Session 2 Handout — Docker / Podman, Spring

1) Docker / Podman

Test if Docker / Podman is running

```
docker run --rm hello-world
```

Building my application

```
docker build -t myapp .
```

Check if image is present

```
docker images
```

Running containers

```
# Forward host OS port 1234 to container 8080 port
docker run -p 1234:8080 myapp

# Set environment variables
docker run -e SERVER_PORT=9999 -p 1234:9999 myapp

# Mounting a host OS folder into the container
docker run -v /host/folder:/container/folder myapp
docker run -v /host/folder:/container/folder:ro myapp
```

Check if container is running

```
docker ps
```

Stopping containers

```
docker stop [container id]
```

Manipulating with containers

```
docker save -o myapp.tar myapp

docker load -i myapp.tar
```

2) Setup and manage environment for Spring

Copy file to remote server via SSH

```
scp spring.zip student[n]@91.98.156.163:/home/student[n]/spring
```

Run application via Maven

```
mvn spring-boot:run
```

Check environment port variables

```
env | grep -i port
```

3) Spring annotations

Service types

- @Component -> the basis
- @Service -> extends @Component, used for service classes
- @Controller -> extends @Component, used for REST controllers
- @Repository -> extends @Component, used for data access repositories

Dependency Injection

- @Autowired
- @Scope
    - singleton
    - prototype
    - request
    - session
    - application

Controllers

- @RestController

- @GetMapping -> accepts path -> @GetMapping("/hello")
- @PostMapping
- @PutMapping
- @DeleteMapping
- ...

Persistence

- @Entity - define an entity class
- @Id - annotates primary key field in an entity class
- @GeneratedValue(strategy = GenerationType.IDENTITY) -> generated field with strategy type set
- @Column -> defines a given member as column in the generated database
- @Column(length = 2048) -> sets the size of the column. Name can be also set.

4) Lombok annotations

@AllArgsConstructor
@NoArgsConstructor
@Getters
@Setters
@Data -> combination of getters, setters, constructor types
@UtilityClass -> creates a final class with private constructor

5) Code examples

pom.xml for legacy app

Add plugins for setting the main class and copying dependency jars into a target directory

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>3.6.1</version>
      <executions>
        <execution>
          <id>copy-runtime-deps</id>
          <phase>package</phase>
          <goals><goal>copy-dependencies</goal></goals>
          <configuration>
```

```
<outputDirectory>${project.build.directory}/deps</outputDirectory>
              <includeScope>runtime</includeScope>
              <excludeTransitive>false</excludeTransitive>
            </configuration>
          </execution>
        </executions>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.4.2</version>
        <configuration>
          <archive>
            <manifest>
              <mainClass>com.ibm.example.App</mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
```

Application properties

```
spring.application.name=spring

spring.datasource.url=jdbc:h2:mem:todo;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FA
LSE;MODE=PostgreSQL
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver

spring.h2.console.enabled=true
spring.h2.console.settings.web-allow-others=true

spring.jpa.hibernate.ddl-auto=validate
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true
```

Hello Controller

```
@RestController
public class HelloController {
```

```
    @GetMapping("/hello")
    public String hello(@RequestHeader Map<String,String> headers) {
        return "Hello, World!";
    }
}
```

Log entry entity

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class LogEntriesEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 2048)
    private String message;

    @Column
    private String timestamp;
}
```

Log entry repository

```
public interface LogEntriesRepository extends
JpaRepository<LogEntriesEntity, Long> {

}
```

Logging service

```
@Service
public class LoggingService {
  private DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-
MM-dd HH:mm:ss");
  private LogEntriesRepository logRepo;

  @Autowired
  public LoggingService(LogEntriesRepsitory logRepo) {
    this.logRepo = logRepo;
```

```
  }

  public void log(String message) {
    LogEntriesEntity entity = new LogEntriesEntity();
    entity.setMessage(message);
    entity.setTimestamp(LocalDateTime.now().toString());
    entity = logRepo.save(entity);
    System.out.println("Entity saved with id: " + entity.getId());
  }
}
```