****

|  |
| --- |
| Database Change Management Done Right  <http://projectroundhouse.org> |

|  |  |
| --- | --- |
| RoundhousE | Getting Started |

**Contents**

[Prerequisuites 4](#_Toc295255559)

[Database Installed 4](#_Toc295255560)

[.NET Framework 3.5 4](#_Toc295255561)

[Other Prerequisites 4](#_Toc295255562)

[Get RoundhousE 4](#_Toc295255563)

[Download 4](#_Toc295255564)

[Get the Source 5](#_Toc295255565)

[Run the Sample for Familiarization 6](#_Toc295255566)

[Sample Prerequisites 6](#_Toc295255567)

[Running It 6](#_Toc295255568)

[Roundhouse Change Drop Folder 9](#_Toc295255569)

[RoundhousE Migration Process 10](#_Toc295255570)

[Information Stage 10](#_Toc295255571)

[Creation Stage 10](#_Toc295255572)

[Versioning Stage 11](#_Toc295255573)

[Migrations Stage 11](#_Toc295255574)

[Post Run Information Stage 15](#_Toc295255575)

[Database Look 15](#_Toc295255576)

[RoundhousE.Version 16](#_Toc295255577)

[RoundhousE.ScriptsRun 16](#_Toc295255578)

[RoundhousE.ScriptsRunErrors 17](#_Toc295255579)

[Running A Second Time 17](#_Toc295255580)

[Running A Second Time After Changing a One Time File 19](#_Toc295255581)

[Add RoundhousE to Your Project 20](#_Toc295255582)

[How To Structure Your Database Scripts 22](#_Toc295255583)

[SQL Folder 22](#_Toc295255584)

[RoundhousE Configuration 22](#_Toc295255585)

[Barebone 22](#_Toc295255586)

[Full Configuration 23](#_Toc295255587)

[Version Your Database 25](#_Toc295255588)

[Environment Specific Scripts 25](#_Toc295255589)

Welcome to RoundhousE (RH)! RH is one of the most intelligent database change management (also known as migration) OSS tools out there. Over the next couple of chapters you are going to learn how to effectively use RoundhousE to solve the complex business problem of managing change for your database. You will see that RH makes it really easy and as you learn the tool, you will wonder how you ever used anything else.

# Prerequisuites

## Database Installed

Whatever database you are going to use, be it SQL Server or be it MySQL, needs to be installed so you can do database change management.

## .NET Framework 3.5

You will also need to have the .NET framework installed. Currently RH uses 3.5.

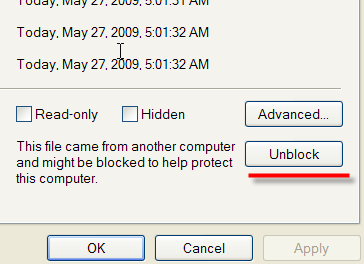
## Other Prerequisites

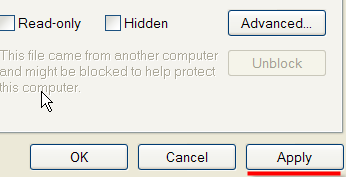
The install agent (usually a person running rh) needs to be able to create databases on the database (which usually means SA).

# Get RoundhousE

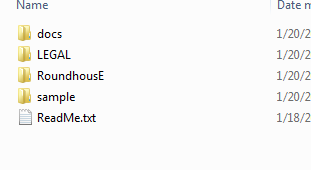
## Download

1. Head out to <http://code.google.com/p/roundhouse/downloads/list> and download the latest.
2. One thing to note, when you do download the zip file, the very first thing you will want to do (before extracting) is right click on that file and select [**Properties**]. On the **General** tab click {**Unblock**} if you have it. Otherwise you may have to unblock each file one at a time later.



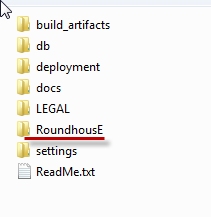


1. Unzip the files. What we need to use RH is in the RoundhousE folder.



## Get the Source

1. If you like living on the bleeding edge, then you can also get the source.
2. The other way to get RH is to point your favorite SVN client at <http://roundhouse.googlecode.com/svn/trunk/>.
3. Or GIT – <https://github.com/chucknorris/roundhouse>
4. Once it has completed downloading, just click on the build.bat (or open a command line in this directory and type build). It may take awhile since it ilmerges 3 different library sets.
5. Once it has completed, there is a folder called **code\_drop**. Inside of that folder is RoundhousE. This is where we get what we need to use RH.



# Run the Sample for Familiarization

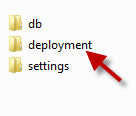
If you’ve downloaded the application, the best way to see the power of RoundhousE is to actually run the sample. It will also help you familiarize yourself with how RoundhousE works.

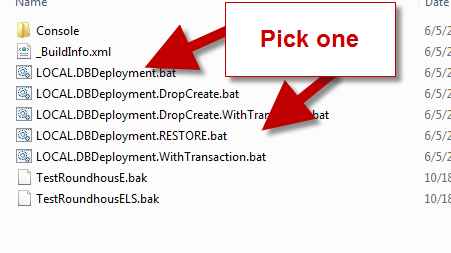
## Sample Prerequisites

The sample is run against SQL Server 2005/2008 (and it works with express as well), so you would need that installed. You can also attempt to run the sample against other database types.

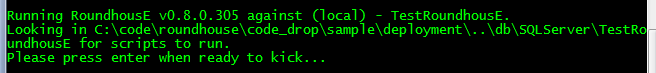
## Running It

1. Go to **code\_drop/sample** (if you have the source) or to **sample** if you downloaded the release. Go into the **deployment** folder.

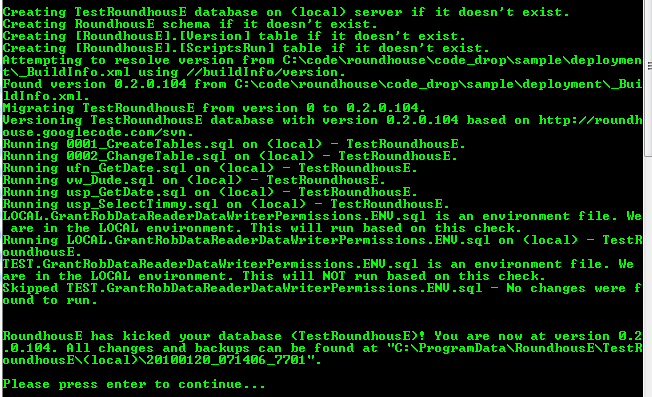
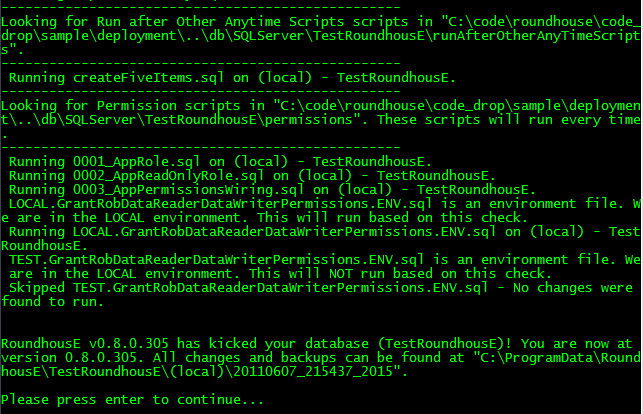
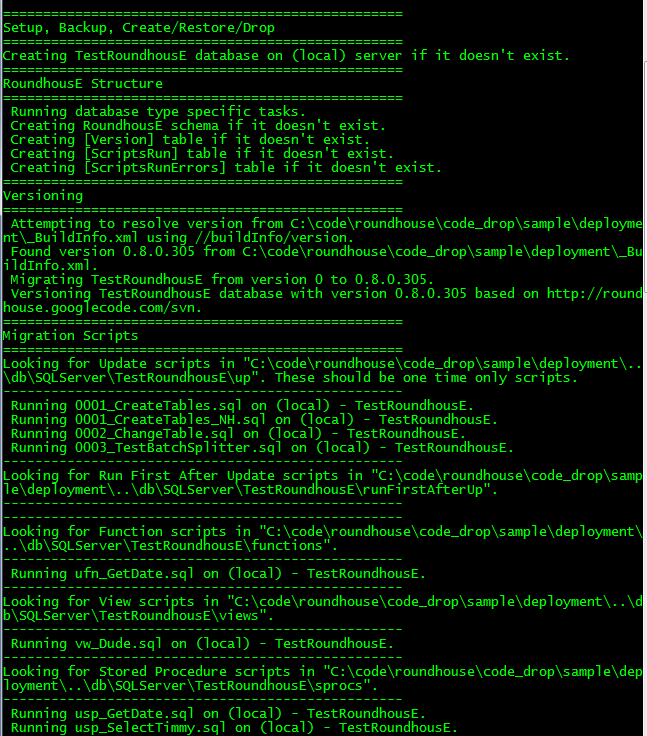




1. Run LOCAL.DBDeployment.bat.



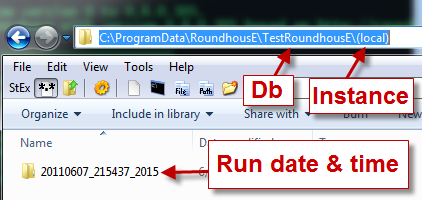
1. Press **{Enter}**. Notice all of the things it does. It just created our database, versioned it, then applied all of the database scripts (marking them as part of this version). Then it saved the output of what it ran to a folder.



## Roundhouse Change Drop Folder

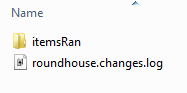
Notice how it ended telling us there was a folder where it put all of the changes it made. Let’s head there.

Below is the output folder. The structure of the output folder is *databasename*\*server\yyyyMMdd\_HHmmss\_ffff*.

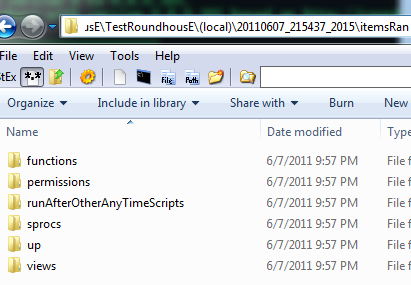


The convention here is the common application data folder on a machine followed by RoundhousE. Then we have a folder for the particular database followed by the instance. The actual folder we drop the changes into are **YYYYMMDD\_HHmmss\_fractions**. Hence the one in the picture was run on 06/07/2011 at 21:54:37 with so many fractions of a second.

In the folder we see two things:



The migration log (roundhouse.changes.log) and the actual sql files that it ran get dropped into the **itemsRan** folder.



You can see that on a first time run, that’s quite a bit of items.

Let’s take a look at that change log file more in depth.

## RoundhousE Migration Process

RH goes through several stages when it runs. Let’s take a look at the change log file and talk about each part of the migration process.

### Information Stage

Running RoundhousE v0.8.0.305 against (local) - TestRoundhousE.

Looking in C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE for scripts to run.

Please press enter when ready to kick...

The first thing you notice is that it tells you is the version and what server and what database it is going to run on. The database does not have to exist prior to run for SQL Server. It will create it automatically. Then it mentions where it is going to look for scripts. Next and slightly important, when it is in interactive mode (the default), it stops and gives you the opportunity to double check to be sure this is correct.

### Creation Stage

==================================================

Setup, Backup, Create/Restore/Drop

==================================================

Creating TestRoundhousE database on (local) server if it doesn't exist.

==================================================

RoundhousE Structure

==================================================

Running database type specific tasks.

Creating RoundhousE schema if it doesn't exist.

Creating [Version] table if it doesn't exist.

Creating [ScriptsRun] table if it doesn't exist.

Creating [ScriptsRunErrors] table if it doesn't exist.

Now it creates the database if it doesn’t exist (you can configure a custom database creation script if you don’t like the default). Then it creates its schema (schema name is configurable) and three roundhouse tables for keeping track of changes and migration errors. Version (table name is configurable) is where it keeps track of versioning. The ScriptsRun table (which has a foreign key back to Version, name is configurable) is where it notes what files have run with every migration. The ScriptsRunErrors table (name is configurable) is where you can get to the actual error that has occurred.

### Versioning Stage

==================================================

Versioning

==================================================

Attempting to resolve version from C:\code\roundhouse\code\_drop\sample\deployment\\_BuildInfo.xml using //buildInfo/version.

Found version 0.8.0.305 from C:\code\roundhouse\code\_drop\sample\deployment\\_BuildInfo.xml.

Migrating TestRoundhousE from version 0 to 0.8.0.305.

Versioning TestRoundhousE database with version 0.8.0.305 based on http://roundhouse.googlecode.com/svn.

Versioning is very important and done exactly how you want to do versioning. RH has a convention that you provide it the version number through an XML file (and an xpath to the version) or a DLL file (uses the file version). The other thing it wants to know is the repository path (if provided). It is preferred that you version based on the same way you version your code. In this sample, versioning is done with source control revisions as the last number in the version. Notice that RH is looking at an XML file and XPath for versioning. As mentioned it can also use a DLL. It is intelligent enough to know the difference based on the file name from the configuration.

==================================================

Versioning

==================================================

Attempting to resolve assembly file version from C:\code\roundhouse\code\_drop\RoundhousE\lib\**roundhouse.dll**.

Found version 0.8.0.305 from C:\code\roundhouse\code\_drop\RoundhousE\lib\roundhouse.dll.

Migrating TestRoundhousE from version 0 to 0.8.0.305.

Versioning TestRoundhousE database with version 0.8.0.305 based on http://roundhouse.googlecode.com/svn.

### Migrations Stage

Migrating your database from one state to another is the most important stage in RH. It follows a particular order.

==================================================

Migration Scripts

==================================================

Looking for Update scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\up". These should be one time only scripts.

--------------------------------------------------

Running 0001\_CreateTables.sql on (local) - TestRoundhousE.

Running 0001\_CreateTables\_NH.sql on (local) - TestRoundhousE.

Running 0002\_ChangeTable.sql on (local) - TestRoundhousE.

Running 0003\_TestBatchSplitter.sql on (local) - TestRoundhousE.

--------------------------------------------------

Looking for Run First After Update scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\runFirstAfterUp".

--------------------------------------------------

--------------------------------------------------

Looking for Function scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\functions".

--------------------------------------------------

Running ufn\_GetDate.sql on (local) - TestRoundhousE.

--------------------------------------------------

Looking for View scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\views".

--------------------------------------------------

Running vw\_Dude.sql on (local) - TestRoundhousE.

--------------------------------------------------

Looking for Stored Procedure scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\sprocs".

--------------------------------------------------

Running usp\_GetDate.sql on (local) - TestRoundhousE.

Running usp\_SelectTimmy.sql on (local) - TestRoundhousE.

--------------------------------------------------

Looking for Run after Other Anytime Scripts scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\runAfterOtherAnyTimeScripts".

--------------------------------------------------

Running createFiveItems.sql on (local) - TestRoundhousE.

--------------------------------------------------

Looking for Permission scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\permissions". These scripts will run every time.

--------------------------------------------------

Running 0001\_AppRole.sql on (local) - TestRoundhousE.

Running 0002\_AppReadOnlyRole.sql on (local) - TestRoundhousE.

Running 0003\_AppPermissionsWiring.sql on (local) - TestRoundhousE.

LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will run based on this check.

Running LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql on (local) - TestRoundhousE.

TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will NOT run based on this check.

Skipped TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql - No changes were found to run.

RH then looks through the scripts folders and executes scripts. Let’s break that down a bit.

#### Up - DML/DDL Updates

Looking for Update scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\up". These should be one time only scripts.

--------------------------------------------------

Running 0001\_CreateTables.sql on (local) - TestRoundhousE.

Running 0001\_CreateTables\_NH.sql on (local) - TestRoundhousE.

Running 0002\_ChangeTable.sql on (local) - TestRoundhousE.

Running 0003\_TestBatchSplitter.sql on (local) - TestRoundhousE.

The **UP** folder (folder name is configurable) and subfolders is where you put your schema changes and database insert scripts. This is a one time only folder. That means once you have run a script here, if RH detects the file has changed (even so much as a space) it will shut down and report errors. Of course there is a configuration that will allow you to just warn on one time script changes (check the configuration section of the documentation).

**Notice the file naming convention here**. 0001 comes prior to 0002. RH runs files in the order it finds them. The convention is to start all scripts with a nice long numeric sequence.

#### Run First After Up – Any time scripts that need to run out of order

Looking for Run First After Update scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\runFirstAfterUp".

--------------------------------------------------

--------------------------------------------------

Scripts run prior to other anytime folders are found here (folder name is configurable) and subfolders. This folder exists to allow you to put sql files in when you need to run out of order, say a stored procecure prior to a function. It is not normal occurrence to have many files in here or any for that matter.

#### Functions

Looking for Function scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\functions".

--------------------------------------------------

Running ufn\_GetDate.sql on (local) - TestRoundhousE.

--------------------------------------------------

Functions are found in a functions folder (folder name is configurable) and subfolders. If you have any that need to run prior to others, make sure they are alphabetically first before the dependent scripts.

#### Views

Looking for View scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\views".

--------------------------------------------------

Running vw\_Dude.sql on (local) - TestRoundhousE.

--------------------------------------------------

Views are found in a views folder (name is configurable) and subfolders. If you have any that need to run prior to others, make sure they are alphabetically first before the dependent scripts.

#### Stored Procedures

Looking for Stored Procedure scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\sprocs".

--------------------------------------------------

Running usp\_GetDate.sql on (local) - TestRoundhousE.

Running usp\_SelectTimmy.sql on (local) - TestRoundhousE.

--------------------------------------------------

Stored procedures are found in a sprocs folder (folder name is configurable) and subfolders. If you have any that need to run prior to others, make sure they are alphabetically first before the dependent scripts.

#### Run After Other Anytime Scripts

Looking for Run after Other Anytime Scripts scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\runAfterOtherAnyTimeScripts".

--------------------------------------------------

Running createFiveItems.sql on (local) - TestRoundhousE.

--------------------------------------------------

This folder exists to allow you to run scripts after you have set up your anytime scripts (folder name is configurable). It’s pretty open what you put in here, but remember that it is still an **anytime folder**.

#### Permissions – Run Every Time

Looking for Permission scripts in "C:\code\roundhouse\code\_drop\sample\deployment\..\db\SQLServer\TestRoundhousE\permissions". These scripts will run every time.

--------------------------------------------------

Running 0001\_AppRole.sql on (local) - TestRoundhousE.

Running 0002\_AppReadOnlyRole.sql on (local) - TestRoundhousE.

Running 0003\_AppPermissionsWiring.sql on (local) - TestRoundhousE.

LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will run based on this check.

Running LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql on (local) - TestRoundhousE.

TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will NOT run based on this check.

Skipped TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql - No changes were found to run.

Permissions scripts are found in a permissions folder (name is configurable) and subfolders. If you have any that need to run prior to others, make sure they are alphabetically first before the dependent scripts.

Permissions may contain auto-wiring of permissions, so **they are run every time**. This is pretty interesting. We have a set of permissions scripts here. One called LOCAL.*something*.ENV.sql and another called TEST.*something*.ENV.sql. Only one of these runs based on the environment we are in. Yes, RH is environment aware (as long as you provide it in the configuration). If you take a look at these scripts again, one is called LOCAL.*something*.ENV.sql and the other is called TEST.*something*.ENV.sql. It’s the “**.ENV.”** that tells RH that it is an environment file. Now RH starts looking for a match in the name of the file to the environment it is in.

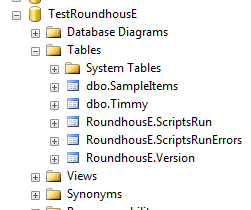
### Post Run Information Stage

RoundhousE v0.8.0.305 has kicked your database (TestRoundhousE)! You are now at version 0.8.0.305. All changes and backups can be found at "C:\ProgramData\RoundhousE\TestRoundhousE\(local)\20110607\_215437\_2015".

The last thing RH does is report success. It also tells you where you can find the changes it made in an output folder.

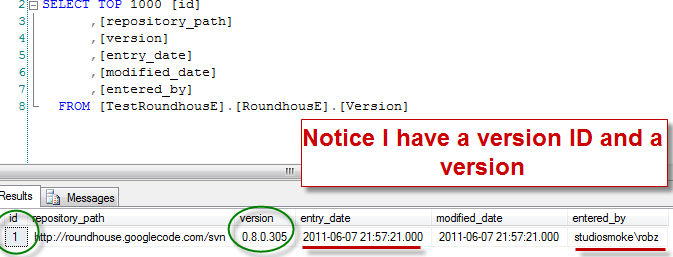
## Database Look

Now let’s take a look at the database we just created.

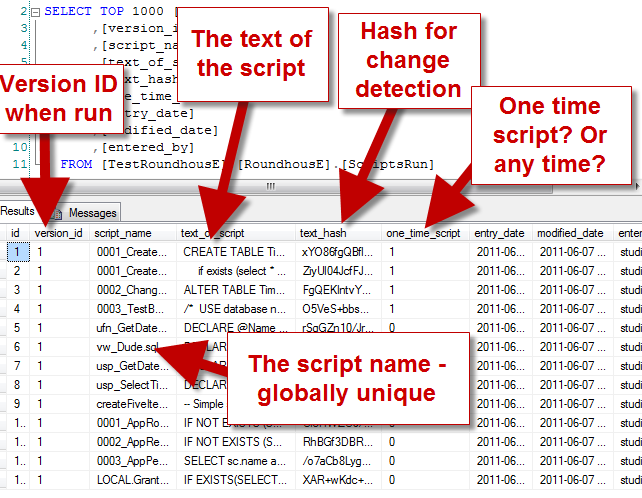


There it is with some tables in it. Let’s take a look at the roundhouse tables.

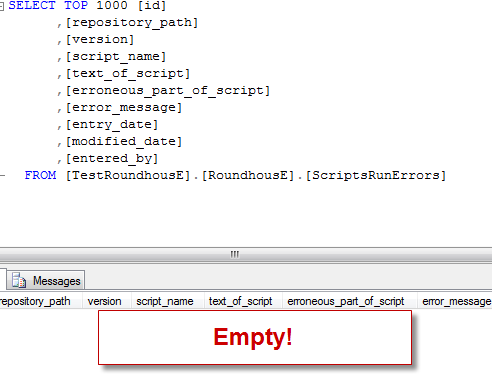
### RoundhousE.Version



### RoundhousE.ScriptsRun

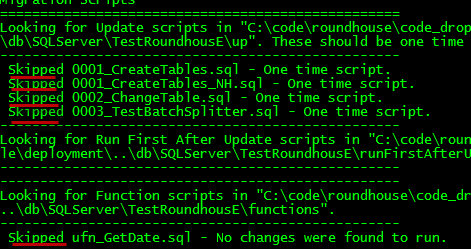


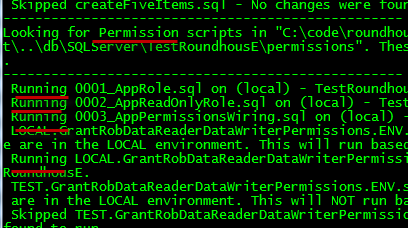
### RoundhousE.ScriptsRunErrors



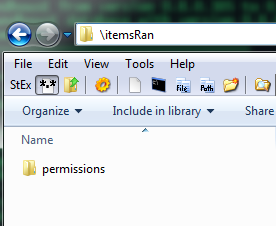
## Running A Second Time

So now you’re starting to see some of the abilities of RH. Now run the sample again. Notice how it skips over everything except the permissions folder.

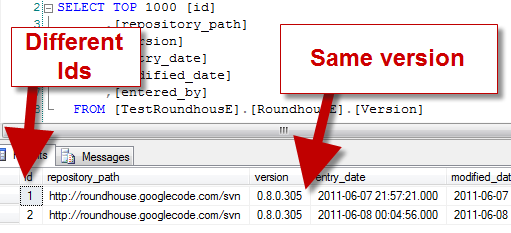




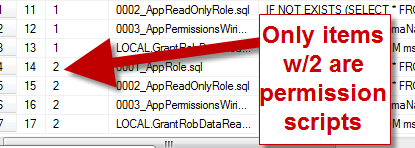
Take a look at that itemsRan folder.



The version table.

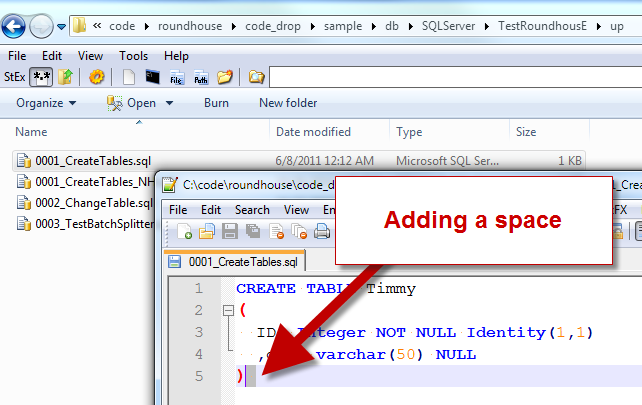


The scripts run tables records that for version id 2, only the permissions scripts were run.

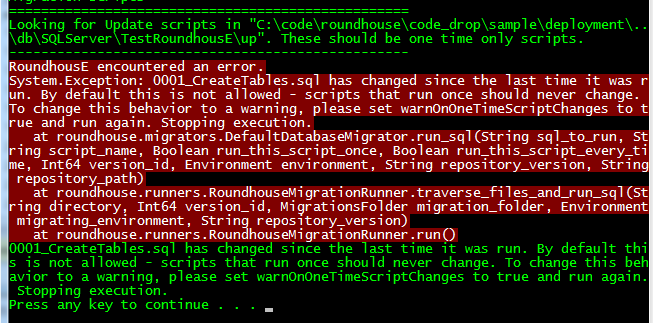


## Running A Second Time After Changing a One Time File

Let’s go into the up folder and add a space to one of the files. Save it and close it.

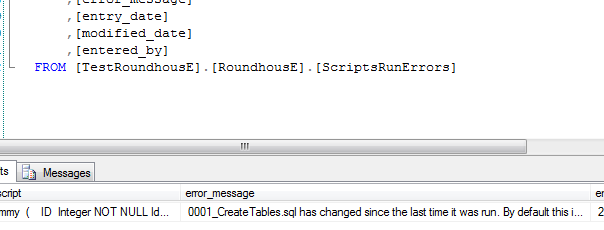


Now let’s run it and see what happens.



It shut down immediately and told me I had an error. This means I should have something in the script run errors table. Let’s take a look.

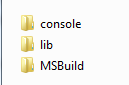
Sweet! Good error message here to work from.



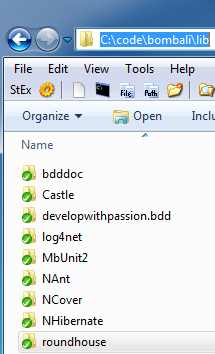
# Add RoundhousE to Your Project

Now that we have RoundhousE, we need to decide what technology we are going to use to do migrations. Are we going to use MSBuild, the dll, or the console application (or a combination of these)?

1. Go into the RoundhousE folder and choose console, lib or MSBuild.



1. In your project, you must decide where you keep the reference. I like to keep RH in my lib (tools) folder and move it to the deployment folder when I build my project. In some cases, you may already want it in your deployment folder to begin with. That is entirely up to you. If I was to make a recommendation, I would keep it in the lib folder and set up instructions for it to be moved to the deployment folder at build time.
2. Add it to either your deployment scripts folder or your lib folder. These instructions are for adding it to your lib (references/libs/thirdparty/3rdparty/etc) folder.



1. If you added it to the lib folder, there is a sample build customization script (in NAnt, specifically written for [UppercuT](http://projectuppercut.org/)) that will copy over RoundhousE console to a folder for deployment.

<target name="copy\_roundhouse\_files\_to\_deploy\_rh">

<echo message="Copying roundhouse assemblies to ${dirs.deployment}\rh."/>

<copy todir="${dirs.deployment}\rh">

<fileset basedir="${dirs.current}\..\lib\RoundhousE">

<include name="\*\*/\*.\*" />

</fileset>

</copy>

</target>

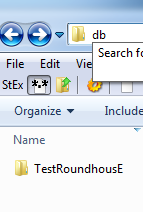
1. Now we are set up to start using RH and we need to set up our database scripts.

# How To Structure Your Database Scripts

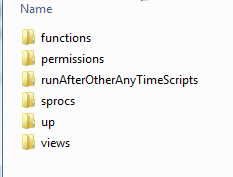
## SQL Folder

I normally just add a class library to a project and put all of my sql scripts in there to cut down on Visual Studio trying to be too smart with some database project. That way I can still get to the Sql files in Visual Studio.

You need a top level folder. Let’s call it DB. Under DB you want to put a folder with the name of the database in it. Let’s call it TestRoundhousE.



Under that folder goes the migrations folders - Up, Functions, Views, Sprocs, Permissions, etc



In each of those folders (and subfolders) go SQL script files. **They must end in .sql** for RH to run them.

# RoundhousE Configuration

## Barebone

At a minimum you just need to tell RoundhousE your database name. It will deploy to the local sql database server looking for sql script folders in the current directory.

rh /d bob

The command above would create a database named bob on the local sql server (default instance) using a trusted connection. You can also pass a connection string

## Full Configuration

This is based wholly on the output that you can get if you type rh /?

This shows how to specify the command followed by a pipe (“|”), the option name and usage. Also notes the default setting.

* -?, --help, -h | Prints out the options.
* -d, --db, --database, --databasename=VALUE | **REQUIRED: DatabaseName** - The database you want to create/migrate.
* -c, --cs, --connstring, --connectionstring=VALUE | **REQUIRED:** **ConnectionString** - As an alternative to ServerName and Database - You can provide an entire connection string instead.
* -f, --files, --sqlfilesdirectory=VALUE | **SqlFilesDirectory** - The directory where your SQL scripts are. Defaults to ".".
* -s, --server, --servername, --instance, --instancename=VALUE | **ServerName** - The server and instance you would like to run on. (local) and (local)\SQL2008 are both valid values. Defaults to "(local)".
* --csa, --connstringadmin, --connectionstringadministration=VALUE | **ConnectionStringAdministration** - This is used for connecting to master when you may have a different uid and password than normal.
* --dt, --dbt, --databasetype=VALUE | **DatabaseType** - Tells RH what type of database it is running on. This is a plugin model. This is the fully qualified name of a class that implements the interface roundhouse.sql.Database, roundhouse. If you have your own assembly, just set it next to rh.exe and set this value appropriately. Defaults to "roundhouse.databases.sqlserver.SqlServerDatabase, roundhouse.databases.sqlserver" which can also run against SQL Server 2005.
* -r, --repo, --repositorypath=VALUE | **RepositoryPath** - The repository. A string that can be anything. Used to track versioning along with the version. Defaults to null.
* --vf, --versionfile=VALUE | **VersionFile** - Either an XML file or a DLL that a version can be resolved from. Defaults to "\_BuildInfo.xml".
* --vx, --versionxpath=VALUE | **VersionXPath** - Works in conjunction with an XML version file. Defaults to "//buildInfo/version".
* -u, --up, --upfolder, --upfoldername=VALUE | **UpFolderName** - The name of the folder where you keep your update scripts. Will recurse through subfolders. Defaults to "up".
* --rf, --runfirst, --runfirstfolder, --runfirstafterupdatefolder, --runfirstafterupdatefoldername=VALUE | **RunFirstAfterUpdateFolderName** - The name of the folder where you keep any functions, views, or sprocs that are order dependent. If you have a function that depends on a view, you definitely need the view in this folder. Will recurse through subfolders. Defaults to "runFirstAfterUp".
* --fu, --functions, --functionsfolder, --functionsfoldername=VALUE | **FunctionsFolderName** - The name of the folder where you keep your functions. Will recurse through subfolders. Defaults to "functions".
* --vw, --views, --viewsfolder, --viewsfoldername=VALUE | **ViewsFolderName** - The name of the folder where you keep your views. Will recurse through subfolders. Defaults to "views".
* --sp, --sprocs, --sprocsfolder, --sprocsfoldername=VALUE | **SprocsFolderName** - The name of the folder where you keep your stored procedures. Will recurse through subfolders. Defaults to "sprocs".
* --ra, --runAfterOtherAnyTimeScripts, --runAfterOtherAnyTimeScriptsfolder,

--runAfterOtherAnyTimeScriptsfoldername=VALUE | **RunAfterOtherAnyTimeScriptsFolderName** - The name of the folder where you keep scripts that will be run after all of the other any time scripts complete. Will recurse through subfolders. Defaults to "runAfterOtherAnyTimeScripts".

* -p, --permissions, --permissionsfolder, --permissionsfoldername=VALUE | **PermissionsFolderName** - The name of the folder where you keep your permissions scripts. Will recurse through subfolders. Defaults to "permissions".
* --sc, --schema, --schemaname=VALUE | **SchemaName** - This is the schema where RH stores it's tables. Once you set this a certain way, do not change this. This is definitely running with scissors and very sharp. I am allowing you to have flexibility, but because this is a knife you can still get cut if you use it wrong. I'm just saying. You've been warned. Defaults to "RoundhousE".
* --vt, --versiontable, --versiontablename=VALUE | **VersionTableName** - This is the table where RH stores versioning information. Once you set this, do not change this. This is definitely running with scissors and very sharp. Defaults to "Version".
* --srt, --scriptsruntable, --scriptsruntablename=VALUE | **ScriptsRunTableName** - This is the table where RH stores information about scripts that have been run. Once you set this a certain way, do not change this. This is definitely running with scissors and very sharp. Defaults to "ScriptsRun".
* --sret, --scriptsrunerrorstable, --scriptsrunerrorstablename=VALUE | **ScriptsRunErrorsTableName** - This is the table where RH stores information about scripts that have been run with errors. Once you set this a certain way, do not change this. This is definitelly running with scissors and very sharp. Defaults to "ScriptsRunErrors".
* --env, --environment, --environmentname=VALUE | **EnvironmentName** - This allows RH to be environment aware and only run scripts that are in a particular environment based on the namingof the script. LOCAL.something.ENV.sql would only be run in the LOCAL environment. Defaults to "LOCAL".
* --restore | **Restore** - This instructs RH to do a restore (with the restorefrompath parameter) of a database before running migration scripts. Defaults to false.
* --rfp, --restorefrom, --restorefrompath=VALUE | **RestoreFromPath** - This tells the restore where to get to the backed up database. Defaults to null. Required if /restore has been set. NOTE: will try to use Litespeed for the restore if the last two characters of the name are LS (as in DudeLS.bak).
* --rco, --restoreoptions, --restorecustomoptions=VALUE | **RestoreCustomOptions** - This provides the restoreany custom options as in MOVE='Somewhere or another'.
* --rt, --restoretimeout=VALUE | **RestoreTimeout** - Allows you to specify a restore timeout in seconds. The default is 900 seconds.
* --cds, --createdatabasescript, --createdatabasecustomscript=VALUE | **CreateDatabaseCustomScript** - This instructs RH to use this script for creating a database instead of the default based on the SQLType.
* --drop | **Drop** - This instructs RH to remove a database and not run migration scripts. Defaults to false.
* --dc, --dnc, --donotcreatedatabase | **DoNotCreateDatabase** - This instructs RH to not create a database if it does not exists. Defaults to false.
* -o, --output, --outputpath=VALUE | **OutputPath** - This is where everything related to the migration is stored. This includes any backups, all items that ran, permission dumps, logs, etc. Defaults to a special folder, common application data with roundhouse as subdirectory, i.e. "C:\ProgramData\RoundhousE".
* -w, --warnononetimescriptchanges | **WarnOnOneTimeScriptChanges** - If you do not want RH to error when you change scripts that should not change, you must set this flag. One time scripts are DDL/DML (anything in the upFolder). Defaults to false.
* --silent, --ni, --noninteractive | **Silent** - tells RH not to ask for any input when it runs. Defaults to false.
* -t, --trx, --transaction, --wt, --withtransaction | **WithTransaction** - This instructs RH to run inside of a transaction. Defaults to false.
* --simple | **RecoveryModeSimple** - This instructs RH to set the database recovery mode to simple recovery. Defaults to false.
* --debug | **Debug** - This instructs RH to write out all messages.
* --runallanytimescripts, --forceanytimescripts | **RunAllAnyTimeScripts** - This instructs RH to run any time scripts every time it is run. Defaults to false

# Version Your Database

# Environment Specific Scripts

# Questions?

Please contact the Chuck Norris Framework group at <http://groups.google.com/group/chucknorrisframework>.