Subject: Business Rules Best Practice
Posted by stephenboey on Sat, 07 Apr 2007 06:59:13 GMT
View Forum Message <> Reply to Message

Hi Tony,

Business rules are usually complex and involves a lot of code.

What is the best practice to implement business rules in Radicore?

Php is usually a non ideal for running ad-hoc/daily batch processing jobs, due to the timeout factor. Is there any way to avoid using stored procedures and another batch program? How do we avoid such duplication of effort, which could result in batch program's business rules out of sync with rules in Radicore?

I have thought about Rules Engine like drools. Need to hear you out on this...

Subject: Re: Business Rules Best Practice
Posted by AJM on Sat, 07 Apr 2007 08:36:52 GMT
View Forum Message <> Reply to Message

The best place to put the business rules for an entity (database table) is the class through which that database table is accessed. Each table class contains the following customisable methods which are activated in particular circumstances:

_cm_validateInsert() is used just before a new record is inserted.
_cm_validateUpdate() is used just before an existing record is updated.
_cm_commonValidation() is used for code which is common to both insert and update.
_cm_validateDelete() is used to verify that a record can be deleted.

By default each database table class can only insert or update records on a single table, but you can use custom code in any custom method to access as many tables as you want to perform whatever operation you want. Any related records on child tables will automatically be deleted if the relationship is set to CASCADE.

When it comes to long operations which may exceed the browser's timeout limit, the answer is not to perform those operations through the browser. Instead you can run them through your operating system as a batch job. Details on how to do this are contained in http://www.tonymarston.net/php-mysql/infrastructure-faq.html #faq56. This procedure makes use of the getData_batch() and fetchRow() methods which allow you to read a record and process it before reading the next record. This is in contrast to the normal getData() method which retrieves a specified number of records for display in the current screen. If more records are available then you must resubmit the form with a request for another page of records.

I hope you find this information useful.