REPORT ON JAR GENERATION

1 DE JUNIO DE 2022

E2.07

https://github.com/anac29/Acme-Toolkits

Jaime Borrego Conde: jaiborcon@alum.us.es

Antonio Campos Gil: antcamgil@alum.us.es

Ana Conde Marrón: anaconmar@alum.us.es

Gonzalo Martínez Fernández: gonmarfer2@alum.us.es

Jaime Moscoso Bernal: jaimosber@alum.us.es

Enrique Muñoz Pérez: enrmunper@alum.us.es

Tabla de contenido

Resumen ejecutivo

El presente documento contiene una explicación detallada acerca de hasta qué punto el equipo de trabajo ha logrado cumplir con el requisito D5.5, es decir, cómo se ha conseguido empacar el proyecto *Acme-Toolkits* en un fichero `.jar`, así como el intento del equipo para incluir el *framework* Acme-Framework en forma de paquete en el proyecto.

El fin del documento es dejar constancia de los pasos que se han seguido para intentar lograr el requisito. Sin embargo, el manejo de estos archivos derivó en numerosos errores y problemas que se detallarán en las siguientes secciones, por ello, a cambio se detallarán las ideas y los conocimientos adquiridos durante la búsqueda de una solución, lo cual quizás sea una de las experiencias más enriquecedoras del proyecto.

Tabla de revisión

Número de revisión	***Fecha***	***Descripción***
1	01/06/2022	Desarrollo del documento

Introducción

Los ficheros `.jar` son los protagonistas de este documento y no presentarlos puede ser visto como acto de descortesía. Esta extensión de ficheros engloba dos funcionalidades distintas. La primera es albergar una aplicación Java que puede ser ejecutada como cualquier otra aplicación. Por otro lado, la segunda función es constituir una biblioteca de archivos y clases, esta será la funcionalidad que se utilizará en este proyecto.

Los archivos con extensión `.jar` se utilizan, por tanto, para facilitar el desarrollo de aplicaciones, permitiendo importar bibliotecas de manera cómoda o aquellos componentes que sean necesarios.

Durante esta asignatura se han implementado varios de ellos permitiendo observar y recoger sus ventajas e inconvenientes, entre los que se destaca, por ejemplo, poder universalizar un proyecto permitiendo su inclusión independientemente de a qué proyecto se dirija.

Una vez se han presentado las características de este archivo se comentará a continuación cómo se trabajó con este tipo de ficheros durante el transcurso del proyecto, indagando en todos aquellos detalles que abarcan desde la implementación hasta la generación de ficheros `.jar` propios.

Pasos previos

Como muestra de la implicación del equipo en la realización de este requisito, se presentará a continuación el procedimiento seguido para la inclusión de una librería para la detección de *spam* dentro del proyecto *Acme-Toolkits*.

En primera instancia, se creó un proyecto *Maven* por defecto en el que se introdujo una clase llamada *SpamDetector* y cuya función sería almacenar los datos relativos al spam que el sistema contiene, así como procesar cadenas de texto para averiguar si para el sistema se trata de una palabra *spam* o no.

El objetivo del proyecto era convertirse en un archivo `.jar` y ser incluido en *Acme-Toolkits* como una librería. Para ello, se ejecutó el comando `Maven install` sobre el proyecto de detección. Tras esto, se generó el fichero deseado, que se trasladó a la carpeta de recursos de *Acme-Toolkits*, donde pasaría a alojarse.

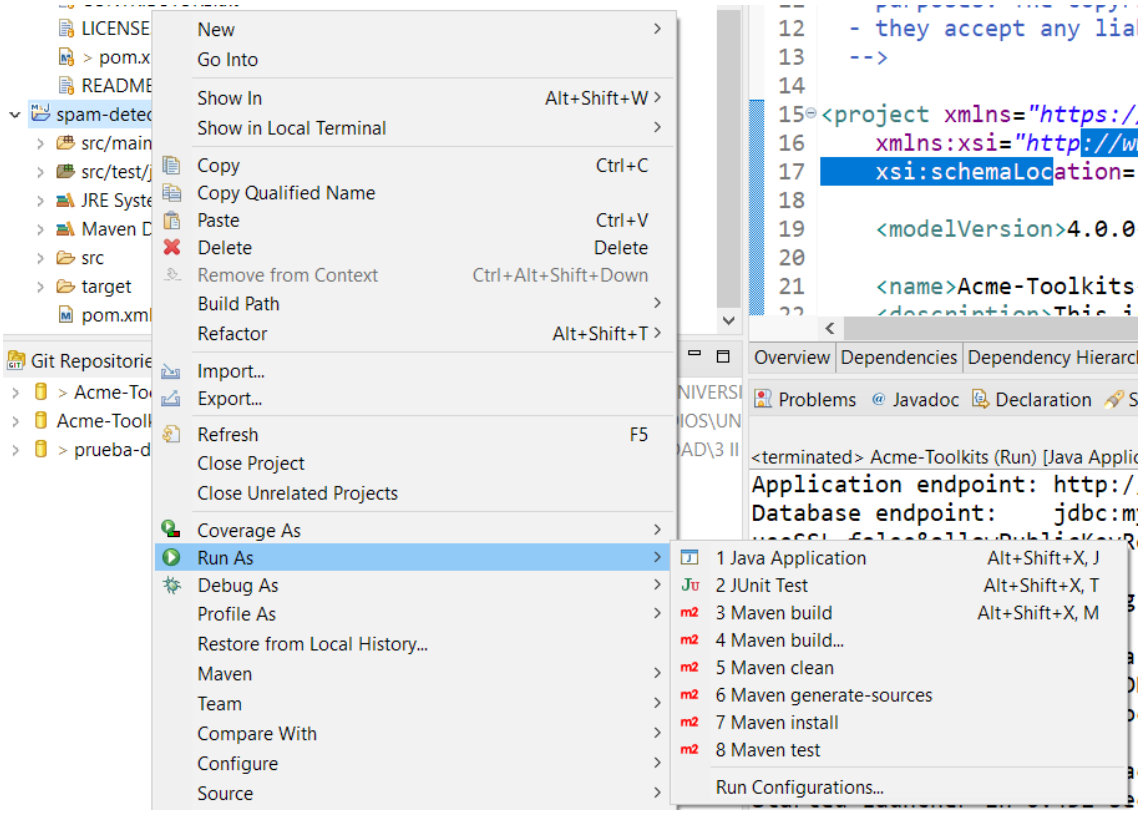

Ilustración 1. Se aplica el comando Maven install sobre spam-detector

Una vez ubicado en un lugar apropiado (a ojos del equipo), se modificó el *Build Path* del proyecto principal para incluir entre sus rutas la del nuevo fichero `.jar`. Posteriormente, se hizo una limpieza de proyecto y a partir de ese momento se pudo comenzar a usar *SpamDetector* como una clase más del proyecto.

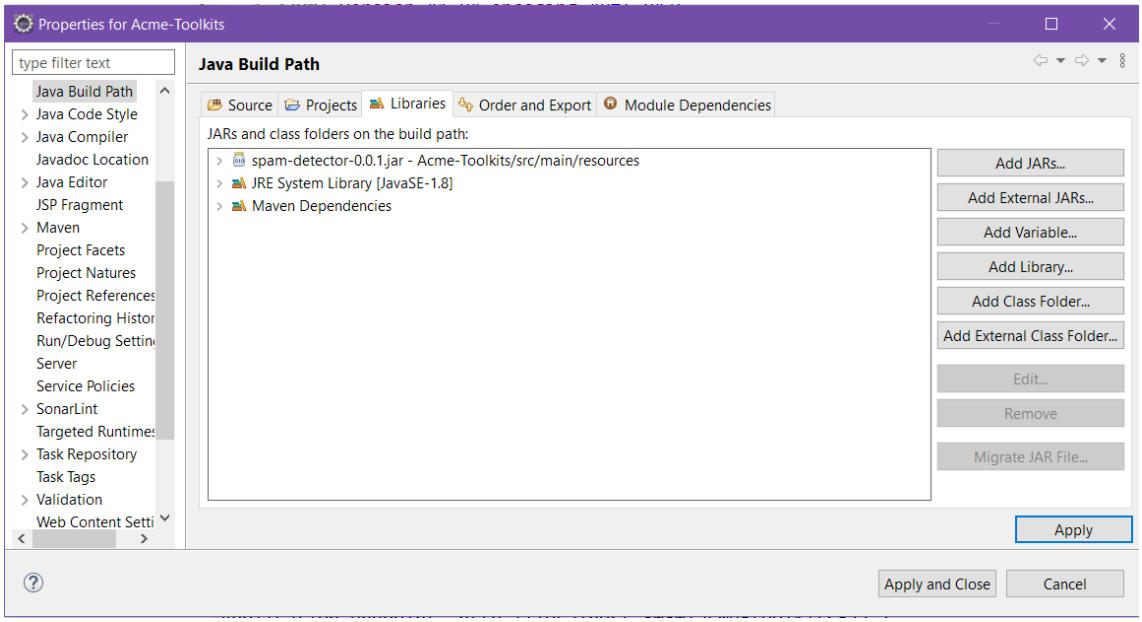

Ilustración 2. El proyecto spam-detector se incluye en la BuildPath de Acme-Toolkits

Primer paso: repetición

Para la compleción del requisito era necesaria, en primer lugar, la generación de un fichero .jar para el proyecto. Por tanto, se modificó el contenido del fichero pom.xml del proyecto, en una suerte de intento por obtener resultados adecuados. El primer detalle que saltaba a la vista era la forma de empaquetado del proyecto, pues en el archivo se especificaba la que la generación deseada era de un fichero .war en vez de uno jar. Por ello, se modificó esa línea tal y como se aprecia en la siguiente captura de pantalla.

```
<groupId>Acme-Toolkits</groupId>
<artifactId>Acme-Toolkits</artifactId>
<version>22.1</version>
<packaging>jar</packaging>
```

Ilustración 3. La etiqueta packaging del fichero pom.xml se modificó

Con esta modificación se pensaba que los cambios necesarios estaban completos, pero se aprovechó la situación para ver en detalle el contenido del archivo `pom.xml` y comprobar si algo más no cuadraba. Efectivamente, en las etiquetas de recursos se encontró algo que posiblemente provocase un comportamiento extraño, pues un directorio se indicaba con una «/» final y otro no. Además, no se incluían los test entre los recursos.

```
<resources>
    <resource>
        <directory>./src/main/webapp/</directory>
        <targetPath>./</targetPath>
    </resource>
    <resource>
        <directory>./src/main/resources</directory>
        <targetPath>./</targetPath>
    </resource>
</resources>
```

Ilustración 4. Recursos en el fichero pom.xml

Una vez se completó la inspección del archivo, se procedió con la ejecución del primer comando `Maven install`. Sin embargo, algo no estaba bien, pues causaba un error debido a los repositorios. Sin modificar nada, se prosiguió con la ejecución del mismo comando. El segundo solo presentó errores sobre la clase *TestHarness*.

Esto dio la posible idea de que faltaba algo relacionado con los test… en los recursos del archivo `pom.xml`.

```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< Acme-Toolkits:Acme-Toolkits >---------------------
[INFO] Building Acme-Toolkits 22.1
[INFO] --------------------------------[ war ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ Acme-Toolkits ---
[INFO] Using 'utf-8' encoding to copy filtered resources.
[INFO] Using 'utf-8' encoding to copy filtered properties files.
[INFO] Copying 117 resources to ./
[INFO] Copying 9 resources to ./
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ Acme-Toolkits ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 120 source files to C:\Users\gonza\Documents\ESTUDIOS\UNIVERSIDAD\3 II SOFTWARE\DP2\Workspace\Workspace-22.0\Projects\Acme-Toolkits\target\classes
[INFO] -------------------------------------------------------------
[ERROR] COMPILATION ERROR :
[INFO] -------------------------------------------------------------
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-
Toolkits/src/main/java/acme/features/administrator/announcement/AdministratorAnnouncementRepository.java:[3,35] package acme.framework.repositories does not exist
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-
Toolkits/src/main/java/acme/features/administrator/announcement/AdministratorAnnouncementRepository.java:[5,62] cannot find symbol
  symbol: class AbstractRepository
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-
Toolkits/src/main/java/acme/features/authenticated/announcement/AuthenticatedAnnouncementRepository.java:[10,35] package acme.framework.repositories does not exist
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-
Toolkits/src/main/java/acme/features/authenticated/announcement/AuthenticatedAnnouncementRepository.java:[13,62] cannot find symbol
  symbol: class AbstractRepository

[INFO] 2 errors
[INFO] -------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.586 s
[INFO] Finished at: 2022-06-01T19:17:15+02:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.8.1:testCompile (default-testCompile) on project Acme-Toolkits: Compilation failure:
Compilation failure:
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-Toolkits/src/test/java/acme/testing/TestHarness.java:
[17,30] package acme.framework.testing does not exist
[ERROR] /C:/Users/gonza/Documents/ESTUDIOS/UNIVERSIDAD/3 II SOFTWARE/DP2/Workspace/Workspace-22.0/Projects/Acme-Toolkits/src/test/java/acme/testing/TestHarness.java:
[19,43] cannot find symbol
[ERROR]   symbol: class AbstractTest
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Ilustración 5. Errores mostrados en consola tras la ejecución de Maven install

Acto seguido, se incluyeron las carpetas de test dentro de la etiqueta de recursos del fichero `pom.xml` y se ejecutó de nuevo el comando `Maven install`. Para sorpresa de todos, en esta ocasión el proyecto compiló correctamente y se generó el deseado fichero `.jar`.

```
<resources>
    <resource>
        <directory>./src/main/webapp/</directory>
        <targetPath>./</targetPath>
    </resource>
    <resource>
        <directory>./src/main/resources</directory>
        <targetPath>./</targetPath>
    </resource>
    <resource>
        <directory>./src/test/java</directory>
        <targetPath>./</targetPath>
    </resource>
    <resource>
        <directory>./src/test/resources</directory>
        <targetPath>./</targetPath>
    </resource>
</resources>
```

Ilustración 6. Nuevas modificaciones sobre pom.xml

Sin embargo, el equipo cree que no es un archivo que realmente funcione de manera correcta, principalmente, porque al intentar ejecutar la aplicación mediante `java -jar Acme-Toolkits-22.1.jar`, no se encuentra la clase *Launcher* (que se ubica en el *framework* y no se alcanza).

Segundo paso: errores

Tras completar con relativo éxito la tarea de generación de un archivo `.jar` para el proyecto *Acme-Toolkits*, se continuó con la generación de un archivo `.jar` para el *framework* del proyecto, con la intención de utilizarlo como sustituto para el propio proyecto del *framework*.

En primera instancia, se comprobó que el fichero `pom.xml` del *framework* poseyese las características mencionadas en la anterior sección. Tras asegurar esos aspectos, se procedió con la ejecución del comando `Maven install`. En esta ocasión, se generó el archivo al primer intento sin más problemas.

No obstante, una vez se introdujo en el proyecto *Acme-Toolkits* tal y como se hizo con el proyecto *spam-detector*, aparecieron una serie de errores y advertencias. Los errores eran todos relativos al método `acme:anyOf()`, ilocalizable. Las advertencias también expresaban la imposibilidad de encontrar etiquetas *jsp* en la *BuildPath*, pero permitían el funcionamiento de la aplicación, a diferencia de los errores.

Realizando una limpieza de proyecto, una mejora de proyecto *Maven* (*Maven Update*, requerido por *Eclipse*) y añadiendo el archivo `.jar` como dependencia del proyecto *Acme-Toolkits* (en `pom.xml`), el proyecto dejó de presentar errores.

En ese instante, era posible ejecutar la población de la base de datos, así como ejecutar la aplicación. Para el pesar del equipo, el funcionamiento resultó ser una falsa ilusión, pues al intentar acceder a la página principal del proyecto, se generó un error 500 causado por la imposibilidad del *servlet* por encontrar las vistas del proyecto.

Tercer paso: investigación

Anteriormente, se mencionó que al repetir el proceso que había sido exitoso con el detector de palabras *spam* se había obtenido un error pues no se podían alcanzar las vistas incluidas en el `.jar`. Para resolver este error se intentaron los puntos que se enumeran a continuación:

1. En primer lugar, se modificó el fichero `pom.xml` del *framework* para que *Maven* generase un fichero `.war` en vez de fichero `.jar`, tal y como se hizo en el sentido contrario para *Acme-Toolkits*, porque se vio que la extensión `.war` tenía como finalidad las aplicaciones web, que en cierto modo daba la sensación de que podría solucionar el problema, sumado a que el proyecto *Acme-Toolkits* generaba un fichero `.war` al principio.

2. Se modificó la línea `spring.config.import = classpath:acme.properties`, intentando especificar la ruta en la que se localizaba el fichero `.jar` para así intentar extraer de manera exitosa las vistas del `.jar` del *framework*.

3. En última instancia, se intentó, además de añadir el archivo `.jar` del *framework* como librería del proyecto *Acme-Toolkits*, se añadió la ruta del fichero como recurso en la pestaña *Source* del *BuildPath*, pero sin éxito alguno.

Tras numerosos intentos infructuosos, el equipo ha decidido que es necesario focalizar el esfuerzo principal del equipo en entregar el resto del incremento.

Conclusiones

Los ficheros `.jar` son herramientas que han resultado ser de gran utilidad pero que han acabado siendo excesivamente complejos para el limitado tiempo del que se disponía. Gracias a ello, generamos un detector de spam que se puede usar en cualquier proyecto independientemente de su contenido y funcionalidad. Esta propiedad es quizás elemental en un mundo como el de la ingeniería informática donde continuamente es necesario combinar las funcionalidades de diferentes librerías y componentes para producir proyectos con nuevas y distintas características.

Empero, no es oro todo lo que reluce y una vez más el equipo de desarrollo advirtió la complejidad que conlleva solucionar un problema en el mundo real. Las vistas del *framework* constituyeron un obstáculo inexpugnable que, sumado al limitado tiempo del que se ha dispuesto las últimas semanas, ha impedido investigar la temática en profundidad y conseguir ofrecer una respuesta clara acerca de la posibilidad de generar un `.jar` de nuestro proyecto *Acme-Toolkits* como nos hubiese deseado.

A lo largo del desarrollo del incremento, se ha recopilado información de gran utilidad acerca de cómo funcionan las dependencias del archivo `pom.xml`, el archivo `.classpath` o la inclusión de distintos archivos de propiedades, aunque dejando el sabor agridulce de no haber logrado obtener una respuesta más clara que un "no es posible" en el grupo. Quizás en otras circunstancias, con más tiempo y recursos se hubiese hallado la respuesta.

Se considera por tanto que, aunque no se haya podido terminar el requisito de manera completa, la experiencia ha sido totalmente provechosa y ha permitido indagar más en el funcionamiento de un proyecto *Maven (Java).*

Bibliografía

Intencionalmente en blanco.