

How should I set up a project structure for The Grinder?

Well the short answer is however works best for you. Many people will already know how they want to set up their directory structure and will have no issue implementing The Grinder as one of their many tools. For those looking for a little guidance it is worth asking yourself questions like:

- How many projects will I be working on?
- Will I need to revisit projects from time to time?
- Do I need repeatability?
- Is this a shared implementation?
- ...etc.

Below is given an example of a directory structure for setting up The Grinder.

```
├- Grinder
  |-- bin
  |   |-- setGrinderEnv.sh/cmd
  |   |-- startAgent.sh/cmd
  |   |-- startConsole.sh/cmd
  |   └-- startProxy.sh/cmd
  |-- engine
  |   |-- grinder-3.0-beta32
  |   |-- grinder-3.0
  |   └-- ...
  |-- etc
  |   |-- grinder.properties
  |   └-- ...
  |-- jvm
  |   |-- jdk1.3
  |   |-- jdk1.4.02
  |   └-- ...
  |-- lib
  |   |-- jython2.1
  |   |-- jdom-1.0
  |   |-- xerces_2_6_0
  |   |-- xerces-2_6_2
  |   |-- oracle
  |   └-- ...
  |-- logs
  |   └-- ...
  |-- projects
  |   |-- website_project
  |   |   |-- httpscript.py
  |   |   |-- httpscript_tests.py
  |   |   └-- ...
  |   └-- db_project
```

```

|-- jdbc.py
|-- ...

```

First off the **bin** directory has been created for storing executable files for the implementation. The sample start scripts from "[How do I start The Grinder?](#)" have been included in this directory. The **engine** directory has been created for storing the versions of The Grinder that may be used. Strictly speaking the versions of The Grinder could be stored under the **lib** directory but for this example The Grinder has been given its own directory. The **etc** directory has been created to store the configuration files for the implementation such as the grinder.properties file. The **jvm** directory has been created to store the various jdks and their versions that could be used in testing. The **lib** directory has been created to store the various third party libraries and their respective versions that projects may require. For example if you wanted to use the full set of [libraries](#) ([./g3/scripts.html#jython-installation](#)) which come with jython then this is the directory into which you would install. Remember to update your CLASSPATH with the libraries you require. The **logs** has been created to store the various logs that the grinder generates during its runs. The **projects** directory has been created to store the scripts to be run by The Grinder and organise them by project/body of work.

The above example would be useful as a simple implementation for one person who works on one project at a time. As the number of projects grows, more people share the implementation, or projects need to be revisited with repeatability ensured, then it makes sense, in this example, to modularize the implementation around the projects. To do this simply create the **bin**, **etc** and **logs** directories under the respective projects like so:

```

|-- projects
|-- website_project
|   |-- bin
|   |   |-- setGrinderEnv.sh/cmd
|   |   |-- startAgent.sh/cmd
|   |   |-- startConsole.sh/cmd
|   |   |-- startProxy.sh/cmd
|   |-- etc
|   |   |-- grinder.properties
|   |   |-- ...
|   |-- httpscript.py
|   |-- httpscript_tests.py
|   |-- logs
|   |   |-- ...
|-- ...
|-- db_project

```

Once this has been done the environment can be set to use the engine, JVM and libraries required by a particular project, rather than setting the environment for all the projects (as would happen in the simple implementation). This allows you, for example, to retain projects which were run using legacy versions of libraries and/or engine and re-run them at a later date with the same setup. Also different projects may require different versions of the same library which would have caused issues when using an implementation-wide CLASSPATH. The grinder.properties file can also be customised on a per project basis.

Modularizing the implementation like this gives greater flexibility and repeatability and opens up the prospect of multiple people using the implementation concurrently.