**ListSubComponentsCommand.java**

*ListSubCompontentsCommand* is an implementation of *AdminCommand,* an interface to provide the command *list-sub-components* to *asadmin* utility of GlassFish server. A brief description of this command is given by GlassFish documentation:

*The*list-sub-commands *subcommand lists EJBs or servlets in a deployed module or in a module of a deployed application. If a module is not specified, all modules are listed. The***--appname***option functions only when the specified module is standalone. To display a specific module in an application, you must specify the module name with the***--appname** *option.*

Here you can see the synopsis of the command, to understand how to use parameters:

list-sub-components [--help] [--type *type*] [--appname *appname*] *modulename*

Thanks to the comment in *AdminCommand.java* we understand that an implementation has to be stateless[[1]](#footnote-1) and needs to have a *Scope* of value *PerLookup[[2]](#footnote-2)*. Command parameters are handled using *@Param* signatures.

Three variables are injected:

Deployment deployment;

Application applications;

CommandRunner commandRunner;

How class works?

*execute* method is the core of the command. In fact, this method is executed when the command *asadmin list-sub-components* is called. A *context* is passed from the caller to the *execute* method. *context* is then used to get a reference to an *ActionReport* object.[[3]](#footnote-3)

*modulename* parametersis then used assigned *applicationName* (only if *appname* parameters is not given). *applicationName* is then passed to *VersioningUtils.checkIdentifier(applicationName)* to check if the provided name is valid given certain constrains defined in *VersioningUtils* class. If the name is not valid then an exception is catch, the error is then reported to the caller via *report.setMessage/report.setActionExitCode* and the execution is terminated. Injected *deployment* (an object to handle deployment) is then used to check if an application/module is currently registered. If the application/module is not a life cycle module[[4]](#footnote-4) no further action are performed. An *ApplicationInfo[[5]](#footnote-5)* object is returned from the injected *appRegistry ApplicationRegister[[6]](#footnote-6)* object. If a the returned *ApplicationInfo* is *null* it means that the provided application name reefers to a disabled application.

1. From Oracle documentation: “*A stateless session bean does not maintain a conversational state with the client. When a client invokes the methods of a stateless bean, the bean’s instance variables may contain a state specific to that client but only for the duration of the invocation.”* [↑](#footnote-ref-1)
2. From HK2 documentation: “PerLookup *is the scope for objects that are created every time they are looked up.”* This document will not explain how dependency injection/HK2 framework works. [↑](#footnote-ref-2)
3. From javadocs.com: *An action report is an abstract class allowing any type of server side action like a service execution, a command execution to report on its execution to the originator of the action. Implementations of this interface should provide a good reporting experience based on the user's interface like a browser or a command line shell.* [↑](#footnote-ref-3)
4. From Oracle documentation: A lifecycle module, also called a lifecycle listener module, provides a means of running long or short Java-based tasks within the GlassFish Server environment, such as instantiation of singletons or RMI servers. [↑](#footnote-ref-4)
5. From javadocs.com: “*Information about a running application. Applications are composed of modules. Modules run in an individual container.”*

   [↑](#footnote-ref-5)
6. *ApplicationRegistry* contains references to *ApplicationInfo* objects. Given an identifier *String* it is possible to obtain an associated *ApplicationInfo*. [↑](#footnote-ref-6)