Web development and back-end call efficiency

Asked 16 years, 1 month ago Modified 14 years, 5 months ago

Viewed 474 times

2

Here's the scenario:

You have an ASP.Net application supported by a Microsoft SQL Server database, and for this example there won't be any caching.

For each page in your web application, what's more efficient:

Attempting to condense everything you need into one (or a few) stored procedure calls that return multiple tables with all of the data you need (which you save into a dataset),

or

to make each call separate (reading each call via a datareader), depending on how it logically makes sense.

The reason I ask is I always end up doing things the 2nd way: creating simple methods that connect to the database and grab some data for each little task I have (i.e. a method for populating a specific dropdown, another for the main information for the page, etc).

My concern is that what ends up happening when a page is requested, if I look at the SQL Profiler for the server, there will end up being up to 10 or so calls for that single page request. Is that excessive? In my mind it seems like it would be more efficient to condense data gathering by pages opposed to by tasks. Anyone have any experience with this?

asp.net sql-server

Share

Improve this question

Follow

edited Jul 22, 2010 at 20:00

Tom H

47.4k ● 15 ● 89 ● 131

asked Nov 5, 2008 at 14:58

John

17.5k ● 17 ● 68 ● 86

6 Answers

Sorted by: Highest score (default)

3

A large part of my day job is working on a huge WinForms app that is backed by a 600+ table sql server database.

We do it the first way to minimise network traffic. It's believed that its better to have a single bulging envelope than a mail sack full of envelopes.

Also, bundling data for transmission should not be confused with tight coupling - the SQL to gather data can be as modularised as ever, lightly bound together with an umbrella stored proc or view.

Share Improve this answer

Follow

answered Nov 5, 2008 at 15:00

Ed Guiness

35.1k ● 16 ● 113 ● 149

3

Don't group data gathering by pages. You're too tightly coupling data and presentation. What if tomorrow the data has to go on a diff page?

Share Improve this answer

Follow

answered Nov 5, 2008 at 15:01

Kon

27.4k ● 12 ● 63 ● 87

2

With connection pooling enabled, there isn't much advantage (if any) to condensing multiple DB calls into one as you're suggesting. There is nothing wrong with getting data as you need it with multiple calls to the DB, even if each call opens and closes a database connection. This is because with pooling enabled, the connections aren't really opened and closed each time.

If your application was instead a Windows client application, then it might make sense to "bundle" the calls into one, if the network speed between client and server

was particularly low. You're working with a web application, so the relevant connection speed here is between the web server and the DB server, which should not be a problem.

Bundling your data is just extra work with no payoff.

Share Improve this answer

Follow

answered Nov 5, 2008 at 15:16

MusiGenesis

75.3k ● 41 ● 197 ● 338

> connection pooling is probably less of a concern with a 'chatty' database interface as is the overhead of each request/response on the db server; in the end the same amount of data is sent, but 1 request vs 10 requests may be significant, depending on the pipe size and db server power of course – Steven A. Lowe Nov 5, 2008 at 16:04

1

If you really want to know which way is more efficient, you are going to have to profile each method. There are too many variables that go into each approach that could give a different result between pages and between your code and mine.

But generally speaking:

- more than one call for data is too many (sql connections and network traffic overhead),
- try to pull only what you need (get a dataset that is specific for your page if you must),

- tune for query performance. (gains from using proper query techniques- parameterized queries / stored procedures, proper indexes, etc - is going to help more than anything)

To add to the comments of fallen and edg, you should take care to keep data (model) and the page (view / presentation) segregated by a controller layer. The controller layer can be made to get the data for the view in either way that you choose (whichever is more efficient). This pattern (MVC / MVP) will allow you to have very manageable and reusable code, and allow you more leeway in testing different approaches.

Share Improve this answer Follow

edited Nov 5, 2008 at 16:26

answered Nov 5, 2008 at 15:20

StingyJack

19.4k ● 11 ● 67 ● 126

0

I had a similar concern a while back while I was putting together a web app that had many dropdown controls on different pages. I took a look at SQL Profiler and I was getting over 20 hits just to populate my dropdowns. Here what I did and maybe it will help you.

My picklist tables names (fictional)

- pl_usertypes

- pl_projecttypes
- pl_stattypes

So what I did was create a static method GetPicklistData(String[] tablename) which returned a DataSet. So my Page_Load method would look something like this:

```
protected void Page_Load(object sender, EventArgs e)
{
  DataSet ds = PicklistHelper.GetPicklistData(
      new String[]{"pl_usertypes","pl_projecttypes","p
  // Bind dropdowns
  ddl_a.DataSource = ds.Tables[0]; // pl_usertypes
  ddl_a.DataBind();

  ddl_b.DataSource = ds.Tables[1]; // pl_projecttypes
  ddl_b.DataBind();

  ddl_c.DataSource = ds.Tables[2]; // pl_stattypes
  ddl_c.DataBind();
}
```

So when I check SQL Profiler it reads

```
Select * from pl_usertypes; Select * from pl_projectty
pl_stattypes;
```

So it pretty obvious that the GetPicklistData() is building my select statements.

This here was probably me being "anal" but hey I dont know why I looked in SQL Profiler anyway. :)

Share Improve this answer

Follow

answered Nov 5, 2008 at 17:21

sykespro

258 ● 1 ● 2 ● 8

0

Doesn't it depend on how often your content backend is changing,? you could output a lot of your content "for XML" and have your site read the xml

Share Improve this answer

Follow

answered Nov 6, 2008 at 2:18

CPU_BUSY

801 ● 2 ● 6 ● 15