Proyecto Adalid, Plataforma jee1
Manual de Referencia

Tabla de Contenido

Archivos de propiedades

Archivos bootstrapping.properties

Cada meta-proyecto debe tener un archivo **bootstrapping.properties**. Este archivo contiene las propiedades de arranque necesarias para la generación de proyectos[1]. Cuando se ejecuta el meta-proyecto, se busca este archivo en el directorio de trabajo del usuario (*user working directory*), cuya ruta se obtiene de la propiedad **user.dir** de la plataforma Java. Si el meta-proyecto se ejecuta utilizando NetBeans o eclipse, el directorio de trabajo del usuario es el directorio raíz del meta-proyecto.

Propiedad root.folder

Ruta absoluta del directorio raíz de los subdirectorios donde se almacenan los archivos de los proyectos generados. El valor por omisión de esta propiedad es el valor de la variable **WORKSPACE** (vea la descripción de esta variable más adelante, en la sección **Variables**).

Propiedad velocity.folder

Ruta absoluta del directorio que contiene las macros y plantillas de las plataformas utilizadas por el meta-proyecto. Esta propiedad no tiene valor por omisión y se puede repetir para especificar varios directorios.

Propiedad velocity.properties.file

Ruta absoluta del archivo de propiedades de configuración de Velocity. El valor por omisión de esta propiedad es ***user-working-directory*/velocity.properties**, donde ***user-working-directory*** corresponde al valor de la propiedad **user.dir** de la plataforma Java.

Adalid suministra un archivo de propiedades de configuración de Velocity (archivo **velocity.properties** del directorio **adalid/source/development/resources/velocity**). Es posible utilizar otro archivo de configuración, siempre que éste especifique **adalid.commons.velocity.VelocityFileResourceLoader** en la propiedad **file.resource.loader.class**, y que incluya **macros/global-macros.vm** en la propiedad **velocimacro.library**.

Para mayor información sobre las propiedades de configuración de Velocity, consulte la sección "***Velocity Configuration Keys and Values***" de la Guía del Desarrollador de Velocity.

Propiedad velocity.supplementary.properties.file

Ruta absoluta del archivo de propiedades complementarias de configuración de Velocity. El valor por omisión de esta propiedad es ***velocity-properties-directory*/velocity.supplementary.properties**, donde ***velocity-properties-directory*** corresponde al directorio donde se encuentra el archivo especificado en la propiedad **velocity.properties.file**.

Adalid suministra un archivo de propiedades complementarias de configuración de Velocity (archivo **velocity.supplementary.properties** del directorio **adalid/source/development/resources/velocity**). Es posible utilizar otro archivo de propiedades complementarias de configuración. Se recomienda que ese archivo incluya las propiedades del archivo suministrado por Adalid.

Con propiedades del archivo **velocity.supplementary.properties** se puede especificar una codificación alternativa para plantillas y documentos generados con una extensión específica.

---

[1] Los meta-proyectos sirven para modelar aplicaciones y la ejecución de los meta-proyectos genera proyectos que implementan las aplicaciones modeladas.

La propiedad **input.encoding** del archivo **velocity.properties** especifica la codificación de caracteres (*encoding*) de las plantillas. El valor por omisión es ISO-8859-1. Se puede utilizar una codificación alternativa, como UTF-8.

También se puede utilizar una codificación alternativa para plantillas con una extensión específica. Para ello, agregue la propiedad correspondiente a **velocity.supplementary.properties**. Por ejemplo, agregue **velocity.template.encoding.java=UTF-8** para decodificar las plantillas con extensión **.java** con UTF-8.

Para especificar una codificación alternativa para una plantilla específica, utilice la propiedad template-encoding del correspondiente archivo de propiedades de plantilla.

La propiedad **output.encoding** del archivo **velocity.properties** especifica la codificación de caracteres (*encoding*) de los documentos generados. El valor por omisión es ISO-8859-1. Se puede utilizar una codificación alternativa, como UTF-8.

También se puede utilizar una codificación alternativa para documentos generados con una extensión específica. Para ello, agregue la propiedad correspondiente a **velocity.supplementary.properties**. Por ejemplo, agregue **velocity.document.encoding.java=UTF-8** para codificar los documentos generados con extensión **.java** con UTF-8.

Para especificar una codificación alternativa para un documento específico, utilice la propiedad file-encoding del correspondiente archivo de propiedades de plantilla.

Variables

El valor de las propiedades del archivo **bootstrapping.properties** y el valor de la propiedad **file.resource.loader.path** del archivo **velocity.properties** puede comenzar con una referencia a la variable **WORKSPACE**. El valor de esta variable se determina de la siguiente manera:

- Primero se busca un archivo de nombre **workspace**; la búsqueda comienza en el directorio de trabajo del usuario y continúa, ascendentemente, hasta conseguir el archivo o hasta llegar al directorio raíz de la unidad. El valor de la variable **WORKSPACE** será la ruta del directorio que contiene el archivo.
- Si la búsqueda anterior no da resultado, entonces se procede de la misma manera a buscar un directorio de nombre **workspace**. El valor de la variable **WORKSPACE** será la ruta de tal directorio.
- Por último, si la búsqueda anterior tampoco da resultado, entonces el valor de la variable **WORKSPACE** será la ruta del directorio inicial del usuario (*user home directory*), cuya ruta se obtiene de la propiedad **user.home** de la plataforma Java. En Windows, el directorio inicial del usuario suele ser **C:\Users*usuario***, donde ***usuario*** corresponde al nombre del usuario utilizado para iniciar la sesión de trabajo.

Siguiendo las reglas de Velocity, la notación formal para las referencias a la variable **WORKSPACE** es **${WORKSPACE}** y la notación abreviada es **$WORKSPACE**.

Observaciones

El carácter separador de ruta (*path separator*) que se utiliza en los valores de las propiedades es la barra oblicua (*slash*), aun en ambiente Windows.

Archivo bootstrapping.properties para la plataforma jee1

```
root.folder = $WORKSPACE
velocity.folder = $WORKSPACE/adalid/source/development/resources/velocity
velocity.properties.file = \
    $WORKSPACE/adalid/source/development/resources/velocity/velocity.properties
platforms.folder = \
    $WORKSPACE/adalid/source/development/resources/velocity/platforms
```

Archivo velocity.properties para la plataforma jee1

```
runtime.log.logsystem.class = org.apache.velocity.runtime.log.Log4JLogChute
runtime.log.logsystem.log4j.logger = adalid.commons.velocity.VelocityEngineer
file.resource.loader.description = Custom Velocity File Resource Loader
file.resource.loader.class = \
    adalid.commons.velocity.VelocityFileResourceLoader
file.resource.loader.path = \
    $WORKSPACE/adalid/source/development/resources/velocity
file.resource.loader.cache = true
velocimacro.library = macros/global-macros.vm
velocimacro.library.autoreload = true
velocimacro.permissions.allow.inline = true
velocimacro.permissions.allow.inline.local.scope = true
velocimacro.permissions.allow.inline.to.replace.global = true
directive.parse.max.depth = 15
```

Archivos de propiedades de plataformas

Cada plataforma (o variante de una plataforma) de Adalid se define mediante un archivo de propiedades que tiene el nombre de la plataforma (o de la variante) y la extensión **.properties**. Cuando se genera un proyecto, los archivos de propiedades de las plataformas utilizadas por el meta-proyecto se buscan en el directorio definido mediante la propiedad **platforms.folder** del archivo **bootstrapping.properties** del meta-proyecto.

Propiedades file.resource.loader.path

Ruta de un directorio que contiene archivos de propiedades de plantillas. La ruta es relativa al directorio definido mediante la propiedad **platforms.folder** del archivo **bootstrapping.properties**.

Un mismo archivo puede contener tantas propiedades **file.resource.loader.path** como sea necesario para especificar todos los directorios de archivos de propiedades de plantillas de la plataforma.

Cuando se genera un proyecto, se procesan todos los archivos de propiedades de plantillas que se encuentran en los directorios especificados, en sus subdirectorios y en los directorios intermedios entre el directorio definido mediante la propiedad **platforms.folder** del archivo **bootstrapping.properties** y los directorios especificados.

Propiedades .string

Las propiedades **.string** permiten definir variables cuyo valor es una cadena de caracteres. Los nombres de estas propiedades tienen la forma ***identificador*.string**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el valor que se da a la variable.

Se pueden incluir referencias a variables definidas mediante propiedades **.string** en los valores de las propiedades **do.isolated.delete** y **do.cascaded.delete** del mismo archivo, en los valores de las

propiedades de los archivos de propiedades de plantillas que se procesan al generar el proyecto y en las plantillas especificadas en tales archivos. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Propiedades .class

Las propiedades **.class** permiten definir variables cuyo valor es una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.class**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java.

Las variables definidas mediante propiedades **.class** pueden ser utilizadas en las plantillas especificadas en los archivos de propiedades de plantillas que se procesan al generar el proyecto. Estas variables se utilizan para ejecutar los métodos estáticos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Existe un conjunto de clases para las cuales se definen variables de manera automática, sin necesidad de utilizar propiedades **.class**. El nombre de estas variables es el nombre simple (sin el paquete) de la clase correspondiente a su valor. La siguiente tabla muestra las variables definidas automáticamente.

Variable	Valor	Notación formal
VelocityAid	adalid.commons.velocity.VelocityAid	${VelocityAid}
StrUtils	adalid.commons.util.StrUtils	${StrUtils}
TimeUtils	adalid.commons.util.TimeUtils	${TimeUtils}
StringUtils	org.apache.commons.lang.StringUtils	${StringUtils}
StringEscapeUtils	org.apache.commons.lang.StringEscapeUtils	${StringEscapeUtils}
Boolean	java.lang.Boolean	${Boolean}
Byte	java.lang.Byte	${Byte}
Character	java.lang.Character	${Character}
Double	java.lang.Double	${Double}
Float	java.lang.Float	${Float}
Integer	java.lang.Integer	${Integer}
Long	java.lang.Long	${Long}
Short	java.lang.Short	${Short}
String	java.lang.String	${String}
System	java.lang.System	${System}
BigDecimal	java.math.BigDecimal	${BigDecimal}
BigInteger	java.math.BigInteger	${BigInteger}
Date	java.sql.Date	${Date}
Time	java.sql.Time	${Time}
Timestamp	java.sql.Timestamp	${Timestamp}

Propiedades .instance

Las propiedades **.instance** permiten definir variables cuyo valor es una instancia de una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.instance**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que tenga un constructor público sin parámetros.

Las variables definidas mediante propiedades **.instance** pueden ser utilizadas en las plantillas especificadas en los archivos de propiedades de plantillas que se procesan al generar el proyecto. Estas

variables se utilizan para ejecutar los métodos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Existe una variable de este tipo que se define de manera automática, sin necesidad utilizar una propiedad **.instance**. El nombre de esta variable es **project**. El valor de la variable **project** es la instancia del proyecto que se genera. Siguiendo las reglas de Velocity, la notación formal para las referencias a esta variable es **${project}** y la notación abreviada es **$project**.

Propiedades .programmer

Las propiedades **.programmer** permiten definir variables cuyo valor es una instancia de una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.programmer**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que implemente la interfaz **adalid.commons.interfaces.Programmer** y tenga un constructor público sin parámetros.

Las variables definidas mediante propiedades **.programmer** pueden ser utilizadas en las plantillas especificadas en los archivos de propiedades de plantillas que se procesan al generar el proyecto. Estas variables se utilizan para ejecutar los métodos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Propiedades .wrapper

Las propiedades **.wrapper** permiten especificar las clases envolventes (*wrapper classes*) de las distintas clases de artefactos de Adalid. Los nombres de estas propiedades tienen la forma ***wrappable*.wrapper**, en donde ***wrappable*** corresponde al nombre completo (incluyendo el paquete) de una clase de artefactos de Adalid. El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que implemente la interfaz **adalid.commons.interfaces.Wrapper** y tenga un constructor con un objeto de la clase identificada por ***wrappable*** como único parámetro. Todas las clases de artefactos de Adalid tienen una clase envolvente predeterminada. Solo para especificar una clase envolvente diferente para una clase de artefacto en particular hay necesidad de utilizar una propiedad **.wrapper**. La siguiente tabla muestra las clases envolventes predeterminadas.

Clase de artefacto	Clase envolvente predeterminada
Expresiones	adalid.core.wrappers.ExpressionWrapper
Campos de Claves de Acceso	adalid.core.wrappers.KeyFieldWrapper
Operaciones	adalid.core.wrappers.OperationWrapper
Entidades Persistentes	adalid.core.wrappers.PersistentEntityReferenceWrapper
Todas las demás clases de entidades	adalid.core.wrappers.EntityWrapper
Propiedades y Parámetros de tipo primitivo	adalid.core.wrappers.DataArtifactWrapper
Todas las demás clases de artefactos	adalid.core.wrappers.ArtifactWrapper

Propiedades do.isolated.delete

Ruta absoluta de un directorio cuyos archivos son eliminados antes de la generación del proyecto.

Un mismo archivo puede contener tantas propiedades **do.isolated.delete** como sea necesario para especificar todos los directorios cuyos archivos deben ser eliminados.

Propiedades do.cascaded.delete

Ruta absoluta de un directorio cuyos archivos son eliminados antes de la generación del proyecto. A diferencia de la propiedad **do.isolated.delete**, con esta propiedad también son eliminados los archivos de los subdirectorios del directorio especificado.

Un mismo archivo puede contener tantas propiedades **do.cascaded.delete** como sea necesario para especificar todos los directorios cuyos archivos deben ser eliminados.

Variables

El valor de las propiedades de los archivos de propiedades de plataformas puede contener referencias a las siguientes variables:

- **project**: instancia del proyecto que se genera.
- **rootFolderSlashedPath**: ruta que se especifica mediante la propiedad **root.folder** del archivo **bootstrapping.properties** del meta-proyecto.
- Variables definidas mediante propiedades **.string** en este mismo archivo

El valor de las propiedades **file.resource.loader.path**, **do.isolated.delete** y **do.cascaded.delete** puede comenzar con una referencia a la variable **rootFolderSlashedPath** y contener referencias a la variable **project**.

El valor de las propiedades **do.isolated.delete** y **do.cascaded.delete** puede contener referencias a las variables definidas mediante propiedades **.string** en este mismo archivo.

Siguiendo las reglas de Velocity, la notación formal para las referencias a una variable es **${*variable*}** y la notación abreviada es **$*variable***, donde ***variable*** corresponde al nombre de la variable.

Observaciones

El carácter separador de ruta (*path separator*) que se utiliza en los valores de las propiedades es la barra oblicua (*slash*), aun en ambiente Windows.

Archivos de propiedades de plantillas

Propiedad template

Ruta de una plantilla para generar archivos. La ruta es relativa al directorio definido mediante el valor de la propiedad **file.resource.loader.path** del archivo **velocity.properties**. Esta propiedad no tiene valor por omisión.

Propiedad template-type

Tipo de la plantilla especificada mediante la propiedad **template**. Los valores que puede tomar esta propiedad son: **document** y **velocity**. Si es **document**, los archivos se generan simplemente copiando la plantilla. Si es **velocity**, los archivos se generan utilizando el motor de plantillas (*template engine*) de Velocity para combinar la plantilla con el modelo de la aplicación. El valor por omisión de esta propiedad es **velocity**.

Propiedad template-encoding

Codificación de caracteres de la plantilla especificada mediante la propiedad **template**. La siguiente tabla muestra los valores que puede tomar esta propiedad.

Valor	Descripción
US-ASCII	ASCII de 7 bits, también conocido como ISO646-US

Valor	Descripción
ISO-8859-1	Alfabeto Latino número 1, también conocido como ISO-LATIN-1
UTF-8	Formato de transformación Unicode de 8 bits
UTF-16	Formato de transformación Unicode de 16 bits
UTF-16BE	Formato de transformación Unicode de 16 bits, orden big-endian
UTF-16LE	Formato de transformación Unicode de 16 bits, orden little-endian

El valor por omisión de esta propiedad es **ISO-8859-1**.

Propiedad for-each

Especifica una o más variables que identifican un elemento de una colección de objetos del modelo de la aplicación. Se genera un archivo para cada elemento de la colección, utilizando la plantilla especificada mediante la propiedad **template**.

En su forma simple, el valor de la propiedad **for-each** es el nombre de una variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la variable es el elemento de la colección para el que se genera el archivo. La variable puede ser utilizada en la plantilla especificada mediante la propiedad **template** para acceder a los métodos del elemento. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*variable*}** y la notación abreviada es **$*variable***, en donde ***variable*** corresponde al nombre especificado.

Cuando se utiliza la forma simple de la propiedad **for-each**, la colección se obtiene ejecutando un método del objeto **project**, es decir, de la instancia del proyecto que se genera. El nombre de este método puede ser determinado a partir del nombre de la variable o puede ser explícitamente especificado mediante una propiedad **.getter**, como se explica en la siguiente sección.

Por ejemplo:

```
for-each = entity
entity.getter = getEntitiesList
```

En el ejemplo anterior se utiliza el método **getEntitiesList** del objeto **project** para obtener una colección de entidades, y se produce un archivo para cada entidad de la colección. La plantilla puede acceder a los métodos de la entidad con referencias a la variable **entity**.

En su forma compuesta, el valor de la propiedad **for-each** es el nombre de dos o más variables, separados por un punto y sin espacios intermedios, al estilo de la notación punteada de Java. Cada uno de los nombres especificados debe cumplir con las reglas de nomenclatura de Velocity. El valor de la última variable es el elemento de la colección para el que se genera el archivo. Los valores de las primeras variables son los objetos intermedios entre el objeto **project** y el elemento. Cada una de las variables puede ser utilizada en la plantilla especificada mediante la propiedad **template** para acceder a los métodos de los objetos correspondientes.

Cuando se utiliza la forma compuesta de la propiedad **for-each**, primero se obtiene un objeto o una colección ejecutando un método del objeto **project**. Luego se ejecuta un método del objeto obtenido, o de cada elemento de la colección obtenida, para obtener otro objeto u otra colección, y esta operación se repite hasta obtener los elementos de la colección correspondiente a la última variable. El nombre de estos métodos puede ser determinado a partir del nombre de las variables o puede ser explícitamente especificado mediante una propiedad **.getter**, como se explica en la siguiente sección.

Por ejemplo:

```
for-each = entity.operation
entity.getter = getEntitiesList
```

```
operation.getter = getBusinessOperationsList
```

En el ejemplo anterior se ejecuta el método **getEntitiesList** del objeto **project** para obtener una colección de entidades; a continuación, para cada entidad de la colección se ejecuta su método **getBusinessOperationsList** para obtener una colección de operaciones, y se produce un archivo para cada operación de la colección. La plantilla puede acceder a los métodos de la entidad con referencias a la variable **entity** y a los métodos de la operación con referencias a la variable **operation**.

Propiedades .getter

Las propiedades **.getter** permiten definir los métodos para obtener los objetos o colecciones correspondientes a variables especificadas mediante la propiedad **for-each**. Los nombres de estas propiedades tienen la forma ***variable*.getter**, en donde ***variable*** corresponde al nombre de alguna de las variables especificadas mediante la propiedad **for-each**. Los valores de estas propiedades son nombres de métodos sin parámetros de los objetos correspondientes.

Se debe especificar una sola propiedad **.getter** para cada variable de la propiedad **for-each**. Si no se especifica la propiedad **.getter** de una variable, entonces el nombre del método que se utiliza para esa variable será el primero de los siguientes nombres que coincida con el de un método sin parámetros del objeto correspondiente:

- **get*Plural*List**, donde ***Plural*** es el sustantivo plural del nombre de la variable. Para determinar el sustantivo plural se asume que el nombre de la variable es un sustantivo simple del idioma inglés.
- **get*Plural***, donde ***Plural*** es el sustantivo plural del nombre de la variable. Para determinar el sustantivo plural se asume que el nombre de la variable es un sustantivo simple del idioma inglés.
- **get*Singular***, donde ***Singular*** es el sustantivo singular del nombre de la variable. Para determinar el sustantivo singular se asume que el nombre de la variable es un sustantivo simple del idioma inglés.
- **get*Variable***, donde ***Variable*** es el nombre de la variable.

Por ejemplo, para la siguiente propiedad **for-each**:

```
for-each = entity
```

Si no se especifica la propiedad **entity.getter**, entonces el nombre del método para la variable **entity** será el primero de los siguientes nombres que coincida con el de un método sin parámetros de la entidad: **getEntitiesList**, **getEntities**, **getEntity**. Note que son solo tres posibilidades porque tanto **get*Singular*** como **get*Variable*** dan como resultado **getEntity**.

Propiedades .predicate

Las propiedades **.predicate** permiten definir predicados para filtrar las colecciones correspondientes a variables especificadas mediante la propiedad **for-each**. Los nombres de estas propiedades tienen la forma ***variable*.predicate**, en donde ***variable*** corresponde al nombre de alguna de las variables especificadas mediante la propiedad **for-each**. Los valores de estas propiedades son nombres completos (incluyendo el paquete) de clases Java que implementen la interfaz **org.apache.commons.collections.Predicate** y tengan un constructor público sin parámetros.

Se pueden especificar tantas propiedades **.predicate** como sea necesario para una misma colección. Los predicados se combinan con el método establecido por la correspondiente propiedad **.predicate.join**.

Propiedades .not.predicate

Las propiedades **.not.predicate** permiten definir predicados negados para filtrar las colecciones correspondientes a variables especificadas mediante la propiedad **for-each**. Los nombres de estas propiedades tienen la forma ***variable*.predicate**, en donde ***variable*** corresponde al nombre de alguna de las variables especificadas mediante la propiedad **for-each**. Los valores de estas propiedades son nombres completos (incluyendo el paquete) de clases Java que implementen la interfaz **org.apache.commons.collections.Predicate** y tengan un constructor público sin parámetros.

Se pueden especificar tantas propiedades **.not.predicate** como sea necesario para una misma colección. Los predicados se combinan con el método establecido por la correspondiente propiedad **.predicate.join**.

Propiedades .predicate.join

Método que se utiliza para combinar los predicados definidos para filtrar una colección. Solo es relevante cuando se especifica más de una propiedad **.predicate** y/o **.not.predicate** para una misma colección. Los valores que puede tomar esta propiedad son: **all, any, none** y **one**. Si es **all**, se generan archivos para los elementos de la colección que cumplan con todos los predicados. Si es **any**, se generan archivos para los elementos de la colección que cumplan al menos con uno de los predicados. Si es **none**, se generan archivos para los elementos de la colección que no cumplan ninguno de los predicados. Si es **one**, se generan archivos para los elementos de la colección que no cumplan con uno y solo uno de los predicados. El valor por omisión de esta propiedad es **all**.

Propiedades .comparator

Las propiedades **.comparator** permiten definir comparadores para ordenar las colecciones correspondientes a variables especificadas mediante la propiedad **for-each**. Los nombres de estas propiedades tienen la forma ***variable*.comparator**, en donde ***variable*** corresponde al nombre de alguna de las variables especificadas mediante la propiedad **for-each**. Los valores de estas propiedades son nombres completos (incluyendo el paquete) de clases Java que implementen la interfaz **java.util.Comparator** y tengan un constructor público sin parámetros.

Se pueden especificar tantas propiedades **.comparator** como sea necesario para una misma colección. Los comparadores se combinan para ordenar los elementos de la colección usando todos los comparadores especificados y en el orden en el que son especificados.

Propiedad path

Ruta absoluta del directorio donde se almacenan los archivos generados. El valor por omisión de esta propiedad es el valor de la propiedad **root.folder** del archivo **bootstrapping.properties**.

Propiedad package

Paquete Java donde se almacenan los archivos generados. Si el archivo generado no es un archivo Java, entonces define el subdirectorio (del directorio definido mediante la propiedad **path**) donde se almacenan los archivos generados. Para determinar la tuta del subdirectorio, se cambian los puntos en el nombre del paquete por el carácter separador de ruta (*path separator*). Esta propiedad no tiene valor por omisión pero es opcional.

Propiedad file

Nombre y, opcionalmente, extensión de los archivos generados. Esta propiedad no tiene valor por omisión.

Propiedad file-encoding

Codificación de caracteres de los archivos generados. La siguiente tabla muestra los valores que puede tomar esta propiedad.

Valor	Descripción
US-ASCII	ASCII de 7 bits, también conocido como ISO646-US
ISO-8859-1	Alfabeto Latino número 1, también conocido como ISO-LATIN-1
UTF-8	Formato de transformación Unicode de 8 bits
UTF-16	Formato de transformación Unicode de 16 bits
UTF-16BE	Formato de transformación Unicode de 16 bits, orden big-endian
UTF-16LE	Formato de transformación Unicode de 16 bits, orden little-endian

El valor por omisión de esta propiedad es **ISO-8859-1**.

Propiedad disabled

Indica si se generan, o no, archivos. Los valores que puede tomar esta propiedad son: **true** y **false**. Si es **true**, no se generan archivos. El valor por omisión de esta propiedad es **false**.

Propiedad preserve

Indica si se generan, o no, archivos que ya existen. Los valores que puede tomar esta propiedad son: **true** y **false**. Si es **true**, no se generan los archivos que ya existen. El valor por omisión de esta propiedad es **false**.

Propiedad execute-command

Comando del Sistema Operativo (Windows) que se ejecuta al finalizar la generación de cada archivo. Esta propiedad no tiene valor por omisión pero es opcional.

Propiedad execute-directory

Ruta del directorio de trabajo (*working directory*) del proceso que se inicia para ejecutar el comando especificado mediante la propiedad **execute-command**. La ruta puede ser absoluta o relativa al directorio de trabajo del usuario (*user working directory*), cuya ruta se obtiene de la propiedad **user.dir** de la plataforma Java. El valor por omisión de esta propiedad es el valor de la propiedad **user.dir** de la plataforma Java.

Propiedades .string

Las propiedades **.string** permiten definir variables cuyo valor es una cadena de caracteres. Los nombres de estas propiedades tienen la forma ***identificador*.string**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el valor que se da a la variable.

Se pueden incluir referencias a variables definidas mediante propiedades **.string** en los valores de las propiedades **template**, **path**, **package**, **file**, **execute-command** y **execute-directory** del mismo archivo y en la plantilla especificada mediante la propiedad **template**. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Propiedades .class

Las propiedades **.class** permiten definir variables cuyo valor es una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.class**, en donde ***identificador*** corresponde al nombre que se

da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java.

Las variables definidas mediante propiedades **.class** pueden ser utilizadas en la plantilla especificada mediante la propiedad **template**. Estas variables se utilizan para ejecutar los métodos estáticos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Existe un conjunto de clases para las cuales se definen variables de manera automática, sin necesidad de utilizar propiedades **.class**. El nombre de estas variables es el nombre simple (sin el paquete) de la clase correspondiente a su valor. La siguiente tabla muestra las variables definidas automáticamente.

Variable	Valor	Notación formal
VelocityAid	adalid.commons.velocity.VelocityAid	${VelocityAid}
StrUtils	adalid.commons.util.StrUtils	${StrUtils}
TimeUtils	adalid.commons.util.TimeUtils	${TimeUtils}
StringUtils	org.apache.commons.lang.StringUtils	${StringUtils}
StringEscapeUtils	org.apache.commons.lang.StringEscapeUtils	${StringEscapeUtils}
Boolean	java.lang.Boolean	${Boolean}
Byte	java.lang.Byte	${Byte}
Character	java.lang.Character	${Character}
Double	java.lang.Double	${Double}
Float	java.lang.Float	${Float}
Integer	java.lang.Integer	${Integer}
Long	java.lang.Long	${Long}
Short	java.lang.Short	${Short}
String	java.lang.String	${String}
System	java.lang.System	${System}
BigDecimal	java.math.BigDecimal	${BigDecimal}
BigInteger	java.math.BigInteger	${BigInteger}
Date	java.sql.Date	${Date}
Time	java.sql.Time	${Time}
Timestamp	java.sql.Timestamp	${Timestamp}

Propiedades .instance

Las propiedades **.instance** permiten definir variables cuyo valor es una instancia de una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.instance**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que tenga un constructor público sin parámetros.

Las variables definidas mediante propiedades **.instance** pueden ser utilizadas en la plantilla especificada mediante la propiedad **template**. Estas variables se utilizan para ejecutar los métodos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Existe una variable de este tipo que se define de manera automática, sin necesidad utilizar una propiedad **.instance**. El nombre de esta variable es **project**. El valor de la variable **project** es la instancia del proyecto que se genera. Siguiendo las reglas de Velocity, la notación formal para las referencias a esta variable es **${project}** y la notación abreviada es **$project**.

Propiedades .programmer

Las propiedades **.programmer** permiten definir variables cuyo valor es una instancia de una clase Java. Los nombres de estas propiedades tienen la forma ***identificador*.programmer**, en donde ***identificador*** corresponde al nombre que se da a la variable y debe cumplir con las reglas de nomenclatura de Velocity; es decir, debe comenzar por una letra y solo puede contener letras, números, guiones y guiones bajos (*underscores*). El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que implemente la interfaz **adalid.commons.interfaces.Programmer** y tenga un constructor público sin parámetros.

Las variables definidas mediante propiedades **.programmer** pueden ser utilizadas en la plantilla especificada mediante la propiedad **template**. Estas variables se utilizan para ejecutar los métodos de la clase especificada. Siguiendo las reglas de Velocity, la notación formal para las referencias a estas variables es **${*identificador*}** y la notación abreviada es **$*identificador***.

Propiedades .wrapper

Las propiedades **.wrapper** permiten especificar las clases envolventes (*wrapper classes*) de las distintas clases de artefactos de Adalid. Los nombres de estas propiedades tienen la forma ***wrappable*.wrapper**, en donde ***wrappable*** corresponde al nombre completo (incluyendo el paquete) de una clase de artefactos de Adalid. El valor de la propiedad es el nombre completo (incluyendo el paquete) de una clase Java que implemente la interfaz **adalid.commons.interfaces.Wrapper** y tenga un constructor con un objeto de la clase identificada por ***wrappable*** como único parámetro.

Todas las clases de artefactos de Adalid tienen una clase envolvente predeterminada. Solo para especificar una clase envolvente diferente para una clase de artefacto en particular hay necesidad de utilizar una propiedad **.wrapper**. La siguiente tabla muestra las clases envolventes predeterminadas.

Clase de artefacto	Clase envolvente predeterminada
Expresiones	adalid.core.wrappers.ExpressionWrapper
Campos de Claves de Acceso	adalid.core.wrappers.KeyFieldWrapper
Entidades Persistentes	adalid.core.wrappers.PersistentEntityReferenceWrapper
Propiedades y Parámetros de tipo primitivo	adalid.core.wrappers.DataArtifactWrapper
Todas las demás clases de artefactos	adalid.core.wrappers.ArtifactWrapper

Variables

El valor de las propiedades de los archivos de propiedades de plantillas puede contener referencias a las siguientes variables:

- **project**: instancia del proyecto que se genera.
- **rootFolderSlashedPath**: ruta que se especifica mediante la propiedad **root.folder** del archivo **bootstrapping.properties** del meta-proyecto.
- Variables definidas mediante la propiedad **for-each**.
- Variables definidas mediante propiedades **.string**, en este mismo archivo y en el archivo de propiedades de la plataforma.

El valor de las propiedades **template**, **path**, **execute-command** y **execute-directory** puede comenzar con una referencia a la variable **rootFolderSlashedPath**.

El valor de las propiedades **template**, **path**, **package**, **file**, **execute-command** y **execute-directory** puede puede contener referencias a la variable **project**, a las variables definidas mediante la propiedad **for-each** y a las variables definidas mediante propiedades **.string**, en este mismo archivo y en el archivo de propiedades de la plataforma.

Siguiendo las reglas de Velocity, la notación formal para las referencias a una variable es **${*variable*}** y la notación abreviada es **$*variable***, donde ***variable*** corresponde al nombre de la variable.

Observaciones

El carácter separador de ruta (*path separator*) que se utiliza en los valores de las propiedades es la barra oblicua (*slash*), aun en ambiente Windows.