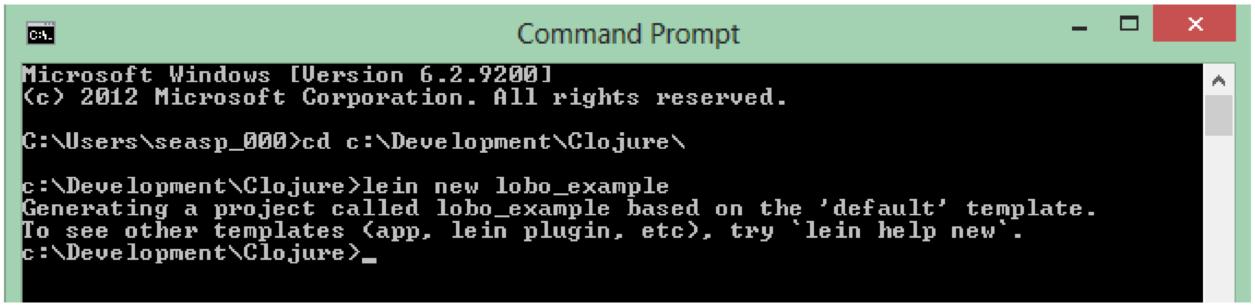
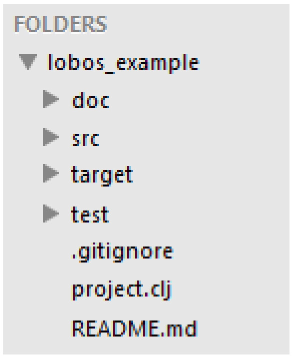
So as I've been exploring the world of Clojure, I've been looking for ways to make it easier on lower primates such as myself. Today I ran across a really easy way to set up tables in a database with Lobos (https://github.com/budu/lobos) and PostgreSql. Lucky for you, I wrote down how to use it… that may or may not have been helped by the actual documentation.

Ok so the first thing you need to do is create an actual project with Lienegen. This turns out to be pretty easy:

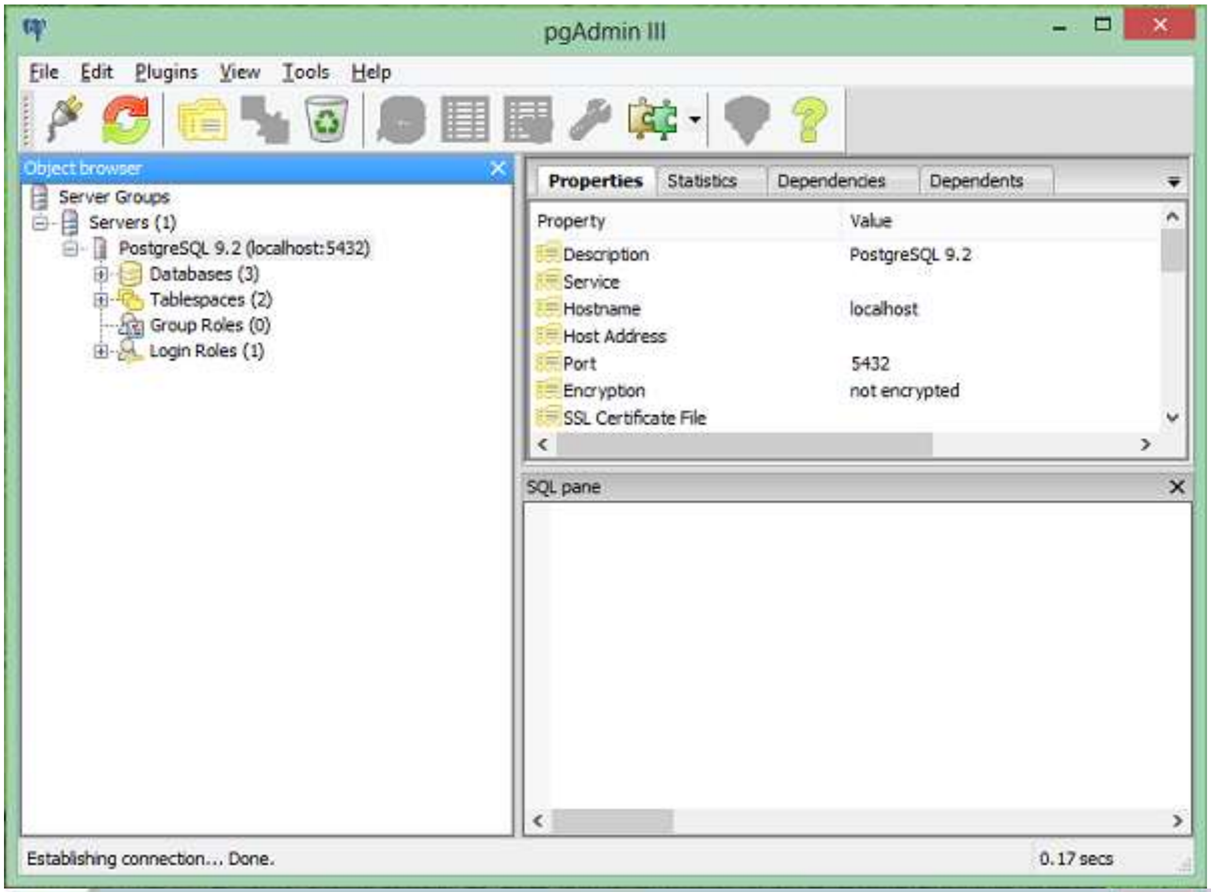


```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\seasp_000>cd c:\Development\Clojure\

c:\Development\Clojure>lein new lobo_example
Generating a project called lobo_example based on the 'default' template.
To see other templates (app, lein plugin, etc), try `lein help new`.
c:\Development\Clojure>_
```

Next step is opening the new project up with some sort of typewriter-less editing machine. I use Sublime 2, but that's not required. Unless you want to be cool… and don't have a Mac.



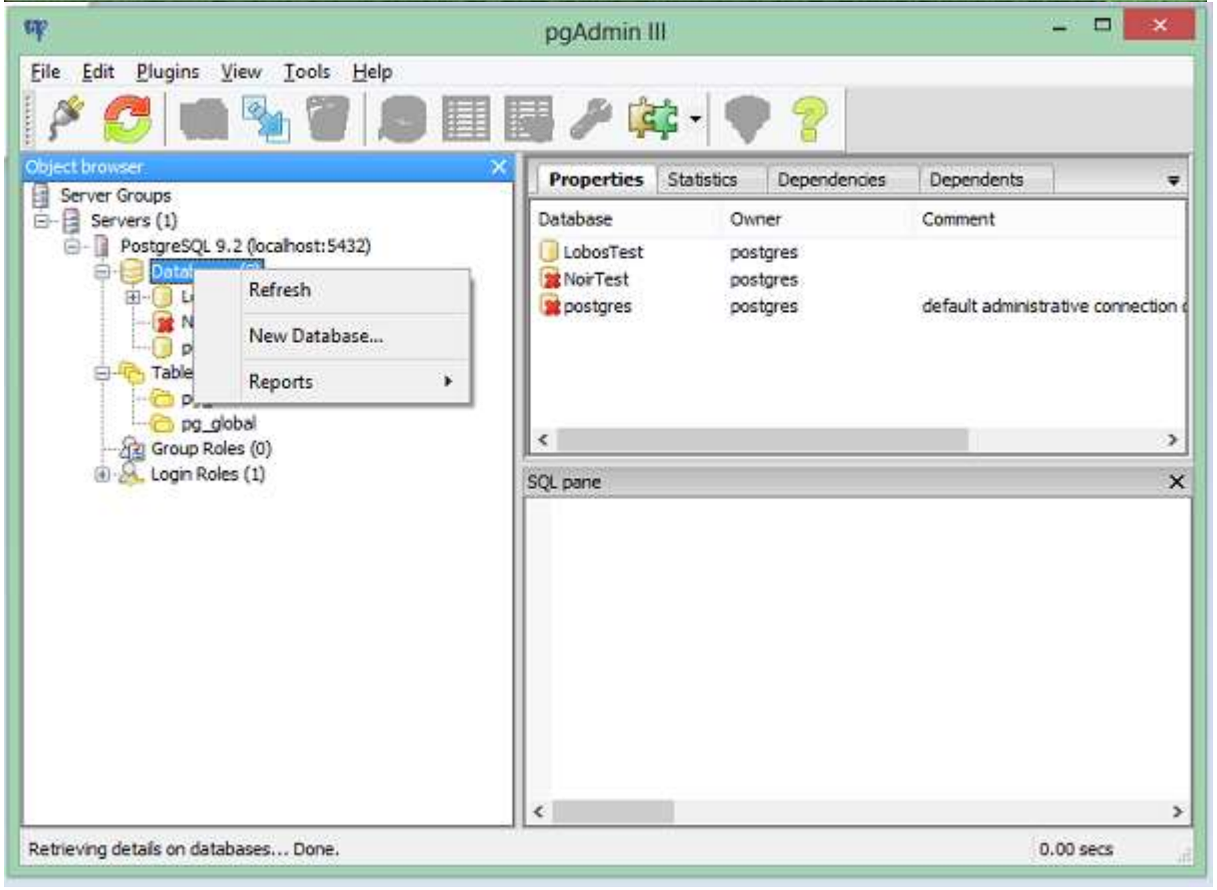
Attempt to contain your excitement. The next step is to open the pgAdmin UI thing (Using III which either means it is the third version, or someone's I key is stuck.), and create a database thing. By the way, I apologize if I'm using too many technical terms.


pgAdmin III
File Edit Plugins View Tools Help
Object browser
Properties Statistics Dependencies Dependents
SQL pane

Server Groups
Servers (1)
PostgreSQL 9.2 (localhost:5432)
Databases (3)
Tablespaces (2)
Group Roles (0)
Login Roles (1)

Property	Value
Description	PostgreSQL 9.2
Service	
Hostname	localhost
Host Address	
Port	5432
Encryption	not encrypted
SSL Certificate File	

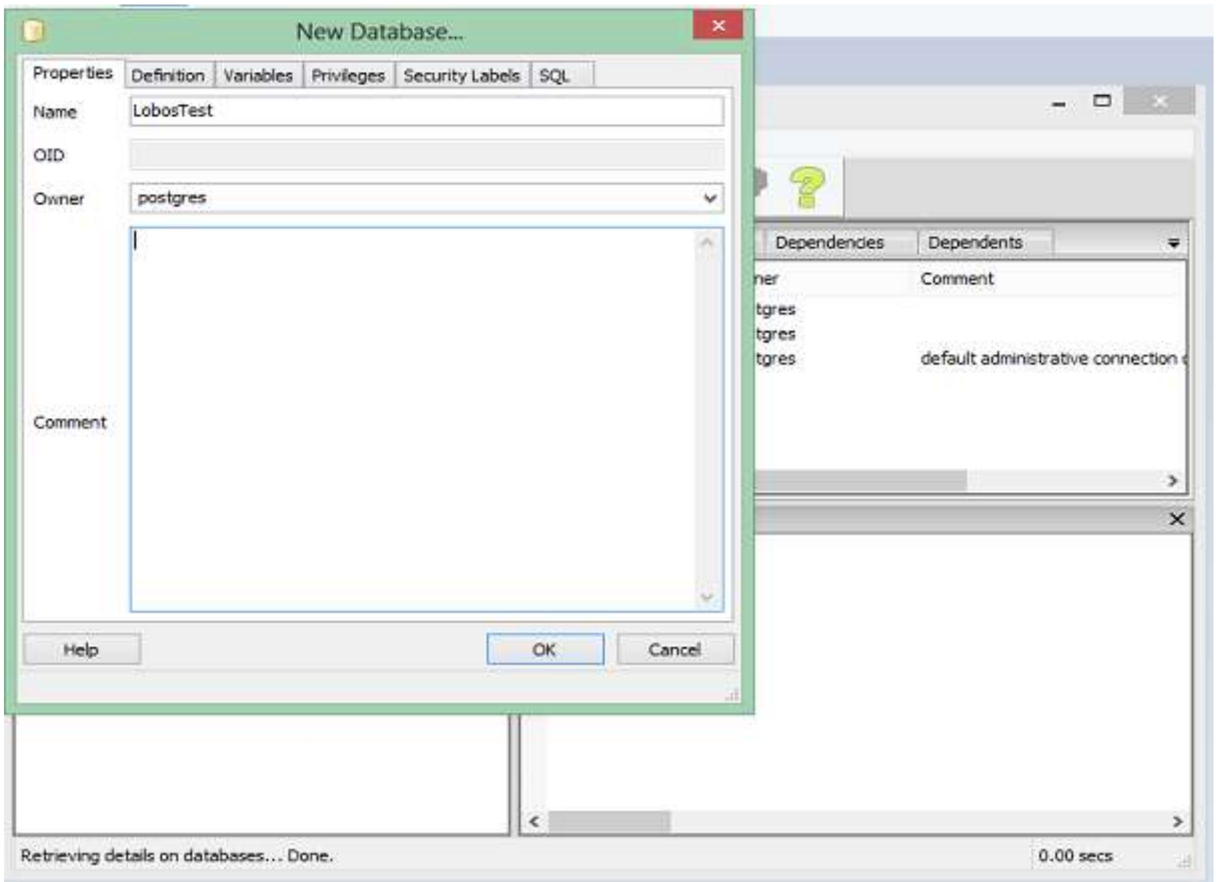
Establishing connection... Done. 0.17 secs


pgAdmin III
File Edit Plugins View Tools Help
Object browser
Refresh
New Database...
Reports
Properties Statistics Dependencies Dependents
SQL pane

Server Groups
Servers (1)
PostgreSQL 9.2 (localhost:5432)
Datal
Tabl
pg_global
Group Roles (0)
Login Roles (1)

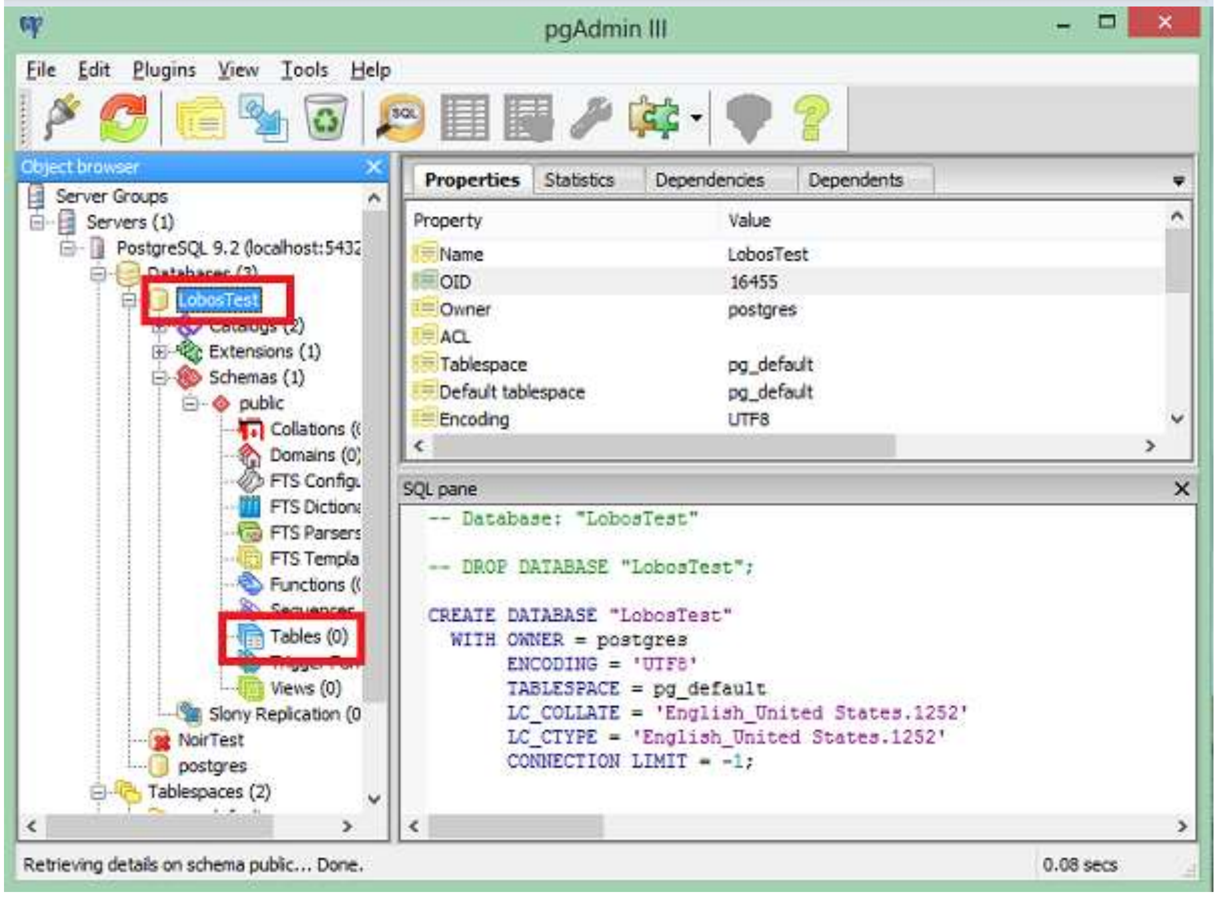
Database	Owner	Comment
LobosTest	postgres	
NoirTest	postgres	
postgres	postgres	default administrative connection

Retrieving details on databases... Done. 0.00 secs



Name	LobosTest
OID	
Owner	postgres
Comment	

Retrieving details on databases... Done. 0.00 secs



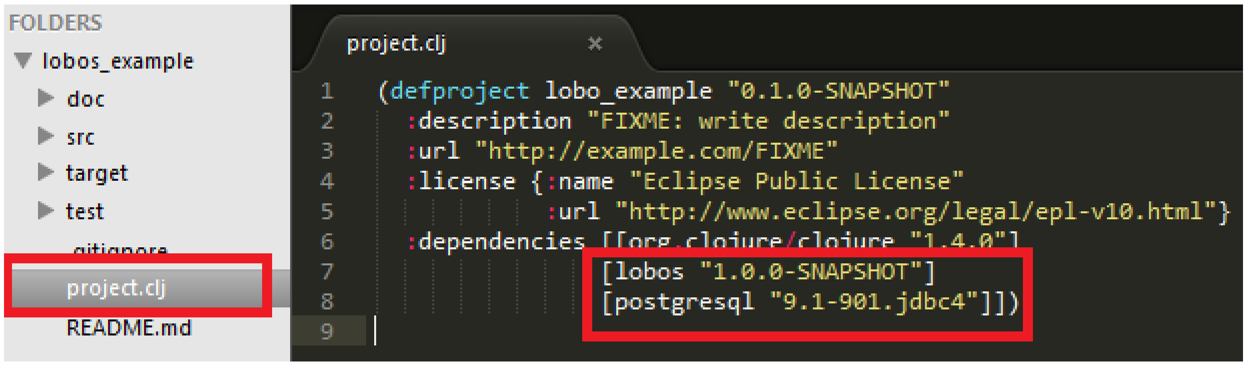
Property	Value
Name	LobosTest
OID	16455
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	UTF8

```
-- Database: "LobosTest"

-- DROP DATABASE "LobosTest";

CREATE DATABASE "LobosTest"
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'English_United States.1252'
       LC_CTYPE = 'English_United States.1252'
       CONNECTION LIMIT = -1;
```

Retrieving details on schema public... Done. 0.08 secs

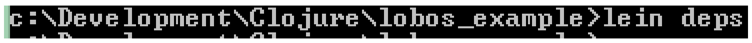
Woo hoo, you now have an empty database. Be proud.

So the next step equally difficult: Updating the project.clj file. Where is such a file? In the root of the project. Crazy.



```
(defproject lobo_example "0.1.0-SNAPSHOT"
  :description "FIXME: write description"
  :url "http://example.com/FIXME"
  :license {:name "Eclipse Public License"
            :url "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.4.0"]
                 [lobos "1.0.0-SNAPSHOT"]
                 [postgresql "9.1-901.jdbc4"]])
```

As you can see, there are now two more dependencies. How do those get resolved?

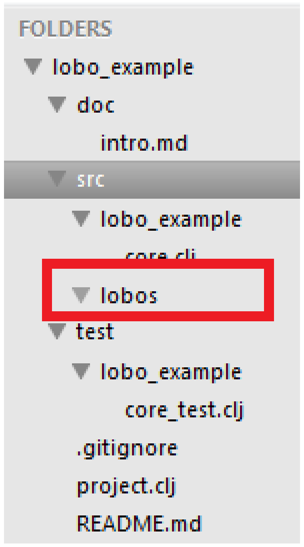


```
c:\Development\Clojure\lobos_example>lein deps
```

Ok so now Lenigen is up to date with what you need.

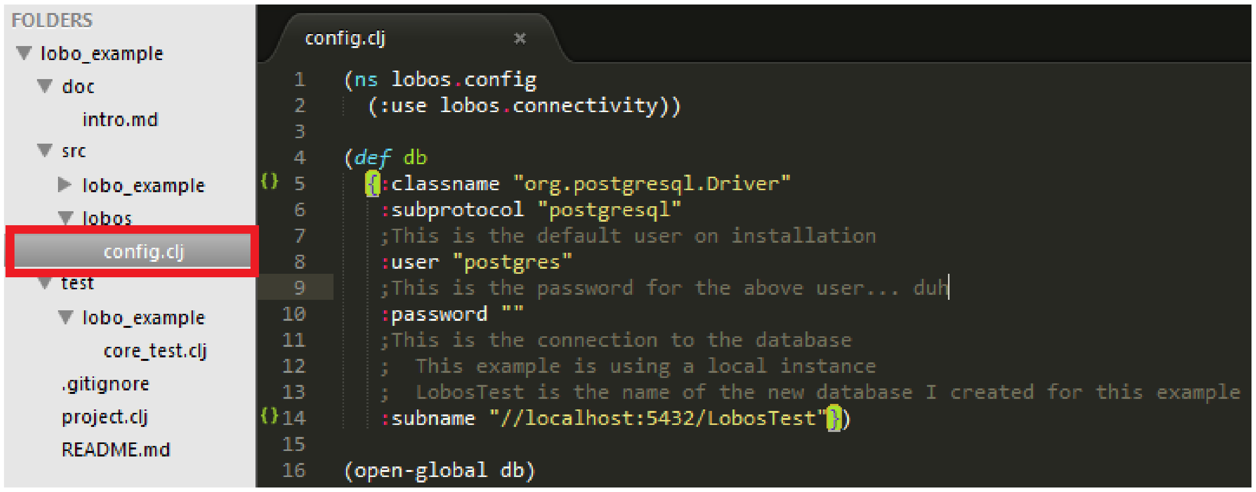
Lobos

First step is to create the Lobos folder:



As you can see, there is nothing special about the folder. Just follows the sort of kinda official folder structure.

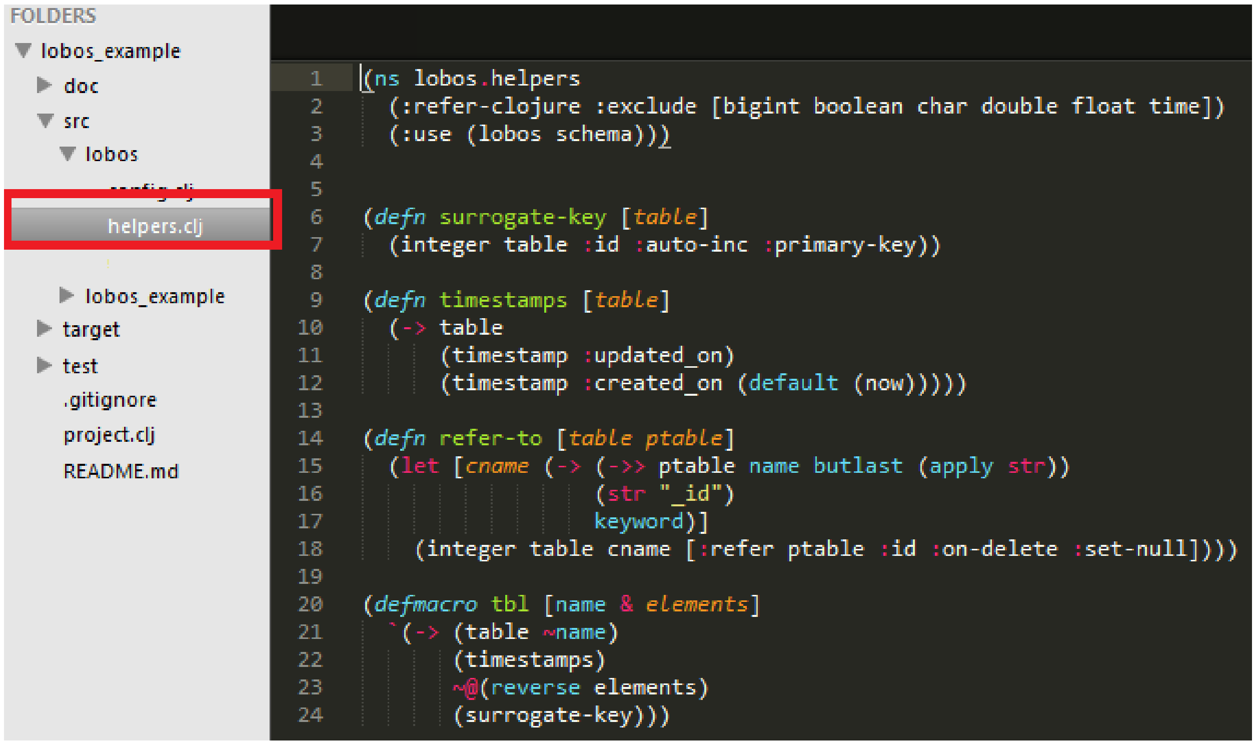
Next create the config file:



```clojure
(ns lobos.config
  (:use lobos.connectivity))

(def db
  {:classname "org.postgresql.Driver"
   :subprotocol "postgresql"
   ;This is the default user on installation
   :user "postgres"
   ;This is the password for the above user... duh
   :password ""
   ;This is the connection to the database
   ;  This example is using a local instance
   ;  LobosTest is the name of the new database I created for this example
   :subname "//localhost:5432/LobosTest"})

(open-global db)
```

Couple things here. This is an example with Postgres, so it will have Postgres information. By default, at least when I set it up, the user is named “postgres”. (Hense why I assigned that user to the new database up top) So as you can see the :user symbol is attached to the “postgres” name. The :subname is the address of the server / database name. Since I named the database “LobosTest”, I think you should be able to figure out where to change if you aren’t using a local instance with a database “LobosTest”.

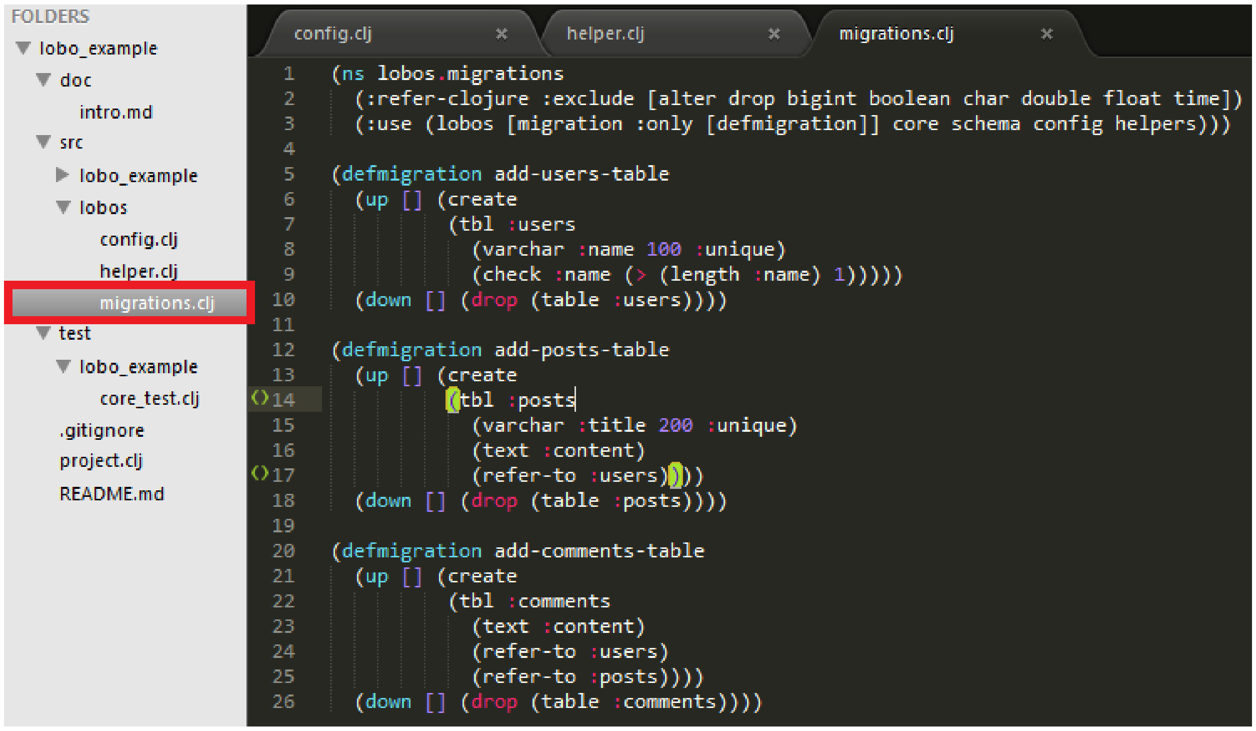
The next step is to create the “helpers” file that was written completely by me using cut and paste. (https://github.com/budu/lobos)



```clojure
(ns lobos.helpers
  (:refer-clojure :exclude [bigint boolean char double float time])
  (:use (lobos schema)))


(defn surrogate-key [table]
  (integer table :id :auto-inc :primary-key))

(defn timestamps [table]
  (-> table
      (timestamp :updated_on)
      (timestamp :created_on (default (now)))))

(defn refer-to [table ptable]
  (let [cname (-> (->> ptable name butlast (apply str))
                  (str "_id")
                  keyword)]
    (integer table cname [:refer ptable :id :on-delete :set-null])))

(defmacro tbl [name & elements]
  `(-> (table ~name)
       (timestamps)
       ~@(reverse elements)
       (surrogate-key)))
```

Just some simple stuff to help with later use. You can get more information from the site I totally didn’t steal this from: https://github.com/budu/lobos

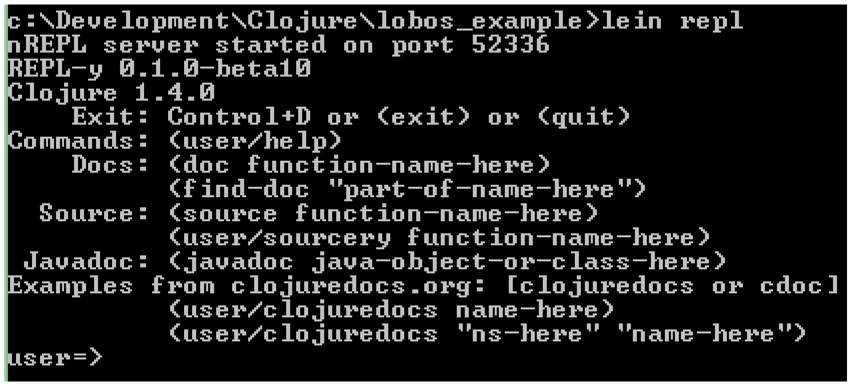
So the final file to create is the migration file.



```clojure
(ns lobos.migrations
  (:refer-clojure :exclude [alter drop bigint boolean char double float time])
  (:use (lobos [migration :only [defmigration]] core schema config helpers)))

(defmigration add-users-table
  (up [] (create
          (tbl :users
            (varchar :name 100 :unique)
            (check :name (> (length :name) 1)))))
  (down [] (drop (table :users))))

(defmigration add-posts-table
  (up [] (create
          (tbl :posts
            (varchar :title 200 :unique)
            (text :content)
            (refer-to :users))))
  (down [] (drop (table :posts))))

(defmigration add-comments-table
  (up [] (create
          (tbl :comments
            (text :content)
            (refer-to :users)
            (refer-to :posts))))
  (down [] (drop (table :comments))))
```

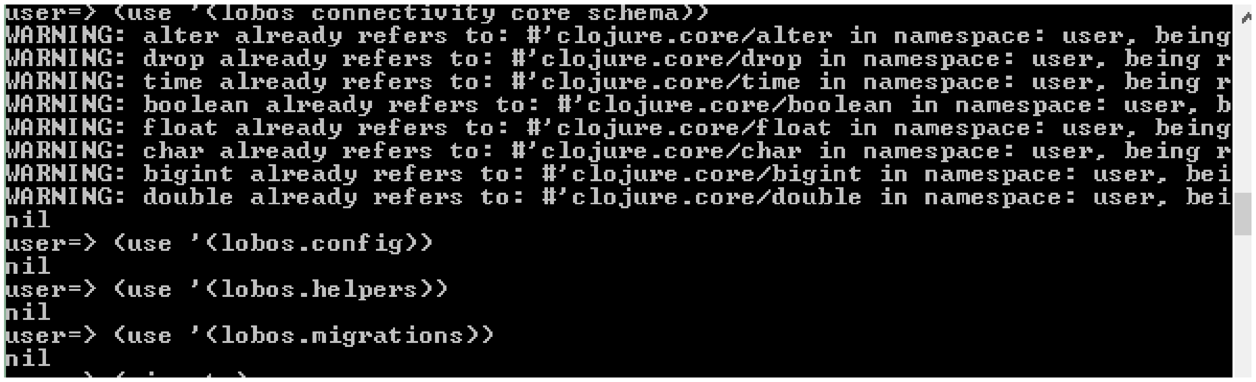
As you can see, there are commands for both creating, and dropping each table. Pretty easy to figure out how to create new tables if you need to.

To get this show on the road, you'll have to fire up an instance of repl with leinigin from the root directory:



```
c:\Development\Clojure\lobos_example>lein repl
nREPL server started on port 52336
REPL-y 0.1.0-beta10
Clojure 1.4.0
    Exit: Control+D or (exit) or (quit)
Commands: (user/help)
    Docs: (doc function-name-here)
          (find-doc "part-of-name-here")
  Source: (source function-name-here)
          (user/sourcery function-name-here)
 Javadoc: (javadoc java-object-or-class-here)
Examples from clojuredocs.org: [clojuredocs or cdoc]
          (user/clojuredocs name-here)
          (user/clojuredocs "ns-here" "name-here")
user=>
```

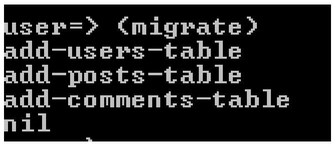
Then you need to include some namespaces for the repl to use.



```
user=> (use '(lobos connectivity core schema))
WARNING: alter already refers to: #'clojure.core/alter in namespace: user, being
WARNING: drop already refers to: #'clojure.core/drop in namespace: user, being r
WARNING: time already refers to: #'clojure.core/time in namespace: user, being r
WARNING: boolean already refers to: #'clojure.core/boolean in namespace: user, b
WARNING: float already refers to: #'clojure.core/float in namespace: user, being
WARNING: char already refers to: #'clojure.core/char in namespace: user, being r
WARNING: bigint already refers to: #'clojure.core/bigint in namespace: user, bei
WARNING: double already refers to: #'clojure.core/double in namespace: user, bei
nil
user=> (use '(lobos.config))
nil
user=> (use '(lobos.helpers))
nil
user=> (use '(lobos.migrations))
nil
```

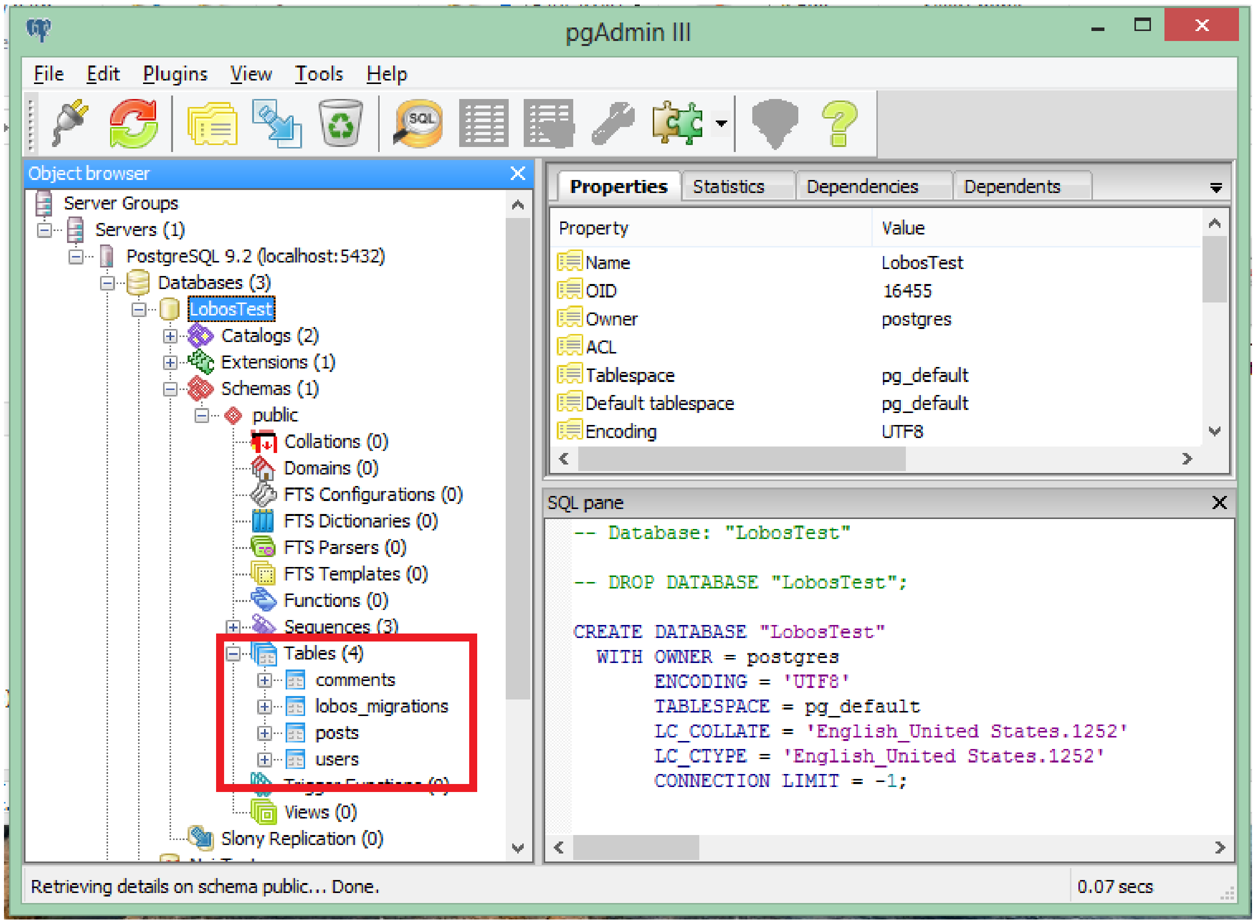
Basically four statements, that probably can be condensed into one, but I wanted to be over thorough… and I was scared to try combining them.

Next up: Migrate



```
user=> (migrate)
add-users-table
add-posts-table
add-comments-table
nil
```

No errors = Happy… or should I say = No errors Happy… Basically this just build the tables you need. If you go back to the PostgreSql manager helper thing, you should see the tables now:



Property	Value
Name	LobosTest
OID	16455
Owner	postgres
ACL	
Tablespace	pg_default
Default tablespace	pg_default
Encoding	UTF8

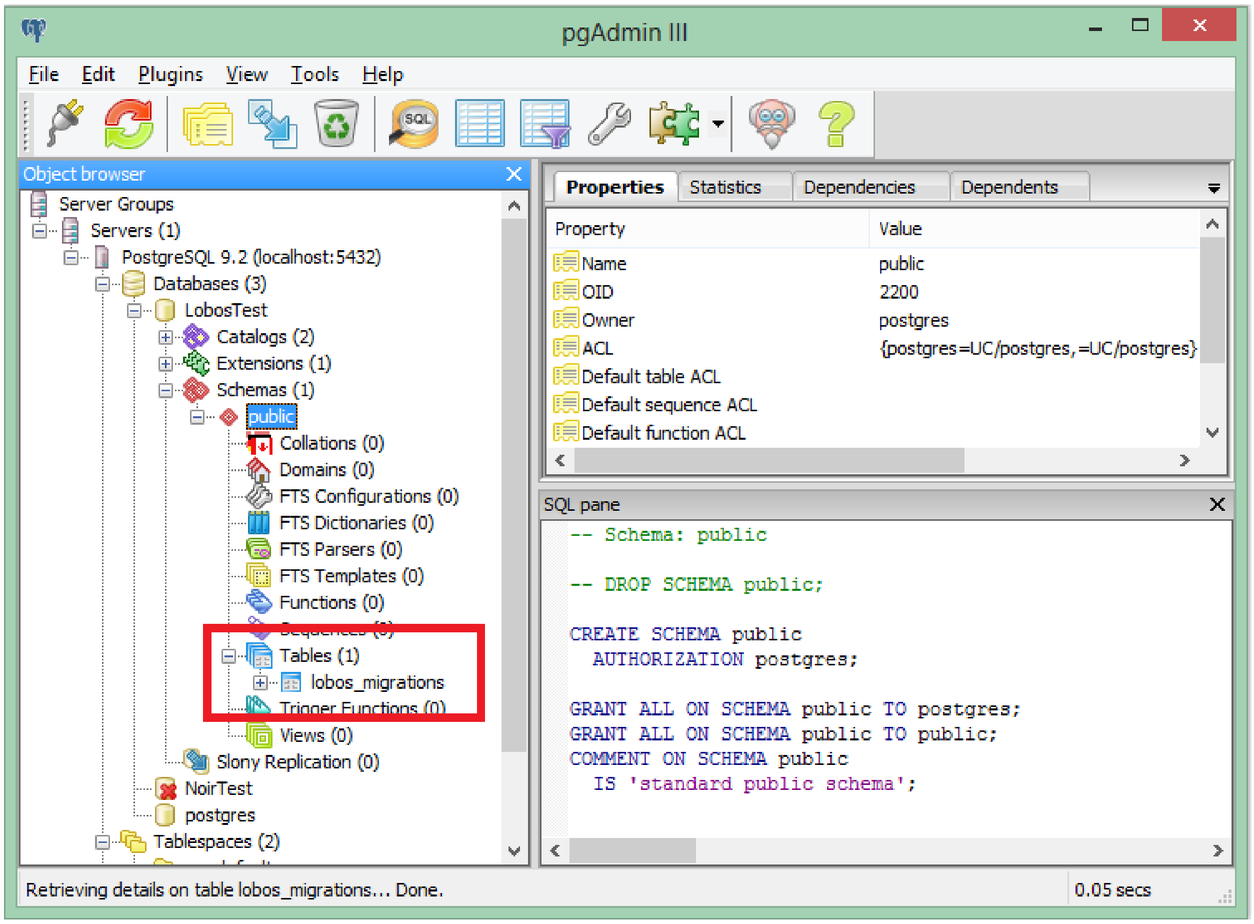
```
-- Database: "LobosTest"

-- DROP DATABASE "LobosTest";

CREATE DATABASE "LobosTest"
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'English_United States.1252'
       LC_CTYPE = 'English_United States.1252'
       CONNECTION LIMIT = -1;
```

And there you go. Lobos integration with PostgreSql.

For a side note, you can rollback each migration from the REPL pretty easily:

```
user=> (rollback)
add-comments-table
nil
user=> (rollback)
add-posts-table
nil
user=> (rollback)
add-users-table
nil
user=> _
```

Now if you look in the management doodad, you'll see the tables are gone:



Property	Value
Name	public
OID	2200
Owner	postgres
ACL	{postgres=UC/postgres,=UC/postgres}
Default table ACL	
Default sequence ACL	
Default function ACL	

```
-- Schema: public

-- DROP SCHEMA public;

CREATE SCHEMA public
  AUTHORIZATION postgres;

GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO public;
COMMENT ON SCHEMA public
  IS 'standard public schema';
```

Retrieving details on table lobos_migrations... Done. 0.05 secs

See, wasn't that easy?