me to JOIN them in an sql SELECT statement. Thus I can use a column as a foreign key without having to define it as a foreign key.
