Fat / Uber jars - Using the Shade Plugin

Shading the container and the application has some challenges like merging correctly resources (`META-INF/services/` typically).

Here is a maven shade plugin configuration working for most cases:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  <version>2.3</version>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>shade</goal>
      </goals>
      <configuration>
        <dependencyReducedPomLocation>${project.build.directory}/reduced-
pom.xml</dependencyReducedPomLocation>
        <transformers>
          <transformer implementation=
"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
            <mainClass>org.apache.tomee.embedded.FatApp</mainClass>
          </transformer>
          <transformer implementation=
"org.apache.maven.plugins.shade.resource.AppendingTransformer">
            <resource>META-INF/cxf/bus-extensions.txt</resource>
          </transformer>
          <transformer implementation=
"org.apache.openwebbeans.maven.shade.OpenWebBeansPropertiesTransformer" />
        </transformers>
        <filters>
          <filter> <!-- we don't want JSF to be activated -->
            <artifact>*:*</artifact>
            <excludes>
              <exclude>META-INF/faces-config.xml</exclude>
            </excludes>
          </filter>
        </filters>
      </configuration>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.apache.openwebbeans</groupId>
      <artifactId>openwebbeans-maven</artifactId>
      <version>1.7.0/version>
    </dependency>
  </dependencies>
</plugin>
```

NOTE	see TomEE Embedded page for more information about tomee embedded options.
IMPORTANT	this shade uses TomEE Embedded but you can do the same with an Application Composer application.
TIP	if you have `META-INF/web-fragment.xml` in your application you will need to merge them in a single one in the shade. Note that tomcat provides one which can be skipped in this operation since it is there only as a marker for jasper detection.

Then just build the jar:

```
mvn clean package
```

And you can run it:

```
java -jar myapp-1.0-SNAPSHOT.jar
```

Fat Jars with Gradle

With gradle you can rely on either jar plugin, fatjar plugin or shadowjar plugin. Last one is likely the closer to maven shade plugin so that's the one used for next sample:

```
// run $ gradle clean shadowJar
import org.apache.openwebbeans.gradle.shadow.OpenWebBeansPropertiesTransformer

buildscript {
    repositories {
        mavenLocal()
        jcenter()
    }
    dependencies {
        classpath 'com.github.jengelman.gradle.plugins:shadow:1.2.3'
        classpath 'org.apache.openwebbeans:openwebbeans-gradle:1.7.0'
    }
}

apply plugin: 'com.github.johnrengelman.shadow'

group 'org.apache.tomee.demo.gradle'
version '1.0-SNAPSHOT'
```

```
apply plugin: 'idea'
apply plugin: 'java'

sourceCompatibility = 1.8

repositories {
    mavenLocal()
    mavenCentral()
}

dependencies {
    compileOnly 'org.projectlombok:lombok:1.16.10'
    compile 'org.apache.tomee:tomee-embedded:7.0.2-SNAPSHOT'
}

// customize exclusions depending your app

// first the not used dependencies like JSF, JAXWS, JMS ones
def excludedDependenciesGroups = [
        // gradle is buggy with poms, scope provided and optional I think
        'com.google.code.findbugs',
        'com.google.guava',
        'javax.annotation',
        'javax.ws.rs',
        'net.sf.ehcache',
        'org.apache.httpcomponents',
        'org.ow2.asm',
        // tomee jaxws, jms, etc...
        'commons-codec',
        'com.sun.xml.messaging.saaj',
        'joda-time',
        'junit',
        'net.shibboleth.utilities',
        'org.apache.activemq',
        'org.apache.activemq.protobuf',
        'org.apache.myfaces.core',
        'org.apache.neethi',
        'org.apache.santuario',
        'org.apache.ws.xmlschema',
        'org.apache.wss4j',
        'org.bouncycastle',
        'org.cryptacular',
        'org.fusesource.hawtbuf',
        'org.jasypt',
        'org.jvnet.mimepull',
        'org.opensaml',
        'wsdl4j',
        'xml-resolver'
]

// then cxf+tomee specific dependencies so we need to be more precise than the group
```

```
// to not exclude everything
def excludedDependenciesArtifacts = [
        'cxf-rt-bindings-soap',
        'cxf-rt-bindings-xml',
        'cxf-rt-databinding-jaxb',
        'cxf-rt-frontend-jaxws',
        'cxf-rt-frontend-simple',
        'cxf-rt-security-saml',
        'cxf-rt-ws-addr',
        'cxf-rt-wsdl',
        'cxf-rt-ws-policy',
        'cxf-rt-ws-security',
        'openejb-cxf',
        'openejb-webservices',
        'tomee-webservices',
        'geronimo-connector',
        'geronimo-javamail_1.4_mail'
]
shadowJar {
    classifier = 'bundle'

    // merge SPI descriptors
    mergeServiceFiles()
    append 'META-INF/cxf/bus-extensions.txt'
    transform(OpenWebBeansPropertiesTransformer.class)

    // switch off JSF + JMS + JAXWS
    exclude 'META-INF/faces-config.xml'
    dependencies {
        exclude(dependency {
            excludedDependenciesGroups.contains(it.moduleGroup) ||
                    excludedDependenciesArtifacts.contains(it.moduleName)
        })
    }

    // ensure we define the expected Main (if you wrap tomee main use your own class)
    manifest {
        attributes 'Main-Class': 'org.apache.tomee.embedded.FatApp'
    }
}
```

Then run:

```
gradle clean build shadowJar
```

and you'll get `build/libs/demo-gradle-tomee-embedded-shade-1.0-SNAPSHOT-bundle.jar` ready to run with:

```
java -jar build/libs/demo-gradle-tomee-embedded-shade-1.0-SNAPSHOT-bundle.jar --as-war
--simple-log=true
```

Fat Wars

Fat Wars are executable wars. Note they can be fancy for demos but they have the drawback to put the server in web resources at packaging time (to ensure the war is actually an executable jar) so adding a filter preventing these files to be read can be needed if you don't already use a web technology doing it (a servlet bound to /*).

Here how to do a fat war:

```
<properties>
  <!-- can be uber (for all), jaxrs, jaxws for lighter ones -->
  <tomee.flavor>uber</tomee.flavor>
</properties>

<dependencies>
  <!-- ...your dependencies as usual... -->
  <dependency>
    <groupId>org.apache.tomee</groupId>
    <artifactId>tomee-embedded</artifactId>
    <classifier>${tomee.flavor}</classifier>
    <version>7.0.0</version>
    <scope>provided</scope>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-war-plugin</artifactId>
      <version>2.6</version>
      <configuration>
        <failOnMissingWebXml>false</failOnMissingWebXml>
        <archive>
          <manifest>
            <mainClass>org.apache.tomee.embedded.Main</mainClass>
          </manifest>
        </archive>
        <dependentWarExcludes />
        <overlays>
          <overlay>
            <groupId>org.apache.tomee</groupId>
            <artifactId>tomee-embedded</artifactId>
            <classifier>${tomee.flavor}</classifier>
            <type>jar</type>
            <excludes />
          </overlay>
        </overlays>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Then just build the war:

```
mvn clean package
```

And you can run it:

```
java -jar myapp-1.0-SNAPSHOT.war
```