Construcción Gradle por entorno

Oficina de Calidad

Versión: 1.0.1

Control de versiones

Versión	Fecha	Autor	Descripción / Comentarios
V01_V00	14-11-2017	Oficina de Calidad	Versión inicial
1.0.1	22-02-2021	Oficina de Calidad	Se actualiza dirección de correo en apartado “RESOLUCIÓN DE DUDAS”

Índice

1. INTRODUCCIÓN

1.1. OBJETIVO

Este documento describe las normas de configuración Gradle que deben seguir los proyectos, desarrollados en el Ministerio del Interior, que tengan la necesidad de generar distintos artefactos en función del entorno de ejecución al que vayan destinados.

1.2. AUDIENCIA

Este documento está dirigido a todas las personas que colaboren en labores relacionadas con la gestión, desarrollo, auditoría, implantación y explotación de los sistemas de información del Área de Desarrollo de la Subsecretaría de MIR.

1.3. ALCANCE

El presente documento trata de forma específica la necesidad de configuración y construcción de artefactos desplegables por entorno mediante la herramienta Gradle, definiendo unas normas base a partir de la cual los equipos de desarrollo extenderán sus funcionalidades de forma análoga, consiguiendo de esta forma que todos los proyectos Gradle contengan una configuración heterogénea y por tanto los procesos de construcción, tanto en entornos locales como de integración continua, se realicen de forma estandarizada y homogénea.

Queda fuera del alcance de este documento, definir la configuración de todas aquellas necesidades particulares de cada uno de los proyectos, teniendo que ser definidas e implementadas por los equipos de desarrollo a partir de las bases propuestas en este documento.

Toda la normativa y los enlaces a la documentación oficial se encuentran asociados a la última versión disponible *current*.

1.4. RESOLUCIÓN DE DUDAS

Para cualquier duda se debe realizar una consulta a la oficina de Calidad del MIR a través del correo electrónico sgsics.calidadsw@interior.es con la etiqueta “[gradle]”.

1.5. GLOSARIO

Término	Descripción
MIR	Ministerio del Interior.
Plugin	Componente de software que añade una funcionalidad concreta a una herramienta de uso más genérico.

1.6. DOCUMENTOS RELACIONADOS

/Ubicación/Documento	Descripción

2. CONFIGURACIÓN GRADLE

Durante los próximos puntos se tratará de forma específica cómo configurar y construir artefactos asociados a entornos de ejecución, utilizando para ello la herramienta Gradle y plugins de extensión. Cada punto expondrá las bases sobre la que se desarrollará la configuración necesaria para poder construir artefactos en función de parámetros Gradle de línea de comandos.

2.1. PARÁMETRO GRADLE ENV

Para poder realizar construcciones Gradle en función del entorno al que vaya destinado, es necesario que durante la construcción se indique el entorno de ejecución mediante un parámetro de línea de comando. Para ello se deberá hacer uso de la inyección de propiedades en línea de comandos a través de la opción -P.

Para obtener más información sobre la aplicación de la opción -P y el paso de propiedades Gradle a través de la línea de comandos, recomendamos la lectura de documentación oficial al respecto publicada en la siguiente web https://docs.gradle.org/current/userguide/build_environment.html#sec:gradle_properties_and_system_properties

Para que la construcción de todos los proyectos desarrollados en el MIR, siga un mismo criterio y contengan una configuración heterogénea, es necesario que el entorno se indique mediante la propiedad env, pudiendo contener un valor dentro del conjunto des, pre y pro. De esta manera la construcción en entornos locales se realizará de la misma forma que en entornos gestionados por el MIR, como puede ser la infraestructura de construcción automatizada basada en Jenkins o la plataforma de Integración Continua.

A continuación, se indica un ejemplo de ejecución Gradle dependiente de entorno, en el caso de necesitar construir artefactos para el entorno preproducción

```
gradle clean build -Penv=pre
```

Ubicado en el raíz del proyecto principal

Como podemos ver en el ejemplo, durante la ejecución Gradle se dispondrá de la propiedad env, cuyo valor será pre, el cual será utilizado por todas las tareas y funcionalidades Gradle destinadas a la generación de artefactos dependientes de entorno.

Para poder determinar la existencia de una propiedad concreta de un proyecto Gradle, es necesario utilizar el objeto project y el método hasProperty, pasándole el nombre de la propiedad, que nos devolverá true en caso de existir o false en caso contrario. Para poder recuperar el valor de una propiedad únicamente es necesario hacer uso del nombre de la propiedad, como veremos en el ejemplo siguiente:

```
println "${project.hasProperty('env') ? env : 'des'}"
```

build.gradle

Como vemos en el ejemplo en caso de existencia de la propiedad env, se imprimirá su valor, en caso contrario se imprimirá el valor determinado por defecto siendo éste des.

2.2. UBICACIÓN DE RECURSOS POR ENTORNO

Para poder utilizar y empaquetar distintos ficheros en función del entorno al que vayan destinados, es necesario que se almacenen en una estructura de carpetas estandarizada y que identifiquen de forma unívoca a cada entorno, permitiendo que la configuración de las tareas y funcionalidades Gradle sean heterogéneas en todos los proyectos desarrollados en el MIR.

Todos los proyectos que necesiten generar ficheros y artefactos por entorno deberán disponer de una estructura de carpetas dentro de la ubicación src/main/environment que permita almacenar aquellos ficheros de recursos y/o configuración dependiente de entorno, que serán utilizados en función de la propiedad que se reciba durante la construcción del proyecto.

A continuación, se indica la estructura de carpetas asociadas a entorno:

```
Root project 'mir-multiproject'
+--- \dao-api'
        +---- build.gradle
+--- \dao-impl
        +---- build.gradle
+--- \war-ws-rest
          +---- \src
              +---- \main
                  +---- \environment
                      +---- \des
                      +---- \pre
                      +---- \pro
          +---- build.gradle
+--- settings.gradle
\--- build.gradle
```

${ROOT_PROJECT}

Dentro de las carpetas des, pre y pro, deberán almacenarse todos aquellos recursos Java, ficheros de configuración de librerías o frameworks y ficheros descriptores de despliegue, que tengan distinto contenido en función del entorno al que vayan destinados. Como ejemplo podemos identificar ficheros de configuración log4j, ficheros descriptores web.xml o archivos asociados a configuración de framework como application-context.xml.

Todos los ficheros dependientes de entorno deberán estar almacenados dentro de esta estructura, pudiendo crear subcarpetas para gestionar de forma sencilla todos los ficheros.

Cada fichero, dependiendo de su tipología, finalidad y ubicación de destino, será tratado por aquella funcionalidad y/o tareas que haya sido creada para tal efecto dentro de la configuración Gradle del proyecto. Éstas serán tratadas por separado en los siguientes puntos del documento.

2.3. FICHEROS DE RECURSOS JAVA

Siguiendo las buenas prácticas, para la estandarización de proyectos Java, en relación a la ubicación de ficheros de recursos, todos los proyectos deberán ubicar aquellos ficheros de recursos Java que no sean dependientes de entorno en la ubicación src/main/resources. Por el contrario, aquellos ficheros que contengan información asociada a entorno, deberán ser ubicados en la ruta src/main/environment/${env} correspondiente, donde ${env} será des, pre o pro.

De esta manera podremos copiar aquellos ficheros de recursos asociados a entornos de ejecución, justo antes de la construcción y empaquetado, alojándolos en la ubicación por defecto definida para recursos Java, siendo src/main/resources. Para ello se deberá utilizar la funcionalidad Copy de la API Gradle, con la que podremos copiar todos los ficheros necesarios desde la ubicación src/main/environment/${env} hasta su destino src/main/resources.

Para obtener más información sobre la ubicación de ficheros Java, recomendamos acceder a la documentación oficial publicada en la web
https://docs.gradle.org/current/userguide/java_plugin.html#sec:java_project_layout

En caso de que el proyecto que contenga ficheros asociados a entorno sea de tipo war, éstos, tras su copiado en la carpeta src/main/resources, tendrán como destino, durante el empaquetado, la ubicación WEB-INF/clases tal y como se indica en la definición de estos artefactos. Para más información acerca del empaquetado de recursos y ficheros de configuración en artefactos de tipo war, recomendamos la lectura de la documentación oficial del plugin war de Gradle, publicado en la web https://docs.gradle.org/current/userguide/war_plugin.html#sec:war_default_settings.

A continuación, se indica un ejemplo de configuración de la funcionalidad Copy en un proyecto de tipo war:

```
Root project 'mir-multiproject'
+--- \dao-api'
        +---- build.gradle
+--- \dao-impl
        +---- build.gradle
+--- \war-ws-rest
            +---- \src
                +---- \main
                    +---- \environment
                        +---- \des
                            +---- \webInf
                            +---- \webXml
                            +---- conf.properties
                            +---- log4j.properties
                        +---- \pre
                            +---- \webInf
                            +---- \webXml
                            +---- conf.properties
                            +---- log4j.properties
                        +---- \pro
                            +---- \webInf
                            +---- \webXml
                            +---- conf.properties
                            +---- log4j.properties
            +---- build.gradle
```

```
+--- settings.gradle
\--- build.gradle
```

${ROOT_PROJECT}

```
copy {
   from "src/main/environment/${project.hasProperty('env') ? env : 'des'}"
   into "src/main/resources"
   include "**/*"
   exclude "webInf"
   exclude "webXml"
}
```

build.gradle

Como vemos en el ejemplo, se copiaran todos aquellos ficheros que existan en la carpeta src/main/environment/${env}, excluyendo las carpetas webInf y webXml, en la ubicación src/main/resources.

Para obtener más información sobre la configuración de la funcionalidad Copy, se puede acceder a la web https://docs.gradle.org/current/dsl/org.gradle.api.tasks.Copy.html

2.4. FICHEROS DE CONFIGURACIÓN WAR

Gradle a través de su plugin war, ofrece la posibilidad de modificar el tratamiento por defecto que se le da a los ficheros de configuración de este tipo de artefactos, y por tanto ésta será la forma con la que conseguiremos incluir dentro del empaquetado war aquellos ficheros de configuración en función del entorno al que vayan destinados.

Para obtener información sobre configuración adicional que se puede realizar sobre la construcción de un proyecto war, se puede acceder a la web oficial del plugin https://docs.gradle.org/current/userguide/war_plugin.html#sec:war_customizing.

A continuación, se indica un ejemplo de configuración por entornos del fichero web.xml y ficheros cuyo destino será la carpeta WEB-INF:

```
Root project 'mir-multiproject'
+--- \dao-api'
        +---- build.gradle
+--- \dao-impl
        +---- build.gradle
+--- \war-ws-rest
           +---- \src
                 +---- \main
                    +---- \environment
                       +---- \des
                          +---- \webInf
                          +---- \webXml
                             +---- web.xml
                          +---- conf.properties
```

```
                        +---- log4j.properties
                    +---- \pre
                        +---- \webInf
                        +---- \webXml
                            +---- web.xml
                        +---- conf.properties
                        +---- log4j.properties
                    +---- \pro
                        +---- \webInf
                        +---- \webXml
                            +---- web.xml
                        +---- conf.properties
                        +---- log4j.properties
            +---- build.gradle
+--- settings.gradle
\--- build.gradle
```

${ROOT_PROJECT}

```
war {
    webInf {
        from "src/main/environment/${project.hasProperty('env') ? env : 'des'}/webInf"
        include "*.*"
    }

    webXml = file("src/main/environment/${project.hasProperty('env') ? env : 'des'}/webXml/web.xml")
}
```

build.gradle

Como se puede ver en el ejemplo, en función del valor de la propiedad env, se copiarán los ficheros de la carpeta webInf y el fichero web.xml existente en la carpeta webXml, almacenados en las carpetas de entorno ubicados en src/main/environment.

Los nombres de artefactos deberán contener el entorno para el cual han sido generados, para el caso de artefactos de tipo war, se deberá realizar a través de la configuración de la propiedad archiveName, tal y como se indica en el siguiente ejemplo:

```
war.archiveName = "${war.baseName}${project.hasProperty('env') ? "-$env" : 'des'}.${war.extension}"
```

build.gradle

2.5. FICHEROS ESTÁTICOS

Aquellos proyectos que contengan contenido estático destinado al despliegue fuera del artefacto war, y que contengan ficheros dependientes de entorno, deberán generar un fichero de tipo comprimido mediante la funcionalidad prediseñada generateStaticFilesZip, después de haber copiado los ficheros dependientes de entorno en su ubicación correspondiente. Esta tarea debe incluirse en un fichero gradle llamado generatStaticFilesZip.gradle, ubicándolo en la carpeta gradle-project-conf.

La configuración y definición de esta tarea necesaria para la generación de ficheros de tipo zip, se detalla a continuación:

- Copiar mediante la tarea Copy de Gradle aquellos ficheros estáticos dependientes de entorno en la carpeta correspondiente para que la tarea generateStaticFilesZip los incluya en el fichero comprimido.
- La tarea generateStaticFilesZip deberá contener una descripción, asignado un grupo del ciclo de vida del plugin Java, la configuración necesaria para crear el comprimido y la asignación de una propiedad con la que poder ser saltada. Esta tarea deberá ser asignada al grupo 'Build' y ser asociada a la propiedad creada con el prefijo skip seguido del nombre de la tarea.
- El nombre indicado en la propiedad archiveName deberá contener el valor de la propiedad env.
- Configurar por medio de los métodos include y exclude aquellos ficheros que deban ser incluidos en función del entorno destino.
- archivado de artefactos: el artefacto generado por la tarea generateStaticFilesZip, deberá ser archivado utilizando para ello su inclusión en la sección artifacts.
- Utilizar el método hasProperty del objeto project para determinar el entorno destino.
- Únicamente deben perfilarse por entornos aquellos ficheros que realmente estén asociados por contenido a entornos de ejecución, los ficheros que no tengan contenido por entorno deberán ubicarse en src/main/webapp o subcarpeta correspondiente.

A continuación, se muestra el contenido del fichero generateStaticFilesZip.gradle, que podrá ser copiado y pegado en cualquier proyecto debido a su parametrización y diseño genérico, debiendo configurarse tanto la ruta de la carpeta raíz, como la ubicación específica de los ficheros a incluir por medio de los parámetros from e include respectivamente.

```
Root project 'mir-multiproject'
+--- \dao-api'
        +---- build.gradle
+--- \dao-impl
        +---- build.gradle
+--- \war-ws-rest
          +---- \src
              +---- \main
                  +---- \environment
                      +---- \des
                          +---- \static
                          +---- \webInf
                          +---- \webXml
                              +---- web.xml
                          +---- conf.properties
                          +---- log4j.properties
                      +---- \pre
                          +---- \static
                          +---- \webInf
                          +---- \webXml
                              +---- web.xml
                          +---- conf.properties
                          +---- log4j.properties
                      +---- \pro
                          +---- \static
                          +---- \webInf
                          +---- \webXml
                              +---- web.xml
```

```
                        +---- conf.properties
                        +---- log4j.properties
            +---- build.gradle
+--- settings.gradle
\--- build.gradle
```

${ROOT_PROJECT}

```
copy {
   from "src/main/environment/${project.hasProperty('env') ? env : 'des'}/static"
   into "src/main/webapp"
   include "**/*"
}
```

${ROOT_PROJECT}/build.gradle

```
task generateStaticFilesZip (type: Zip) {
  group = 'Build'
  description = "[Create-by: ${projectInfo.name}] - Generate static web content ZIP file."
  archiveName = "static-${war.baseName}-${project.hasProperty('env') ? env : 'des'}.zip"
  from "src/main/webapp"
  include "images/*.*"
  include "styles/**/*.*"
}
generateStaticFilesZip.onlyIf { !project.hasProperty('skipGenerateStaticFilesZip') }

artifacts {
  archives generateStaticFilesZip
}
```

${ROOT_PROJECT}/gradle-project-conf/generateStaticFilesZip.gradle

```
//Configuración de construcción
…
war {
    ...
   exclude("images")
   exclude("styles")
    ...
}
...
apply from: "gradle-project-conf/generateStaticFilesZip.gradle"
...
```

${ROOT_PROJECT}/build.gradle

Como se puede ver en el ejemplo, aquellos ficheros estáticos incluidos en el archivo zip a generar, deberán ser excluidos durante la generación del artefacto war.