Wrocław 2018
Cezary Hołub

Wrocławska Wyższa Szkoła Informatyki Stosowanej

Przedmiot	Zaawansowane praktyki programistyczne Wprowadzenie do Apache Maven
Semestr	Lato 2018

Lab 4
05.05.2018

Materiały do ćwiczeń

Lista nie jest na ocenę!

ZADANIE 1	Instalacja Mavena

Cel ćwiczenia:

- Przeprowadzenie procesu instalacji Mavena w systemie Microsoft Windows

Wymagane wiadomości wstępne:

- Obsługa systemu Windows na poziomie użytkownika
- Obsługa przeglądarki WWW

Przebieg ćwiczenia:

1. Przy pomocy przeglądarki WWW przejdź do strony http://maven.apache.org/
2. Przejdź do sekcji pobierania (Download)
3. Pobierz najnowszą wersję Mavena (3.5.0) wybierając format "Binary zip"
4. Rozpakuj zawartość pobranego pliku do katalogu **C:\java_cwiczenia\apache**. Jeśli katalog nie istnieje, stwórz go. W tym momencie na dysku powinien się znajdować katalog **C:\java_cwiczenia\apache\apache-maven-3.5.0** wraz z podkatalogami.
5. Dodaj do zmiennej środowiskowej **PATH** ścieżkę bezwzględną do katalogu gdzie jest Maven przypisując jej wartość **C:\java_cwiczenia\apache\apache-maven-3.5.0\bin** (katalog w którym znajduje się Maven)
   a. Wciśnij klawisze WinKey+Pause
   b. Wybierz Zakładkę "Zaawansowane" i dalej "Zmienne środowiskowe"
   c. W tym samym oknie zmień wartość zmiennej **PATH** dodając na jej końcu **;C:\java_cwiczenia\apache\apache-maven-3.5.0\bin**
   d. W tym samym oknie sprawdź czy w zmiennych użytkownika istnieje zmienna **JAVA_HOME** i czy jej wartość wskazuje na lokalizację zainstalowanego środowiska JDK, np. C:\Program Files\Java\jdk1.8.0_121. Jeśli nie, utwórz ją.
   e. Upewnij się, że wartość **%JAVA_HOME%\bin** znajduje się w zmiennej **PATH**
6. Otwórz okno linii komend (WinKey + R i wprowadź *cmd*) i wprowadź polecenie mvn --version. Jeśli instalacja mavena powiodła się, na ekranie powinna pojawić się informacja o wersji Mavena. Jej przykład znajduje się poniżej:
   C:\>mvn --version
   Apache Maven 3.0.4 (r1232337; 2012-01-17 09:44:56+0100)
   Maven home: C:\apache-maven-3.0.4
   Java version: 1.6.0_37, vendor: Sun Microsystems Inc.
   Java home: C:\Program Files (x86)\Java\jdk1.6.0_37\jre
   Default locale: pl_PL, platform encoding: Cp1250
   OS name: "windows 7", version: "6.1", arch: "x86", family: "windows"

ZADANIE 2	Archetypy - tworzenie projektu z archetypu

Cel ćwiczenia:

- Praktyczne zapoznanie się z tworzeniem projektu z archetypu

Wymagane wiadomości wstępne:

-

Przebieg ćwiczenia:

1. W folderze `C:\java_cwiczenia` wprowadź polecenie mvn archetype:generate. W efekcie jej wprowadzenia na ekranie powinna pojawić się lista dostępnych archetypów oraz prośba o wybranie numeru archetypu.
   ...
   877: remote -> sk.seges.sesam:sesam-annotation-archetype (-)
   878: remote -> tk.skuro:clojure-maven-archetype (A simple Maven archetype for Clojure)
   879: remote -> uk.ac.rdg.resc:edal-ncwms-based-webapp (-)
   Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 328:

2. Wybierz domyślny numer archetypu wciskając Enter.

3. W tym momencie pojawi się prośba o wybranie wersji archetypu
   Choose org.apache.maven.archetypes:maven-archetype-quickstart version:
   1: 1.0-alpha-1
   2: 1.0-alpha-2
   3: 1.0-alpha-3
   4: 1.0-alpha-4
   5: 1.0
   6: 1.1
   Choose a number: 6:

   Należy wybrać wartość domyślną, wciskając Enter

4. W tym momencie pojawi się prośba o podanie wartości dla groupId
   Define value for property 'groupId': :
   Wpisz pl.wwsis.mvn w ciśnij Enter

5. W tym momencie pojawi się prośba o podanie nazwy artefaktu
   Define value for property 'artifactId': :
   Wpisz projekt1 i wciśnij Enter

6. W tym momencie pojawi się prośba o podanie numeru wersji artefaktu

Define value for property 'version': 1.0-SNAPSHOT: :
Wybierz wartość domyślną wciskając Enter

7. W tym momencie pojawi się prośba o podanie nazwy pakietu
   Define value for property 'package': pl.wwsis.mvn: :
   Zaakceptuj proponowaną nazwę wciskając Enter

8. W tym momencie pojawi się prośba o potwierdzenie konfiguracji
   Confirm properties configuration:
   groupId: pl.wwsis.mvn
   artifactId: projekt1
   version: 1.0-SNAPSHOT
   package: pl.wwsis.mvn
   Y: :
   Nalezy potwierdzić wciskając Enter

9. Na zakończenie Maven wypisze podsumowanie wygenerowanego projektu
   [INFO] --
   [INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.1
   [INFO] --
   [INFO] Parameter: groupId, Value: pl.wwsis.mvn
   [INFO] Parameter: packageName, Value: pl.wwsis.mvn
   [INFO] Parameter: package, Value: pl.wwsis.mvn
   [INFO] Parameter: artifactId, Value: projekt1
   [INFO] Parameter: basedir, Value: C:\java_cwiczenia
   [INFO] Parameter: version, Value: 1.0-SNAPSHOT
   [INFO] project created from Old (1.x) Archetype in dir: C:\java_cwiczenia\projekt1
   [INFO] --
   [INFO] BUILD SUCCESS
   [INFO] --
   [INFO] Total time: 21.483s
   [INFO] Finished at: Thu Nov 21 22:14:27 CET 2013
   [INFO] Final Memory: 7M/26M
   [INFO] --

10. W katalogu C:\java_cwiczenia powinien w tym momencie znajdować się katalog projekt1. Jest to główny katalog stworzonego z archetypu projektu. Powinine on zawierać plik pom.xml i katalog src. Wewnątrz katalogu src odszukaj plik main\java\pl\wwsis\mvn\App.java i zapoznaj się z jego zawartością. Jest to wygenerowany przez Mavena, na podstawie wybranego archetypu plik źródłowy.

ZADANIE 3	Cykl budowania i lokalne repozytorium

Cel ćwiczenia:

- Zapoznanie się z podstawowymi krokami cyklu budowania aplikacji

Przebieg ćwiczenia:

1. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1** **i wprowadź polecenie mvn compile. W jej wyniku Maven rozpoczyna kompilację projektu wyświetlając odpowiednie informacje.**

   . . . . .
   [INFO] Compiling 1 source file to C:\java_cwiczenia\projekt1\target\classes
   [INFO]
   --
   [INFO] BUILD SUCCESS
   [INFO]
   --
   [INFO] Total time: 1.200s
   [INFO] Finished at: Fri Nov 22 19:00:43 CET 2013
   [INFO] Final Memory: 5M/15M
   [INFO]
   --

   Po zakończeniu kompilacji, w folderze **C:\java_cwiczenia\projekt1, obok folderu src, powinien znajdować się również folder target wraz z podfolderem classes. W tym miejscu, znajdują się skompilowane klasy w podfolderach zgodnych z ich pakietami. Odszukaj klasę pl.wwsis.mvn.App. Zwróć też uwagę, że folder classes jest jedynym elementem w folderze target.**

2. **Z poziomu okna linii komend przejdź do folderu C:\java_cwiczenia\projekt1\target\classes a następnie uruchom skompilowaną klasę za pomocą komendy java pl.wwsis.mvn.App. Czy efekt jej uruchomienia odpowiada zawartości pliku źródłowego** C:\java_cwiczenia\projekt1\src\main\java\pl\wwsis\mvn\App.java?
   Powinien.

3. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wprowadź polecenie mvn clean. Jest to faza służąca do czyszczenia projektu z wszelkich produktów wygenerowanych**

przez fazy mavena. Po jej wykonaniu folder C:\java_cwiczenia\projekt1\target powinien zostać usunięty.

4. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wprowadź polecenie mvn package. Faza package w procesie budowania aplikacji służy do pakowania skompilowanych klas do pliku jar i wykonuje się po fazie compile (chociaż nie bezpośrednio). W związku z tym na efekty wykonania tej komendy składają się te wynikające z wykonania fazy compile, i wszystkich innych faz następujących po niej aż do fazy package. Sprawdź zawartość katalogu C:\java_cwiczenia\projekt1\target\. Oprócz folderu classes powinny znajdować się też inne katalogi, będące produktem wykonania faz następujących po fazie compile, oraz plik projekt1-1.0-SNAPSHOT.jar, który jest produktem fazy package.**

5. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wykonaj polecenie mvn clean. Podkatalog target powinien zostać usunięty.**

6. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wykonaj polecenie mvn install. Faza install w cyklu budowania aplikacji służy do instalowania skompilowanego projektu do lokalnego repozytorium i wykonuje się po fazie package (chociaż nie bezpośrenio). Produkty wszystkich poprzedzających ją faz powinny zostać zbudowane - sprawdź zawartość podkatalogu target. Czy jego zawartość jest taka sama jak po wykonaniu mvn package? Powinna.**
   Zwróć uwagę na końcowe linie wyświetlone po wykonaniu mvn install a w szczególności na wskazaną tam lokalizację katalogu .m2 - będzie ona inna niż w poniższym przykładzie.

```
[INFO] --- maven-install-plugin:2.3.1:install (default-install) @ projekt1 ---
[INFO] Installing C:\java_cwiczenia\projekt1\target\projekt1-1.0-SNAPSHOT.jar to
C:\Users\orsn\.m2\repository\pl\wwsis\mvn\projekt1\1.0-SNAPSHOT\projekt1-1.0-SNAPSHOT.jar
[INFO] Installing C:\java_cwiczenia\projekt1\pom.xml to
C:\Users\orsn\.m2\repository\pl\wwsis\mvn\projekt1\1.0-SNAPSHOT\projekt1-1.0-SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Powinna tam znajdować się informacja (zaznaczona powyżej na czerwono) o kopiowaniu określonych plików do lokalnego repozytorium Mavena - katalog .m2\repository. Od tej pory klasy zainstalowanego projektu mogą być używane w innych projektach po zdefiniowaniu odpowiednich zależności. Sprawdź zawartość katalogu .m2\repository a w szczególności znajdującego się tam katalogu pl

ZADANIE 4	Zależności

Cel ćwiczenia:

- Zapoznanie się z mechanizmem zależności Mavena
- Dołączenie do projektu biblioteki Apache Commons Collections (http://commons.apache.org/proper/commons-collections/download_collections.cgi)

Przebieg ćwiczenia:

1. Otwórz w edytorze plik **C:\java_cwiczenia\projekt1\src**main\java\pl\wwsis\mvn\App.java i zastąp jego treść następującym kodem

```
package pl.wwsis.mvn;

import org.apache.commons.collections.buffer.PriorityBuffer;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
          PriorityBuffer buffer = new PriorityBuffer();
        System.out.println( "Hello World!" );
    }
}
```

Do klasy App dodane zostało odwołanie do klasy PriorityBuffer z biblioteki Apache Commons Collections (http://commons.apache.org/proper/commons-collections/download_collections.cgi)

2. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wykonaj polecenie mvn clean compile. Czy projekt się kompiluje? Dlaczego? Co należałoby zrobić by projekt się skompilował?**

1. Otwórz w edytorze plik **C:\java_cwiczenia\projekt1\pom.xml i odszukaj sekcję <dependencies>. Wewnątrz sekcji <dependencies> dodaj następujący wpis**

<dependency>

```
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>3.2.1</version>
    </dependency>
```

2. Zapisz plik pom.xml
3. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt1 i wykonaj polecenie mvn clean compile. Zwróć uwagę na pojawiające się informacje o pobieraniu plików**

```
. . .
 [INFO] Building projekt1 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
Downloading:
http://repo.maven.apache.org/maven2/commons-collections/commons-collections/3.2.1/commo
ns-collections-3.2.1.pom
Downloaded:
http://repo.maven.apache.org/maven2/commons-collections/commons-collections/3.2.1/commo
ns-collections-3.2.1.pom (13 KB at 35.2 KB/sec)
. . .
```

Kompilacja powinna zakończyć się sukcesem. Mechanizm zależności Mavena przed rozpoczęciem procesu budowania projektu pobrał zdefiniowaną w pliku pom.xml bibliotekę i umieścił ją w lokalnym repozytorium - sprawdź zawartość katalogu .m2\repository

ZADANIE 5	Pluginy Mavena na przykładzie plugina servera tomcat6

Cel ćwiczenia:

- Zapoznanie się z pluginem tomcat6 dla Mavena

Przebieg ćwiczenia:

1. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\ stwórz nowy projekt korzystając z mechanizmu archetypów Mavena. Użyj poniższego polecenia:**

```
mvn archetype:generate -DgroupId=pl.wwsis.mvn -DartifactId=projekt2
-DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
```

Na koniec potwierdź wprowadzone dane wciskając Enter. Właśnie został stworzony projekt aplikacji webowej (www) z archetypu mavena.

1. **Przejdź do nowostworzonego katalogu C:\java_cwiczenia\projekt2 i otwórz w edytorze znajdujący się tam plik pom.xml**
2. **W sekcji <build> dodaj następującą treść**

```
<plugins>
      <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat6-maven-plugin</artifactId>
            <version>2.2</version>
      </plugin>
</plugins>
```

3. Z poziomu okna linii komend przejdź do folderu **C:\java_cwiczenia\projekt2 i wykonaj polecenie mvn tomcat6:run. Jest to polecenie uruchomienia pluginu tomcat6 z poleceniem run.**

4. **Po uruchomieniu plugina na końcu długiej listy pojawiających się wiadomości powinna pojawić się informacja o uruchomieniu serwera www tomcat6**

```
[INFO] Running war on http://localhost:8080/projekt2
[INFO] Creating Tomcat server configuration at
C:\java_cwiczenia\projekt2\target\tomcat
2013-11-22 21:05:14 org.apache.catalina.startup.Embedded start
INFO: Starting tomcat server
2013-11-22 21:05:14 org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/6.0.35
2013-11-22 21:05:14 org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-8080
2013-11-22 21:05:14 org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8080
```

5. Otwórz przeglądarkę internetową i przejdź do adresu: http://localhost:8080/projekt2

6. Otwórz w edytorze plik C:\java_cwiczenia\projekt2\src\main\webapp\index.jsp

7. Zmień znajdujący się tam komunikat "Hello World" a następnie odśwież stronę http://localhost:8080/projekt2 w przeglądarce www.