Hogyan futtatható Docker környezetben egy Maven projekt?

Használjuk a docker playground-ot: http://docker.iit.uni-miskolc.hu

Klónozzuk a *Wildfly* alkalmazás szerver példáit:

```
git clone https://github.com/wildfly/quickstart.git
cd quickstart/helloworld
```

Docker segítségével lefordíthatunk egy tetszőleges példát:

```
docker run -it --rm --name helloworld -v "$(pwd)":/usr/src/helloworld -w
/usr/src/helloworld maven:3.8.7-openjdk-18-slim mvn clean install
```

A parancs magyarázata az alábbi:

- **docker run**: Ez a parancs futtatja a Docker konténert.
- **-it**: Ez a kapcsoló lehetővé teszi az interaktív módot és csatlakoztatja a terminált a konténerhez, hogy kommunikálni lehessen vele.
- **–rm**: Ez a kapcsoló jelzi a Dockernek, hogy törölje a konténert, mielőtt újraindítjuk. Ezzel elérhetjük, hogy a próbálgatások során újra létrejön a tároló újra és nem mentődik el az előző állapot.
- **–name helloworld**: Ez a kapcsoló nevet ad a konténernek. Ebben az esetben a konténer neve “helloworld” lesz.
- **-v “$(pwd)”:/usr/src/helloworld**: Ez a kapcsoló köti össze a jelenlegi munkakönyvtárat (ahol a Docker parancsot futtatják) a konténerben lévő /usr/src/helloworld mappával. Ez azt jelenti, hogy a jelenlegi munkakönyvtár tartalma elérhető lesz a konténerben.
- **-w /usr/src/helloworld**: Ez a kapcsoló beállítja a munkakönyvtárat a konténerben a /usr/src/helloworld mappára. Ez azt jelenti, hogy a következő parancs a konténerben ebben a mappában fog végrehajtódni.
- **maven:3.8.7-openjdk-18-slim**: Ez a Docker image, amelyet a konténer alapként használ. Itt a maven:3.8.7-openjdk-18-slim image-t használja, amely tartalmazza a Maven-t és az OpenJDK 18-at.
- **mvn clean install**: Ez a parancs futtatódik a konténerben. Itt a Maven-t indítja el a konténerben a “clean install” céllal. Ez a Maven parancs kitörli az előző fordítási eredményeket, majd újrafordítja és telepíti a projektet.

Hozzunk létre egy másik Dockerfile-t a ./target könyvtárban:

```
FROM quay.io/wildfly/wildfly
ADD helloworld.war /opt/jboss/wildfly/standalone/deployments
```

Indítsuk el az alábbi két parancsot:

Az első létrehozza a kontainert, a második pedig elindítja.

```
docker build -t quay.io/wildfly/wildfly .
docker run -p 8080:8080 quay.io/wildfly/wildfly
```

“Open port” gomb megnyomása után a 8080-as portot kérjük megnyitni, majd az url.hez írjuk hozzá: /helloworld/

Ugyanez a feladat hogyan oldható meg docker-compose.yml használatával?

Ha egy új konténert is indítunk, akkor a *git clone*-ról se feledkezzünk meg:

```
git clone https://github.com/wildfly/quickstart.git
cd quickstart
```

A docker-compose.yml rugalmasabb megoldást ad, mert nem kell több parancsot megjegyezni, hanem egy fájlban kezelhetjük a beállításokat.

Tehát a *quickstart* könyvtárban hozzuk létre az alábbi *docker-compose.yml*-t:

```
version: '3'
services:
    compiler:
        image: maven:3.8.7-openjdk-18-slim
        volumes:
          - ./helloworld:/usr/src/helloworld
        working_dir: /usr/src/helloworld
        command: mvn clean install
```

A fenti megoldás sem eléggé rugalmas, mert a helloword-on kívül más példát is el szeretnénk indítani. Menjünk vissza a *quickstart* könyvtárba és hozzunk létre egy *.env* fájlt és tároljuk el környezeti változóként az elérési utat:

Az .env állományt a beépített editor nem listázza így vi-al is tudjuk szerkeszteni.

```
EXAMPLE_FOLDER=helloworld
```

Készítsünk egy Dockerfile-t az alábbi tartalommal:

```
FROM quay.io/wildfly/wildfly
ADD ./$EXAMPLE_FOLDER/target/$EXAMPLE_FOLDER.war
/opt/jboss/wildfly/standalone/deployments
```

A *docker-compose.yml* az alábbi lesz:

```
version: '3'
services:
    compiler:
        image: maven:3.8.7-openjdk-18-slim
        volumes:
```

```
            - ./${EXAMPLE_FOLDER}:/usr/src/${EXAMPLE_FOLDER}
        working_dir: /usr/src/${EXAMPLE_FOLDER}
        command: mvn clean install
    wildfly:
        build: .
        environment:
            - "EXAMPLE_FOLDER=helloworld"
        ports:
          - 8080:8080
        depends_on:
          - compiler
```

Indítsuk el a *docker-compose up* paranncsal.

Megjegyzés: Ha magunk hozzuk létre a *Dockerfile*-t vagy *docker-compose.yml*-t akkor óhatatlanul is elronthatjuk, ilyenkor a következő paranccsal lehet újrafordítani:

```
docker-compose build --no-cache
```

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:wildfly_in_docker

Last update: **2023/05/04 12:03**