Notatki - Maven - Intro

Spis treści

Czym jest Maven?

Maven jest narzędziem, które służy do m.in. budowania. Nie jest to oczywiście jedyne narzędzie, które do tego służy. Czym jest samo budowanie projektu, wyjaśni się w trakcie.

W trakcie trwania warsztatów poruszymy podstawowe koncepcje działania tego narzędzia, nie wspomnimy natomiast o wszystkich możliwych jego mechanizmach - o tym często przekonasz się już w pracy w praktyce. Skupimy się na zrozumieniu najważniejszych koncepcji.

Obraz 1. Maven logo. Źródło: https://pl.wikipedia.org/

Czym jest budowanie?

Wspomnieliśmy, że **Maven** jest narzędziem do budowania. Na stwierdzenie `budowanie` mogą składać się poniższe czynności:

- **zarządzanie zależnościami** - dotychczas dociągaliśmy do naszego projektu bardzo mało zależności zewnętrznych. Czyli gotowych fragmentów kodu, pozwalających osiągnąć nam jakiś cel szybciej. Nie musimy wtedy takiego kodu pisać na własną rękę. Narzędzia do budowania pozwalają nam w prosty sposób zarządzać takimi zależnościami zewnętrznymi w naszym projekcie.
- **kompilacja kodu źródłowego** - poznaliśmy już tę czynność wcześniej. Można było uruchomić kompilację programu albo za pomocą komendy `javac`, albo przy wykorzystaniu Intellij i przycisku `run` (tutaj kod się kompilował i od razu uruchamiał).
- **generowanie dokumentacji** - w kodzie źródłowym możemy opisywać nasze klasy i metody przy wykorzystaniu `javadoc`, a następnie przygotować w ten sposób dokumentację projektu, która będzie podobna do np. Dokumentacji klasy String.

- **generowanie kodu źródłowego** - w praktyce kod źródłowy może być oczywiście pisany (i to jest to czego się uczymy cały czas). Może natomiast też wystąpić sytuacja, w której kod źródłowy będziemy generować automatycznie. Wtedy z pomocą przyjdą nam narzędzia do budowania i pluginy (rozszerzenia) pozwalające na generację kodu.
- **przygotowywanie paczek** - jeżeli chcielibyśmy nasz program Java wysłać komuś w postaci jednego pliku, który może zostać uruchomiony, to możemy w tym celu przygotować archiwum `.jar` lub `.zip`. Plik `.jar` (Java archive) został już pokazany w przypadku korzystania z JDBC, gdzie potrzebowaliśmy pobrać sterownik do bazy danych. Sterownik taki był dostarczony w pliku `.jar`. Pokażemy również, jak przygotować program w pliku `.jar`, który można uruchomić mając plik `.jar`.
- **instalacja programu na serwerze** - po ukończeniu programu możemy zainstalować go na serwerze (na innym komputerze dostępnym w sieci, z którego nasz program będzie mógł być uruchamiany).

Oprócz czynności wymienionych powyżej, narzędzia służące do budowania projektów, mogą wykonywać wiele więcej. Wszystkie z tych czynności mogą być oczywiście wykonywane ręcznie. Narzędzia tego typu powstają natomiast aby automatyzować procesy tego typu i eliminować możliwość popełnienia błędu przez ludzi. Z reguły najczęstszym powodem powstania błędu w projektach informatycznych jest czynnik ludzki ☺. Do tego jest szybciej, komputer klika szybciej niż człowiek.

Słowa kluczowe

Jak z każdym narzędziem, dochodzi nam kwestia terminologii, która jest nowa i którą trzeba wyjaśnić. Poniżej wyjaśniam tylko skrótowo pojęcia, w dalszych częściach materiału będziemy rozmawiać o nich bardziej szczegółowo.

POM

`POM - Project Object Model`. **POM** jest plikiem w formacie `.xml`, który zawiera opis tego jakie zasoby w naszym projekcie powinny zostać zbudowane. Wspomniałem również o zarządzaniu zależnościami. W **POM**ie dodajemy informacje o zależnościach, które są nam potrzebne w naszym projekcie. Plik `pom.xml` powinien zostać umiejscowiony "na górze" struktury katalogów naszego projektu (zobaczysz w praktyce co mam na myśli). Jednocześnie też, jeżeli wywołujemy proces budowania, musimy wskazać lokalizację pliku `pom.xml` na dysku (po to żeby narzędzie wiedziało, gdzie jest opis tego co i jak ma zbudować).

Build lifecycle

W procesie budowania projektu możemy wyróżnić pojęcia takie jak: `lifecycle`, `phase` oraz `goal`. Lifecycle składa się z `phases`, a każda `phase` składa się z `goals`. Domyślnie wbudowane mamy 3 rodzaje `lifecycle`: `default`, `clean` i `site`. (Cały czas będę starał się używać angielskich nazw, wybacz mi łamańce językowe)

- **default** - główny `lifecycle`, odpowiedzialny za utworzenie zbudowanej wersji projektu i jej ewentualne wdrożenie.
- **clean** - odpowiedzialny za wyczyszczenie plików powstałych w wyniku uruchomienia poprzedniego builda.
- **site** - pozwala na utworzenie strony internetowej z dokumentacją naszego projektu.

Głównym `lifecycle` jest `default`.

Dalej nie rozpisuje się o konkretnych `phase` oraz `goals` bo zobaczysz, że w praktyce będziesz pamiętać tylko parę z nich, a resztę trzeba będzie Googlować w indywidualnych przypadkach. Dla zainteresowanych umieszczam link Life cycles, Phases, Goals

Wdrożenie jest stwierdzeniem, pod którym kryje się fizyczne umieszczenie kodu naszej aplikacji na serwerze, na którym ma być ona uruchamiana. W pracy często spotkasz sie ze stwierdzeniem "wdrożenie na PROD" lub "wdrożenie na TEST".

Najpierw wyjaśnijmy czym są tak zwane "środowiska". Jeżeli jesteś klientem jakiegoś banku prawdopodobnie możesz dostać się do aplikacji webowej swojego banku poprzez stronę `https://www.mojwymyslonybank.pl`. Strona ta jest dostępna dla klienta, czyli Ciebie jako klienta banku. Przed wdrożeniem aplikacji (czyli faktycznym umieszczeniem napisanego kodu na serwerze, na którym taka aplikacja będzie działała) należy jeszcze taką aplikację przetestować (czyli pochodzić po stronie internetowej i poklikać czy nic się nie popsuło w wyniku dodawanych zmian w kodzie). Jeżeli aplikacja już działa i dodajemy do niej dodatkowe funkcjonalności, również należy taką aplikację przetestować. W idealnym świecie mamy do tego tak zwane środowiska DEV (deweloperskie) oraz TEST (testowe). Możemy również mieć takich środowisk więcej, jest to kwestia polityki firmy. Środowiska DEV i TEST nie są dostępne na zewnątrz firmy. Oznacza to, że istnieją takie strony jak np. `https://www.mojwymyslonybank-d.pl` lub `https://www.mojwymyslonybank-t.pl`, pod którymi znajdziemy wersję aplikacji, która nie jest jeszcze wdrożona na środowisko produkcyjne, a może zawierać nowe funkcjonalności. Strony takie powinny być dostępne tylko z sieci firmowej, czyli dopiero jak pójdziesz do biura i połączysz się tam z Internetem (albo połączysz się do sieci firmowej w inny sposób) to będziesz w stanie zobaczyć te strony u siebie w przeglądarce.

Stwierdzenia "wdrożenie na PROD" lub "wdrożenie na TEST" oznaczają umieszczenie napisanego przez Ciebie kodu na każdym z serwerów, który dostarcza nam aplikację dostępną pod adresami `mojwymyslonybank-d` lub `mojwymyslonybank-t`.

Dependencies

Każdy większy projekt posiada pewne zależności. Mówiąc zależności mam na myśli zewnętrzne biblioteki (libraries), które mogą zostać przez nas wykorzystane do wytworzenia naszego oprogramowania.

`Biblioteka` (library) jest gotowym napisanym przez kogoś innego kodem, z którego możemy korzystać. Pisząc aplikacje w pracy, nie piszemy wszystkiego sami. Korzysta się z bardzo dużej ilości gotowych rozwiązań, które pozwalają nam reużyć kod napisany przez kogoś innego. Często ktoś inny i tak zrobi to za nas lepiej (szybciej, krócej), głównie dlatego, że często jest specjalistą w temacie. My dzięki temu możemy skupić się na dostarczeniu oprogramowania, dzięki któremu firma, dla której pracujemy faktycznie zarobi pieniądze.

Zależności takie są dostarczane w postaci plików `.jar`.

Repositories

Jeżeli już wspominamy o terminie `dependencies` należy również wspomnieć o terminie `repositories`. Zależności takie, po pobraniu do nas na komputer, są przechowywane w tak zwanym `lokalnym repozytorium` (local repository) - wspomnimy o tym później. Zanim natomiast pobierzemy takie zależności do nas, musimy ich wyszukać w jakimś repozytorium, np. Maven Central.

Żebyśmy też wiedzieli o czym mówię, umieszczam link do biblioteki Lombok, z której będziemy korzystać później. Widać tutaj wersje biblioteki, z których możemy korzystać.

Oprócz `Maven Central` możemy również zdefiniować inne dostępne w internecie repozytoria.

Build Plugin

Do konfiguracji budowania projektu możemy też dołożyć pluginy. Plugin jest oprogramowaniem dokładającym do naszego cyklu budowania specyficzne akcje, które są nam potrzebne na etapie budowania naszego projektu. Akcje takie np. nie są dostępne standardowo i musimy dołożyć je na własną rękę. Przykładowo, jeżeli na etapie budowania projektu będziemy chcieli wygenerować gotowy kod, którego nie chcemy pisać sami (po co, jeżeli można go wygenerować), to do takiej akcji możemy potrzebować pluginu do generowania kodu.

Build Profile

Profile budowania są wykorzystywane jeżeli chcemy wskazać różnice między sposobami budowania naszej aplikacji w zależności od tego, na które środowisko ma zostać ona wdrożona (`DEV`, `TEST`, `PROD`). Możemy wtedy włączać lub wyłączać konkretne operacje lub ustawienia, w zależności od środowiska, które nas interesuje. Mielibyśmy wtedy np. profil `DEV`, `TEST` oraz `PROD` i każdy profil przygotowywałby zbudowany projekt na dedykowane środowisko.

W praktyce spotkasz się ze stwierdzeniem **build**. Stwierdzenie "Przygotuj builda" oznacza zbudowanie projektu. Przykładowo, plik końcowy, który może zostać umieszczony na serwerze nazywany jest `buildem`. Buildy mogą też mieć wersje, oznacza to kolejne wersje aplikacji, zbudowane i wdrożone na serwer. Czasem można też spotkać się ze stwierdzeniem `solucja`, ale w praktyce słyszałem to bardzo, bardzo rzadko.