3

Zaletą Javy jest elastyczność, czyli możemy każdy projekt dowolnie skonfigurować, zaleta ta zamienia się w wadę, kiedy przy jednym projekcie pracuje wiele osób, który strukturyzuje projekt i dla każdego kod staje się przejrzysty (struktura).

Maven zrodził się z potrzeby zdefiniowania standardowej struktury projektu, standardowego sposobu budowania aplikacji oraz sposobu definiowania zależności pomiędzy modułami projektu.

Apache Maven jest narzędziem automatyzującym budowę oprogramowania na platformę Java

5.

validate – walidacja modelu projektu

compile – kompilacja źródeł

test – uruchamia testy zdefiniowane dla kodu

package – skompilowany kod jest pakowany do wynikowego archiwum np. JARa

integration-test – przygotowuje i wdraża gotowy plik projektu w środowisku w którym będzie on przedmiotem testów integracyjnych.

verify – faza w której dokonywane są wszystkie możliwe sprawdzenia dotyczące jakości stworzonego archiwum

install – instaluje stworzoną postać dystrybucyjną w lokalnym repozytorium zależności, tak aby mogły z niego korzystać inne projekty.

deploy – kopiuje postać dystrybucyjną do zewnętrznego środowiska w którym może być wykorzystywane przez innych programistów.

Żaden następny krok nie może zostać wykonany bez wykonania kroków poprzednich

Wszystkie cykle życia to tak naprawdę pluginy wbudowane dostępne domyślnie (mvn plugin:goal)

http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

7. **compile** - Kompilacja, testowanie i uruchomienie. Biblioteka dostępna zawsze i dołączana do wynikowego artefaktu. Jest to domyślna wartość elementu scope ustawiana, jeśli zostanie on pominięty przy definicji zależności.

provided - Kompilacja i testowanie. Zakładamy, że biblioteka będzie dostarczona przez środowisko uruchomieniowe (kontener), nie jest więc dołączana (np. ServletAPI)

test - Kompilacja testów i testowanie. Dostępne tylko podczas kompilacji i uruchomienia testów.

runtime - Testowanie i uruchomienie. Biblioteka dostępna podczas uruchamiania testów oraz wymagana do działania aplikacji, jest więc dołączana do artefaktu wynikowego.

10. - *groupId* – informuje o grupie produktów, w której znajduje się projekt, zwykle (choć nie zawsze) składa się - na podobieństwo javowych pakietów - z domeny, dostawcy oraz nazwy całej grupy produktów,

- *artifactId* – to bezpośrednia nazwa projektu w obrębie grupy, np. crm-api, dao-jdbc itp. (patrz uwaga wyżej),

- *version* – określenie numeru wersji, za pomocą liczb lub liczb mieszanych z opisem, np. 1.0, 2.1, 2.2-BETA, 2.2-RC1, itp. (wersjom przyjrzymy się dokładnie w kolejnym numerze),

- *name* – nazwa dla projektu, nieco bardziej opisowa niż zbitek składający się z grupId orazartifactId, nazwa ta będzie używana w kilku miejscach, warto wybrać taką, która będzie kojarzyć się z projektem, ale nie będzie długa, np. SuperCRM.API – dla klasa i interfejsów stanowiących API projektu,

- *description* – opis projektu, może być dowolnie długi, ale warto zachować umiar, gdyż jeśli do wyszukiwania artefaktów będących wynikiem naszej pracy będziemy używać wyszukiwarki podobnej do opisanej na początku artykułu (oczywiście przeszukującej nasze firmowe lub prywatne repozytorium) to zawartość sekcji *description* pojawi się w wynikach wyszukiwania,

- *url* – tutaj można umieścić adres witryny przechowującej informacje o projekcie, (adres w wersji pełnej np. http://dev.supercrm.com/ lub http://supercrm.com,

- *organization* – element zbiorczy zawierający nazwę i adres strony domowej (zwyczajowo) organizacji zajmującej się rozwijaniem projektu, czyli – w przypadku klasycznej działalności komercyjnej - po prostu firmy, która zajmuje się jego tworzeniem; składa się z dwóch zagnieżdżonych elementów:

- *name* – nazwa organizacji,

- *url* – adres strony domowej organizacji,