

nuts, the Java Package Manager

<https://github.com/thevpc/nuts> (git repo)

<https://thevpc.github.io/nuts> (website)

nuts.packagemanager@gmail.com

thevpc, 2025-01-04

Plan

1. API
2. Nuts as Library
3. Nuts as a Framework
4. Spring Integration

1. Main Components



2. Nuts as A Library

¥ Simply add nuts to your dependencies

¥ Compatible with java 1.8+

<dependency>

```
Ê <groupId>net. thevpc. nuts</groupId>  
Ê <artifactId>nuts-lib</artifactId>  
Ê <version>0.8.5</version>  
</dependency>
```

3. Nuts as A Library

¥ You can also add runtime to force the runtime version

```
<dependency>  
  <groupId>net.thevpc.nuts</groupId>  
  <artifactId>nuts-runtime</artifactId>  
  <version>0.8.5.0</version>  
</dependency>
```

3.1. Session API

```
Ê // sharedInstance allows NSession/NWorkspace to be accessible globally as a
singleton
Ê Nuts.openWorkspace("-Z", "-S", "y", "--json").setSharedInstance();
Ê // then you can get the current session anywhere in your code
Ê NSession session=NSession.of();
Ê session.setConfirm(NConfirmationMode.ASK);
Ê session.setOutputFormat(NContentType.XML);

Ê session.out().println("Hello");
Ê session.out().printlnf("Hello");

Ê session.out().println(Arrays.asList("Hello"));
Ê session.out().printlnf("Hello %s", "world");
Ê session.out().println(NMsg.ofC("Hello %s", "world"));
Ê session.out().println(NMsg.ofJ("Hello {0}", "world"));
Ê session.out().println(NMsg.ofV("Hello $v", NMaps.of("v", "world"));
```

3.2. Messages and text formatting

Nuts Library allows multiple variants of string interpolation

```
Ê  NSession session=NSession.of();
Ê  session.setConfirm(NConfirmationMode.ASK);
Ê  session.setOutputFormat(NContentType.XML);

Ê  session.out().println("Hello");
Ê  session.out().printlnf("Hello");
Ê  session.out().println(Arrays.asList("Hello"));
Ê  session.out().printlnf("Hello %s", "world");
Ê  session.out().println(NMsg.ofC("Hello %s", "world"));
Ê  session.out().println(NMsg.ofJ("Hello {0}", "world"));
Ê  session.out().println(NMsg.ofV("Hello $v", NMaps.of("v", "world"));
```

3.3. Std In/Out/Err API

```
Ê Nuts.openWorkspace("-Z", "-S", "y", "--json").setSharedInstance();  
Ê NSession session=NSession.of();  
Ê session.out().println("Hello");  
Ê session.out().printlnf("Hello");  
  
Ê session.out().println(Arrays.asList("Hello"));  
Ê session.out().printlnf("Hello");  
  
Ê session.err()....;  
Ê session.in()....;
```


3.4. Find API

```
Ê  NSession session=...;
Ê  NStream<NId> ids=NSearchCmd.of()
Ê    .addId("org.jedit:jedit")
Ê    .setLatest(true)
Ê    .setDistinct(true).getResultIds();
Ê  for(NId id:ids){
Ê    ...
Ê  }
Ê  NStream<NDefinition> defs=NSearchCmd.of()
Ê    .addId("org.jedit:jedit")
Ê    .setLatest(true)
Ê    .setDistinct(true).getResultDefinitions();
Ê  for(NDefinition d:defs){
Ê    session.out().println(d.getInstallInformation()
Ê    .getInstallFolder());
Ê  }
```

3.5. ClassPath API

```
Ê  NSession session=...;  
Ê  ClassLoader loader=NSearchCmd.of()  
Ê    .addId("org.jedit:jedit")  
Ê    .addId("org.springframework:spring-context")  
Ê    .setLatest(true)  
Ê    .setDistinct(true).getResultClassLoader();
```

3.6. NTF API

```
Ê  NSession session=...;  
Ê  session.out().println("#Hello1# ##Hello2## ##:_:Hello3## ");  
Ê  session.out().println("```java public static class MyClass {}```");  
Ê  session.out().println("```js public static class MyClass {}```");  
Ê  session.out().println("```xml <a>hello</a>```");  
Ê  session.out().println("```json {a:'hello'}```");
```

3.7. Format API

```
Ê  NSession session=...;
Ê  class Customer{String id;String name;}
Ê  Customer customer1,customer2,customer3; ...
Ê  //
Ê  session.setOutputFormat(NContentType.JSON).out().println(Arrays.
asList(customer1,customer2,customer3))
Ê  session.setOutputFormat(NContentType.TREE).out().println(Arrays.
asList(customer1,customer2,customer3))
Ê  session.setOutputFormat(NContentType.PLAIN).out().println(Arrays.
asList(customer1,customer2,customer3))
Ê  session.setOutputFormat(NContentType.XML).out().println(Arrays.
asList(customer1,customer2,customer3))
Ê  session.setOutputFormat(NContentType.PROPS).out().println(Arrays.
asList(customer1,customer2,customer3))
Ê  session.out().println(Arrays.asList(customer1,customer2,customer3))
```

3.8. Format API

```
Ê  NSession session=...;  
Ê  Object a,b,c,d; ...  
Ê  NMutableTableModel m = NMutableTableModel.of();  
Ê  m.newRow().addCells(a,b,c,d);  
Ê  session.out().printlnf(m);
```

3.9. Exec API

```
Ê  NSession session=Nuts.openWorkspace("-Z", "-S");  
Ê  int code=NExecCmd.of().addCommand("ls", "-l").getResult();  
Ê  String out=NExecCmd.of().addCommand("nsh", "ls", "--table")  
Ê    .grabOutputString()  
Ê    .getOutputString();
```

3.10. IO API

```
Ê  NCP.of()
Ê    .from("http://my-server.com/file.pdf")
Ê    .to("/home/my-file")
Ê    .setProgressMonitor(true)
Ê    .setValidator((in)->checkSHA1Hash(in))
Ê    .run();

Ê  NPS ps=NPS.of()
Ê  if(ps.isSupportedKillProcess()){
Ê    ps.killProcess("1234");
Ê  }
```

4. Nuts as a Framework

¥ Nuts Application Framework

- ! Add support for Base Directory API
 - " API to manage per application directories (log, cache, config,É)
- ! Add support for Base Commandline API
 - " standardized commandline options
 - " inherit common options (--table, --json, É)

5. Nuts as a Framework

- ¥ Add support for Application Lifecycle (Hooks for install, update, uninstall)
- ¥ Add support for auto update
- ¥ Add support for isolated input/output (via session in/out)
- ¥ Add support for Desktop Integration
 - ! Add Shortcuts, Menus
 - ! Add Aliases

6. Nuts Application Framework

¥ Implement NApplication

¥ Add Description Properties in pom.xml

7. NAF Example

```
public class Main implements NApplication {  
    public static void main(String[] args) {  
        new Main().runAndExit(args);  
    }  
    @Override  
    public void run() {  
        NCmdLine cmd=NApp.of().getCmdLine();  
        ...  
    }  
}
```

8. NAF Example

```
public class Main implements NApplication {
    public static void main(String[] args) {new Main().runAndExit(args);}
    @Override
    public void run() {
        NCmdLine cmd=NAApp.of().getCmdLine();
        ...
    }
    @Override
    public void onInstallApplication() {}
    @Override
    public void onUpdateApplication() {}
    @Override
    public void onUninstallApplication() {}
}
```

9. NAF + Spring

```
@SpringBootApplication
@Import(NutsSpringBootConfig.class)
public class AppExample implements NApplication {
    public static void main(String[] args) {
        SpringApplication.run(AppExample.class, args);
    }

    @Override
    public void run() {
        NPrintStream out = NSession.of().out();
        out.println("Hello ##World##");
    }
}
```

10. NAF + Spring

while adding the following maven dependency

```
Ê      <dependency>  
Ê          <groupId>net.thevpc.nuts</groupId>  
Ê          <artifactId>nlib-spring-boot</artifactId>  
Ê          <version>0.8.5.0</version>  
Ê      </dependency>
```

10.1. Conclusion

- ¥ **nuts** can be used as a library or as a framework
- ¥ Using **nuts** provides many valuable features
- ¥ I invite you to
 - ! Take a shot, try to use it and give feedback
 - ! **Star(*)** the repository <https://github.com/thevpc/nuts>
 - ! Spread the word
 - ! Join the Core Team to enhance **nuts**

Thank you

please support us by starring our repo at

<https://github.com/thevpc/nuts> (git repo)

<https://thevpc.github.io/nuts> (website)

nuts.packagemanager@gmail.com