Good architecture and solution structure Cheat Sheet

by Skepticoder via cheatography.com/144584/cs/31050/

DATABASE

Create tables with good names (singular and often similar to domain model class names)

Use good column names (singular, with only standard abbreviations in the names

Normalize DB to 3rd normal form

Add foreign key and unique constraints

Use IDENTITY (autogenerated keys) on surrogate key columns

Implement robust cascading update/delete choices on foreign key constraints for referential integrity

Add good denormalization choices for speeding up working with the DB

DATA ACCESS LAYER

Add a DAO interface per database table with the CRUD actions needed

Add a concrete implementation of each DAO interface (preferably in a package/namespace that designates the persistence layer brand, i.e. MSSQL, MySQL, Oracle)

Add a DAO factory which can provide instances of the DAO classes for the controllers through static methods (e.g. `public static Cu stom erDao getCus tom erDao()`)

Add a Datacontext class with a static method which can provide a Connection object (e.g. `getObj ect()` method) for use by the DAO classes

Add a unique database login (preferable Active Directory account for Windows authentication or non-SA for SQL logins) with only minimal permissions necessary (e.g. dbreader/dbwriter) on the relevant DB/tables

MODEL LAYER

Mimic the database tables' structure with regards to naming and data types (often one model class per table)

Add a good toString() method to your model classes for debugging/visualizing object state

Use good encapsulation (especially of collections) so you can change the underlying data structure without changing the interface

Code good constructors which ensure you can't instantiate objects in an invalid state

DAO CLASSES SHOULD HAVE

DAO CLASSES SHOULD HAVE (cont)

`update (model object)`

`delete (model object)`

the extra "crud" methods

`deleteById(int id) (or other name for key column: deleteByAccount-Number, etc.) - findByPartOfName(String partOfName) (or other searchable attribute)

`findBy Par tOf Nam e(S tring partOf Name)` (or other searchable attribute)

transactions wherever you need to do two or more operations as a unit: e.g.

read inventory count and place order if product is available

move assets (e.g. funds from one account to another)

update denormalized values (e.g. a numberOfLikes column, which stores the number of likes in the like table for fast access)

Proper transaction structure with a

`beginT ran sac tion()` (C#)/`setAut oCo mmi t(f alse)` (Java)

`commit()` at the end of the "try..." which contains the CRUD code

a `rollback()` in the "catch..."

any cleaning up (Connection.setAutoCommit(false), etc.) in the "finally..."

Check for the number of rows affected on updates (UPDATE, INSERT, DELETE) and throw an Exception if necessary with a good exception message (e.g. "Unknown error. Account number 354 was not deleted.").

try-catch code which rethrows any exceptions that happen while interacting with the database, with a message about what was attempted when the exception happened plus the original exception.

a helper method for converting a single row in a ResultSet to an object

a helper method which converts an entire ResultSet to a List of objects reusing the above method

the 5 basic CRUD methods

`getAll()`

`getByI d(int id)` (or other name for key column, `getByS SN( ssn)`, etc.)

`insert (model object)`

By **Skepticoder**

cheatography.com/skepticoder/

Not published yet.
Last updated 6th March, 2022.