In case you read **my recent tweet** and don't know what I'm talking about: Let me break it down for you.

In my years of experience in the development field of work being a systems administrator/developer hybrid the one thing I learned the hardest was: *Think big*!

This means, that you'll have to not only think about the present uses of your script, but also the future. And, believe me: The future is so uncertain, it will certainly kick you in the ass!

Let's say, you have a script, that searches something in a ldap server. You could do something like this:

```
#!/bin/bash

ldapsearch -x -h ldapserver.company.com -b "dc=company,dc=com" "uid=foobar" >
/tmp/ldif
```

Yeah. That's enough. Now you got the entry „uid=foobar" LDIF-formatted in a file /tmp/ldif. Great. You *could* have made it in a more common and extensible language like PHP or python, but, hey, it works.

Okay. Now, two months later, you need the script again. After you *found* the script (that's most of the time, really) you go through it and – as it's fairly simple – adjust it to your needs. Say, we need the following new features:

* Not only foobar, but foobaz is also searched
* Only output cn and sn

Our script now:

```
#!/bin/bash

ldapsearch -x -h ldapserver.company.com -b "dc=company,dc=com"
"(|(uid=foobar)(uid=foobaz))" "cn,sn" > /tmp/ldif
```

Good. Now, let another year go by, where new things are added or removed. Your script grew to five lines and is kinda hard to read already.

But suddenly someone comes up and says: „Hey, why not reformat the ldif and put it in a database?". You say „great idea", already thinking about wether to stay bashy or go with a *real* (sorry, bash-people) programming language.

You decide to switch languages, remove the script and wrap up something in PHP like this:

```
$ldap_conn = ldap_connect("ldap://ldapserver.company.com");

ldap_bind($ldap_conn);

$result = ldap_search($ldap_conn, "(|(uid=foobar)(uid=foobaz))", array("cn", "sn"));

$my_conn = mysql_connect(...
```

etc,etc,etc. Your script already is run through a cronjob every night and people are relying on the database for -eh, printing out labels.

Some weeks go by where new fields are added or removed and so on.

Then your database admin comes to you and says: „We don't like MySQL anymore, we go with PostgreSQL".

You groan and exchange your mysql-specific-code with the postgresql-code.

A year later, the database admin quits and another one comes, that **hates** OpenSource (I'm completely making this up, you know) and runs to Oracle.

Now is the moment when you think about database independence. And, because you're a smart one, make configuration files for it.

Two weeks later you ask something about oracle in a mailing list and someone says, that she would really like to have your script, because it's exactly what she needs. You send it to her.

Some days go by and suddenly thirty different people use your script. You think about releasing it as open source. You do so.

And again, some days pass, and suddenly you got a vibrant community of developers all working on your script and already google and facebook's using it internally (though they don't admit it).

At **that** point you ask yourself, why you didn't think of that moment in the first place. Now you want your script to comply to some coding standard or you have to rewrite it, because of its history it lacks a few things and is insecure in some other places. And you count your code and it has grown up to several files and thousand lines of code.

That’s my point. Yeah, you shouldn’t set up a github repository first thing when you create a cleanup-script for your temporary directory, but never, never ever do something without thinking about it. Don’t use static parameters, use configuration variables on top of your script or even parse configuration files. Don’t use database-dependent things, other than when you really need special features from certain databases.

Because when you start your work with this, these things aren’t hard to develop. Use libraries or frameworks for complex things. You will thank yourself afterwards.

Thought, I’ll let you know.