Com realitzar les tasques de construcció amb Maven

A qui va dirigit

Aquest how-to va dirigit a tots els desenvolupadors d'aplicacions basades en Canigó.

Versió de Canigó

Els passos descrits en aquest document són vàlids per a la versió 2.3.4 de Canigó i podran ser d'aplicació per a versions posteriors.

Context

Davant la necessitat d'independitzar al màxim les fases del cicle de vida d'un projecte Canigó amb la plataforma, i degut a les mancances que s'han anat trobant amb l'eina ANT, el plugin Maven 2 de Canigó ha quedat obsolet.

En substitució d'aquest s'utilitzaran plugins propis de Maven que realitzaran les mateixes funcions que l'antic plugin.

Tot seguit, mostrem una taula amb la correspondència dels goals de:

Goals Plugin Maven de Canigó: ctti (deprecat)	Goals Maven (actualment)
`refresh-libs`	**`dependecy:copy-dependencies`**
Copia les dependències (jar) del projecte a *.deployables/<contexte>/WEB-INF/lib*, útil per tal de desplegar el projecte amb les Web Tools Platform de Eclipse.	Es pot afegir l'opció –DoutputDirectory per indicar el directori de sortida on es volen deixar les llibreries.
`refresh-webapp`	**`package`**
Aplica l'AOP AspectWerkz a la carpeta *.deployables*, que és on estan situats els binaris compilats per l'Eclipse i addicionalment executa el goal `ctti:refresh-libs`.	Compila el projecte si és necessari, a més d'aplicar l'AOP, i empaqueta l'aplicació en format war.
`deploy.j2ee.ctti.weblogic.static` **`deploy.j2ee.ctti.weblogic.dynamic`**	**`assembly:assembly`**
El primer construeix un zip amb el contingut estàtic de l'aplicació a la carpeta *'target/ear/weblogic'*. El segon goal construeix el war amb el contingut dinàmic; aquest goal assumeix que el projecte esta construït a *.deployables/<nom_del_projecte>.*	Genera els dos artefactes en una mateixa tasca. Obtenim la part dinàmica (war) i estàtica (zip) amb un sol goal.

Com realitzar les tasques de construcció amb Maven

Objectiu

L'objectiu que es pretén assolir amb aquest document és proporcionar una guia amb els passos que s'han de seguir a l'hora de construir aplicacions Canigó amb Maven: compilar i empaquetar.

Passos a seguir

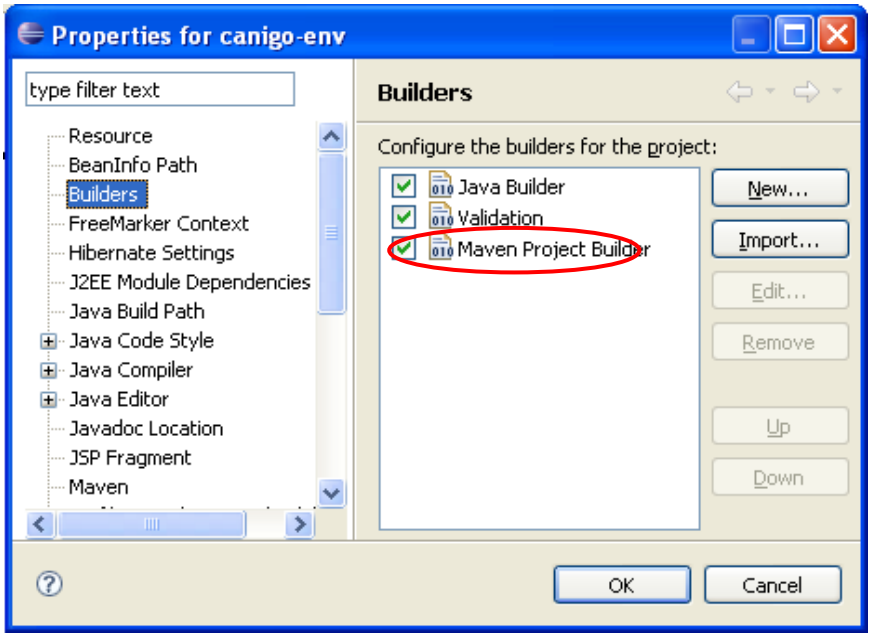
1. **Afegir Maven en el procés de construcció del projecte**

   Per tal de què Maven interfereixi en el procés de build d'Eclipse (de manera transparent al desenvolupador i sense necessitat de fer-ho manualment posteriorment), és necessari afegir el builder de Maven en aquest procés.

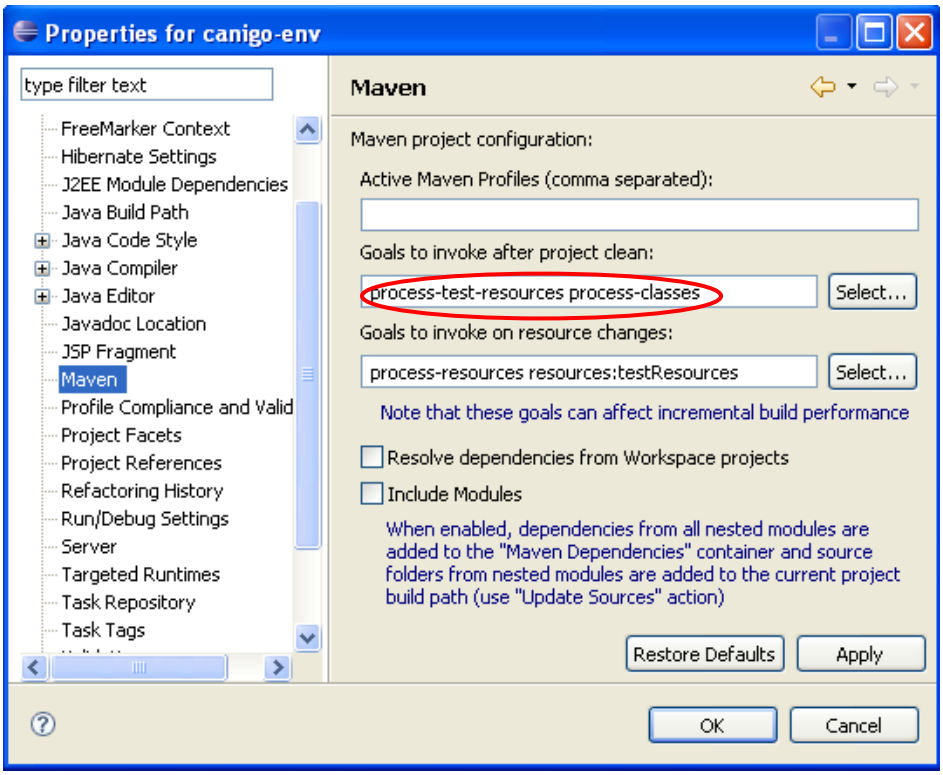
   ***NOTA**: Primer de tot assegurar-nos que el projecte té natura Maven. Per comprovar-ho, fer clic amb el botó dret sobre el projecte, i en l'opció Maven si només apareix "Enable dependency management" vol dir que no està activat. Si és així, seleccionar aquesta opció i tindrem el projecte amb natura Maven.*

Per afegir Maven al procés de construcció, s'ha d'anar a les propietats del projecte i:

a) seleccionar l'opció *Builders* i marcar Maven Project Builder (si aquesta no està seleccionada).



b) seleccionar l'opció *Maven* i en el camp "Goals to invoke after project clean", a més del que hi ha, afegir la fase "process-classes" de la manera com es mostra a continuació:



Així s'executarà (en la fase process-classes) la tasca que hi hagi definida en el pom de l'aplicació per aquesta fase del cicle de vida, que en el nostre cas es tracta aplicar l'AOP a les classes compilades.

2. Construcció del projecte

Per construir el projecte, el seleccionem i fem un *Project – Clean...* Si no tenim seleccionada l'opció *Build automatically*, després del clean, seleccionar *Project – Build Project*.

Serà en aquest moment que el builder de Maven intervindrà en el procés de construcció, configurat en el pas previ, per tal d'aplicar, tal com hem esmentat, l'AOP a les classes compilades.

ATENCIÓ: El fet de refrescar (F5) el projecte un cop l'Eclipse hagi acabat de compilar i construir el projecte, fa que es perdi l'aspectització de les classes.

Com realitzar les tasques de construcció amb Maven

3. Desplegament de l'aplicació en local

Un cop haguem configurat el servidor, i la connexió a la base de dades, publiquem el projecte al servidor (Publish) i ja estem en disposició d'arrancar l'aplicació.

Amb la publicació es desplega l'aplicació en el directori de deploy del servidor local.

4. Empaquetat de l'aplicació

- Si es vol desplegar l'aplicació en entorns (desenvolupament, integració) en que sigui necessari disposar de l'aplicació empaquetada, en aquest cas, executant els goals de maven **`package`** o **`install`** a l'arrel del projecte, ens quedarà el war generat a la carpeta *<project-root>/target* (i al nostre repositori local, si executem `install`).

- Si s'ha de desplegar l'aplicació en els entorns corporatius de la Generalitat (que per norma general consten d'Apache + BEA Weblogic), serà necessari separar la part dinàmica de l'estàtica. Amb el goal **`assembly:assembly`** obtindrem aquesta separació en dos artefactes diferents: pel que fa a la part dinàmica es generarà un war amb les classes, JSPs, llibreries, etc.; en canvi, per la part estàtica obtindrem un zip amb les imatges, css, html, etc.

- Hi pot haver casos en què sigui necessari realitzar un tercer tipus d'empaquetat: EAR. En aquest cas, és necessari descarregar-se l'aplicació plantilla de Canigó, versió 2.3.4.

  Dins de la carpeta arrel canigo-env, hi ha una nova jerarquia de carpetes, canigo-ear. Per a generar l'EAR cal disposar prèviament del war de l'aplicació; tot seguit, executant els goals de maven **`package`** o **`install`** ubicant-nos a canigo-ear, ens quedarà el fitxer ear generat a la carpeta *canigo-ear/target* (i al nostre repositori local, si executem `install`).