Managing projects from the command line: basics
Let us create a small tutorial data set with an accompanying project.

We will use as template a ribosome in a 32x32x32 cube

```
>> dtutorial trib -p prib -template ribosome32.em -tight on;
```

Notice the template assigned to the project:

```
>> ddb prib:template -j;
```

which is a lowpass of the template in the tutorial:

```
>> dslices trib/original_template.em x|y|z -j *
```
open the generated project with `dynamo_project_manager` (or its shortform `dpm`)

>> dpm prib

The command `dtutorial` creates numerical parameters that are more suitable for the default thermosome template.

Besides, in this tutorial we want to produce more iterations than just two, each in one round.
Obviously you can make the changes in the GUI (that's what it is for!).

But notice the alternate way of passing parameters into a project by the command line

```plaintext
>> dvput prib disk -inround 1 ite 2 -cr 360 -cs 60 -ir 360 -is 60;
>> dvput prib disk -inround 2 ite 4 -cr 60 -cs 20 -ir 60 -is 20 -rf 3;
>> dvput prib disk -inround 3 ite 4 -cr 20 -cs 8 -ir 20 -is 8 -rf 6;
```

* The syntax of `dvput` is explained in its documentation (`ddoc dvput`)

* Each parameter is explained with the `dvhelp` command:
  - without arguments lists all project parameters
  - with a parameter name as argument, it will look for specific help on that parameter:

```plaintext
* Parameter: "cone_range"
The first two Euler angles are used to define the orientation of the vertical axis of the protein.
First Euler angle (tdrot) rotates the template around its z axis.
Second Euler angle (tilt) rotates the template around its x axis.
Dynamo scans for this axis inside a cone: The "cone_range" parameter defines the angular aperture of this cone.
360 degrees is thus the value for a global scan.
To skip the part of the angular search that looks for orientations, you have to set
1) "cone range" to zero, and
2) "cone_sampling" to 1.
```
Now, if you load the project against from the GUI dynamo_project_manager:

... the GUI updates, as the numerical scheme for the iterations will have changed...
We can make sure that the project does not need a lot of computation time:

```
>> dvtiming prib
```

so we can unfold and run the project in the usual way:

```
>> dvunfold prib
>> prib
```

When running Dynamo from a shell you need to execute the produced execution script (with extension .exe,.bat) or submit it to a queuing system (with extension .sh)

We can make sure that the project does not need a lot of computation time:

```
>> dvtiming prib
```

when running Dynamo from a shell you need to execute the produced execution script (with extension .exe,.bat) or submit it to a queuing system (with extension .sh)

... and after completion we can check the actually used computation time:

```
>> dvtiming_check prib
```

which (in this case) turns out to be quite accurate.

With multicore and MPI runs things won't be so accurate!
You probably know how to retrieve the results from the database using the GUI:

Pressing here for this parameter combination would create a simple depiction: the projections along x,y and of the averages attained in iterations 3 and 8.
but the files can also be located, accessed and operated upon with the database:

1- pick the project
2- query for standard results
3- select items of interest
4 operate on them:
   4a with the Dynamo linker, or
   4b with local tools
... and also with the command line tool for database browsing ddb:

```
>> ddb prib:a:ite=[3,8] -m
```
The `ddb` tool lets you access different elements in a project or set of projects:

For instance:

```
>> ddb prib:a:ite=* -d
```

inside `prib`, look for database items of type `average` (shortcut `{a})`
The syntax is general for anything that has to do with a project:

```
>> prib:eo_fsc:ite=[1:10] -p
```

The syntax and list of “database items” that can be retrieved with `ddb` is in its documentation (`ddoc  ddb`)

Closer information on the database items can be invoked with `dbhelp`

and the tutorial on plugins