Provisioning SQL 2008 Database with C# Application

Asked 16 years, 1 month ago Modified 16 years, 1 month ago

Viewed 386 times

0

I have an internal enterprise application I've developed for my company built on .Net 3.5 / SQL 2008.

I have two types of databases. The main system database which contains all of our global data, such as user names, and customers, etc. And project databases, which contain the actual data pertaining to our clients project.

When the system creates a new project for a customer, it needs to provision a new SQL database using a custom schema providing the tables, views, sps, etc. The name of the project database corresponds to the Project ID of the project stored in the system database. So a new project will create a new project database with the name: Project_XXX where XXX is the project id.

My question is what is the best way to provision a custom database programatically? Right now the only way I can think to do it is have a class which reads a SQL script from the file system and does a parse to replace the project ID for the name of the database. This is easy, but seems rather inelegant.

Any approaches some veterans out there prefer over this?

.net sql-server

Share

Improve this question

Follow

asked Nov 18, 2008 at 2:12

Jiyosub

1,220 ● 1 ● 11 ● 15

3 Answers

Sorted by: Highest score (default)

1

Generally if there is only ever one (or a handful) of databases, and you have direct control over them (typicall corporate environment), I'd recommend not auto upgrading the databases as it is more hassle than it's worth. Just pass the script on to the people doing the install.

For more widespread releases in the past I have used a script as you suggest along with the SQL Server SMO library (Server.CurrentContext.ExecuteNonQuery()). I don't find it inelegant is it is simple and it works.

For the first release we would include a full DB build script, then add an upgrade script for each subsquent release. So if someone installs v1.2 over v1.1 we would only run the v1.2 script. However if they did a fresh install we would run v1.0, v1.1 and v1.2.

Share Improve this answer

Follow

answered Nov 18, 2008 at 2:24

Robert Wagner
17.8k ● 9 ● 58 ● 72

1

I would think the best way would be to have the script for this maintained on your Source Code Control server; it is, after all, the source code for a big part of your systems infrastructure, and would benefit from proper code maintenance, testing, etc. as much as any of the rest. Then just install from there.

In case you have any nagging doubts about this architecture, I'll presume to reinforce your uncertainty about separate databases per project. I call this the Big Denorm; I've seen it a number of times in a number of contexts, and I've yet to see it turn out well. I'd even nominate it for consideration as an Antipattern.

Share Improve this answer

Follow

answered Nov 18, 2008 at 2:46

dkretz
37.6k ● 13 ● 83 ● 140

Big Denorm is interesting. Our problem is we are an ASP and the project databases are giant. It's really the only way we could partition this much data. It also allows us to archive the entire database. But if you have a better pattern, I'm all ears.
– Jiyosub Nov 18, 2008 at 2:58

Nothing fancy, just have one set of tables and an extra column for each for the "ProjectID". It's a little extra work to save a lot of work maintaining consistency across numerous copies of schemas (including table defs, indexes, sps, views, whatever). – dkretz Nov 19, 2008 at 0:26

The best way to visualize it is to say, if you end up with 50 projects, you really will end up asking youself "how did I end up doing *every single thing* 50 times.". – dkretz Nov 19, 2008 at 0:28

Whereas if the partitioning is in the code with project ids, once you've built in the overhead for two projects, there's nothing extra to do for the next 48. – dkretz Nov 19, 2008 at 0:30

0

Since you are using .NET 3.5 you may want to consider using LINQ. LINQ has the ability to dynamically create a database (link to howto) using a mapping file or the strongly typed DBML file you've created. I believe that it can create the appropriate tables in multiple places. This probably works ok for simple databases.

I can't see how it would work, however, if you had functions or stored procedures and/or indexes on non-primary keys. The mapping file may be able to track these, but I've never used one so I'm skeptical. Probably @doofledorfer's idea on using SQL scripts to recreate the database structure in a new database is the way to go. I typically keep these scripts in my source code control anyway.

If you haven't already done so you may want to look into tools for managing updates. Presumably you'll need to roll any db changes across existing projects as your code/db changes. We've had good results with Red Gate SQL tools.

Share Improve this answer Follow

answered Nov 18, 2008 at 3:49

tvanfosson
532k ● 102 ● 699 ● 798

Yea, thought of that. The problem is I have items outside of linq, such as derived columns and heirarchyid. – Jiyosub Nov 18, 2008 at 12:07