What exactly consists of 'Business Logic' in an application? [closed]

Asked 16 years, 3 months ago Modified 15 years, 3 months ago

Viewed 7k times

21

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 3 years ago.

Improve this question

I have heard umpteen times that we 'should not mix business logic with other code' or statements like that. I think every single code I write (processing steps I mean) consists of logic that is related to the business requirements..

Can anyone tell me what exactly consists of business logic? How can it be distinguished from other code? Is there some simple test to determine what is business logic and what is not?

business-logic-layer

Share

Improve this question

Follow

edited Sep 2, 2008 at 11:46

Rob Cooper

28.9k ● 26 ● 105 ● 142

asked Sep 2, 2008 at 11:40

Niyaz

54.8k ● 56 ● 152 ● 183

8 Answers

Sorted by: Highest score (default)

49

Simply define what you are doing in plain English. When you are saying things businesswise, like "make those suffer", "steal that money", "destroy this portion of earth" you are talking about business layer. To make it clear, things that get you excited go here.

When you are saying "show this here", "do not show that", "make it more beautiful" you are talking about the presentation layer. These are the things that get your designers excited.

When you are saying things like "save this", "get this from database", "update", "delete", etc. you are talking about the data layer. These are the things that tell you what to keep forever at all costs.

Share Improve this answer

edited Sep 17, 2009 at 20:30

Follow

mmr
14.9k ● 29 ● 97 ● 148

answered Sep 2, 2008 at 11:43

Serhat Ozgel
23.7k ● 29 ● 104 ● 141

1 Its not that easy as explained by Serhat imho. There are things you can fetch from database and do them in-memory in the business layer or you do them completely in the data access layer. – Elisabeth Nov 4, 2012 at 12:09

12

It's probably easier to start by saying what *isn't* business logic. Database or disk access isn't business logic. UI isn't business logic. Network communications aren't business logic.

To me, business logic is the rules that describe how a business operates, not how a software architecture operates. Business logic also has a tendency to change. For example, it may be a business requirement that every customer has a single credit card associated with their account. This requirement may change so that customers can have several credit cards. In theory, this should just be a change to the business logic and other parts of your software will not be affected.

So that's theory. In the real world (as you've found) the business logic tends to spread throughout the software. In the example above, you'll probably need to add

another table to your database to hold the extra credit card data. You'll certainly need to change the UI.

The purists say that business logic should always be completely separate and so would even be against having tables named "Customers" or "Accounts" in the database. Taken to its extreme you'd end up with an incredibly generic, impossible to maintain system.

There's definitely a strong argument in favour of keeping most of your business logic together rather than smearing it throughout the system, but (as with most theories) you need to be pragmatic in the real world.

Share Improve this answer

Follow

answered Sep 2, 2008 at 11:59

roomaroo

5,871 ● 4 ● 32 ● 31

5

To simplify things to a single line...
Business Logic would be code that doesn't depend on/won't change with a specific UI/implementation detail..
It is a code-representation of the rules, processes, etc. that are defined by/reflect the business being modelled.

Share Improve this answer

Follow

answered Sep 2, 2008 at 12:01

Gishu

137k ● 47 ● 226 ● 311

I think you confusing business logic with your application requirements. It's not the same thing. When someone

5

explains the logic of his/her business it is something like:

"When a user buys an item he has to provide delivery information. The information is validated with x y z rules. After that he will receive an invoice and earn points, that gives x% in discounts for the y offers" (sorry for the bad example)

When you implement this business rules you'll have to think in secondary requirements, like how the information is presented, how it will be stored in a persistent way, the communication with application servers, how the user will receive the invoice and so on. All this requirements are not part of business logic and should be decoupled from it. This way, when the business rules change you will adapt your code with less effort. Thats a fact.

Sometimes the presentation replicates some of the business logic, mostly in validating user input. But it has to be also present in the business logic layer. In other situations, is necessary to move some business logic to the Database, for performance issues. This is discussed by Martin Fowler here.

Share Improve this answer

Follow

answered Sep 2, 2008 at 14:36

Marcio Aguiar

14.5k ● 6 ● 40 ● 42

I would add that business logic must be at the database level many times for data integrity reasons not just performance. The reality is that many sources can affect data in a

database, if the business rule must in all cases be applied, it belongs in the database. – HLGEM Sep 17, 2009 at 20:57

2

I dont like the BLL+DAL names of the layers, they are more confusing than clarifying.
Call it DataServices and DataPersistence. This will make it easier.

Services manipulate, persistence tier CRUDs (Create, Read, Update, Delete)

Share Improve this answer

Follow

answered Sep 2, 2008 at 11:51

Lars Mæhlum

6,102 ● 3 ● 29 ● 32

0

For me, " *business logic* " makes up all the entities that represent data applicable to the problem domain, as well as the logic that decides on "what do do with the data"..

So it should really consist of "data transport" (not access) and "data manipulation".. Actually data access (stuff hitting the DB) should be in a different layer, as should presentation code.

Share Improve this answer

Follow

answered Sep 2, 2008 at 11:43

Rob Cooper

28.9k ● 26 ● 105 ● 142

0

If it contains anything about things like form, button, etc.. it's not a business logic, it's presentation layer. If it contains persistence to file or database, it's DAL. Anything in between is business logic. In reality, anything non-UI sometimes gets called "business logic," but it should be something that concerns the problem domain, not house keeping.

Share Improve this answer

Follow

answered Sep 2, 2008 at 11:46

Eugene Yokota

95.5k ● 45 ● 217 ● 320

0

Business logic is pure abstraction, it exists independent of the materialization/visualization of the data in front of your user, and independent of the persistence of the underlying data.

For example, in Tax Preparation software, one responsibility of the business logic classes would computation of tax owed. They would not be responsible for displaying reports or saving and retrieving a tax return.

@Lars, "services" implies a certain architecture.

Share Improve this answer

Follow

answered Sep 2, 2008 at 11:53

Aidan Ryan

11.6k ● 13 ● 59 ● 87