TomEE Embedded

TomEE Embedded is based on Tomcat embedded and starts a real TomEE in the launching JVM. It is also able to deploy the classpath as a webapp and to use either `META-INF/resources` or a folder as web resources.

Here is a basic programmatic usage based on `org.apache.tomee.embedded.Container` class:

```
try (final Container container = new Container(new Configuration())
.deployClasspathAsWebApp()) {
    System.out.println("Started on http://localhost:" + container.getConfiguration()
.getHttpPort());

    // do something or wait until the end of the application
}
```

All EE features are then accessible directly in the same JVM.

TomEE Embedded Configuration

The default configuration allows to start tomee without issue but you can desire to customize some of them.

Name	Default	Description
httpPort	8080	http port
stopPort	8005	shutdown port
host	localhost	host
dir	-	where to create a file hierarchy for tomee (conf, temp, ...)
serverXml	-	which server.xml to use
keepServerXmlAsThis	false	don't adjust ports/host from the configuration and keep the ones in server.xml
properties	-	container properties
quickSession	true	use Random instead of SecureRandom (for dev)
skipHttp	false	don't use the http connector
httpsPort	8443	https potr
ssl	false	activate https
withEjbRemote	false	use EJBd

Name	Default	Description
keystoreFile	-	https keystore location
keystorePass	-	https keystore password
keystoreType	JKS	https keystore type
clientAuth	-	https client auth
keyAlias	-	https alias
sslProtocol	-	SSL protocol for https connector
webXml	-	default web.xml to use
loginConfig	-	which LoginConfig to use, relies on `org.apache.tomee.embedded.LoginConfigBuilder` to create it
securityConstraints	-	add some security constraints, use `org.apache.tomee.embedded.SecurityConstaintBuilder` to build them
realm	-	which realm to use (useful to switch to `JAASRealm` for instance) without modifying the application
deployOpenEjbApp	false	should internal openejb application be delpoyed
users	-	a map of user/password
roles	-	a map of role/users
tempDir	${java.io.tmpdir}/tomee-embedded_${timestamp}	tomcat needs a docBase, in case you don't provide one one will be created there
webResourceCached	true	should web resources be cached by tomcat (set false in frontend dev)

Note: passing to `Container` constructor a `Configuration` it will start the container automatically but using `setup(Configuration)` to initialize the configuration you will need to call `start()`.

You can also pass through the properties `connector.xxx` and `connector.attributes.xxx` to customize connector(s) configuration directly.

Standalone applications or TomEE Embedded provided main(String[])

Deploying an application in a server is very nice cause the application is generally small and it allows to update the container without touching the application (typically insanely important in case of security issues for instance).

However sometimes you don't have the choice so TomEE Embedded provides a built-in `main(String[])`. Here are its options:

NOTE	this is still a TomEE so all system properties work (for instance to create a resource).

Name	Default	Description
--path	-	location of application(s) to deploy
--context	-	Context name for applications (same order than paths)
-p or --port	8080	http port
-s or --shutdown	8005	shutdown port
-d or --directory	./.apache-tomee	tomee work directory
-c or --as-war	-	deploy classpath as a war
-b or --doc-base	-	where web resources are for classpath deployment
--renaming	-	for fat war only, is renaming of the context supported
--serverxml	-	the server.xml location
--tomeexml	-	the server.xml location
--property	-	a list of container properties (values follow the format x=y)

Note that since 7.0.0 TomEE provides 3 flavors (qualifier) of tomee-embedded as fat jars:

- uber (where we put all request features by users, this is likely the most complete and the biggest)
- jaxrs: webprofile minus JSF
- jaxws: webprofile plus JAX-WS

These different uber jars are interesting in mainly 2 cases:

- you do a war shade (it avoids to list a bunch of dependencies but still get a customized version)
- you run your application using `--path` option

NOTE	if you already do a custom shade/fatjar this is not really impacting since you can depend on `tomee-embedded` and exclude/include what you want.