Spring advanced Taken

Inhoudsopgave

1 TAKEN

1.1 Weekend

Je maakt een website in twee talen: Nederlands (standaard taal) en Engels.

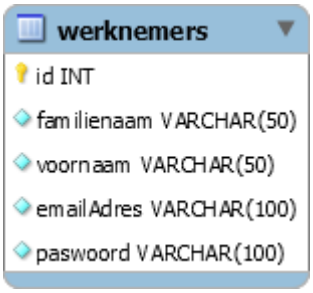
Je bepaalt de taal van de gebruiker aan de hand van de request header `Accept-Language`.

Als het weekend is toon je aan een Nederlandstalige gebruiker: `Weekend, tijd voor kwaliteit`.
Je toont aan een Engelstalige gebruiker: `Weekend, quality time`.

Als het werkdag is toon je aan een Nederlandstalige gebruiker: `Werkdag, centjes verzamelen`.
Je toont aan een Engelstalige gebruiker: `Working day, make some money`.

1.2 Firma

Je maakt met het script `firma.sql` de database `firma`. De database bevat één table:



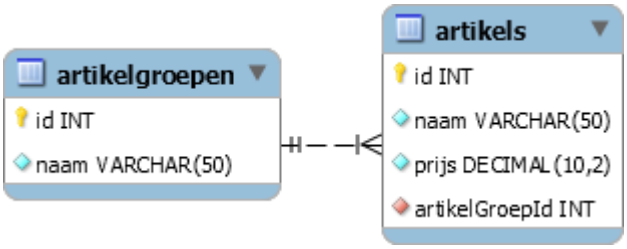
Het paswoord is bij elke werknemer `zorro`.

Je maakt een website met een welkompagina en een tweede pagina.
Je toont in de tweede pagina het geluksgetal 7.

Alle gebruikers mogen de welkompagina zien.
Als gebruiker de tweede pagina openen, moeten ze eerst inloggen.
Ze loggen in met hun email adres als gebruikersnaam.
Alle gebruikers hebben één rol: `gebruiker`.

1.3 Artikels en artikelgroepen

Je maakt met het script `artikelsEnArtikelGroepen.sql`
de database `artikelsenartikelgroepen`. De database bevat twee tables:



De tables zijn leeg.

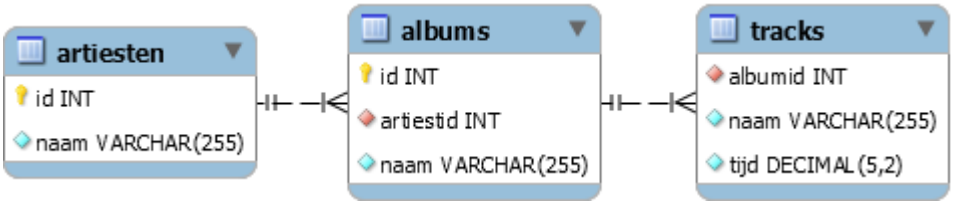
Je maakt een project met een repository layer.
Die bevat drie methods, zodat je volgende vragen kan beantwoorden:

1. Wat zijn de artikels met een prijs tussen een minimum waarde en een maximum waarde ?
2. Wat is de hoogste artikelprijs ?
3. Wat zijn de artikels met een artikelgroep met een bepaalde naam ?

Je test de methods met Junit.

1.4 Muziek

Je voert het script `Muziek.sql` uit. Dit script maakt een database `muziek`:



Je maakt een website.

Een request naar http://localhost:8080/albums/1 geeft volgende data:

```
{
  "album": "Retro - John McCready FAN",
  "artiest": "New Order",
  "_links": {
    "self": {
      "href": "http://localhost:8080/albums/1"
    },
    "tracks": {
      "href": "http://localhost:8080/albums/1/tracks"
    }
  }
}
```

Een request naar http://localhost:8080/albums/1/tracks geeft volgende data:

```
[
  {
    "naam": "Lonesome Tonight",
    "tijd": "00:05:19"
  },
  {
    "naam": "In A Lonely Place",
    "tijd": "00:06:26"
  },
  …
]
```

1.5 Muziek test

Je test met Junit of een request naar `/albums/{id}` een response met status code `404` geeft bij een onbestaande id. Je test of je een correcte response krijgt bij een bestaande id.

1.6 Temperatuur

Je maakt een website. Deze bevat een rest client class. De class bevat een method.

De method heeft de naam van een gemeente als parameter.

De method doet een `GET` request naar
`http://api.openweathermap.org/data/2.5/weather?q=xxx&units=metric&APPID=yyy`

- Je vervangt `xxx` door de gemeente.
- Je vervangt `yyy` door je API key.
  Je krijgt die door een account te maken op `https://openweathermap.org`.

De method geeft de temperatuur als een `Optional<BigDecimal>` terug.
De `Optional` is leeg als je geen temperatuur vindt voor de gevraagde gemeente.

Je test met Junit dat deze temperatuur ligt tussen `-60` en `60`.

1.7 Muziek client

Je maakt een statische website.

De gebruiker tikt een nummer van een album. Je doet `GET` requests naar de website die je vroeger in de taken maakte om de naam van het album, de artiest en de tracks te tonen.

1.8 Ondernemingsnummer

De class Firma bevat een variabele ondernemingsNummer.

Een ondernemingsnummer bevat een controle:
de laatste 2 cijfers moeten gelijk zijn aan `97` – de rest van de deling van de overige cijfers door `97`.
Een voorbeeld van een correct ondernemingsnummer: `426388541`.

Je schrijf de validation annotation `@OndernemingsNummer`.

Je bewijst met een unit test dat deze annotation corect werkt.

```
package be.vdab.ondernemingsnummer.domain;
import be.vdab.ondernemingsnummer.constraints.OndernemingsNummer;
public class Firma {
  @OndernemingsNummer
  private long ondernemingsNummer;
  // getter en setter
}
```

1.9 Docker

Je start een container met MySQL. Je maakt daarin met het script `muziek.sql` de database `muziek`.

Je voert het project `muziek` uit in een container.
Je verbindt het TCP/IP poort `8082` op je computer met de TCP/IP poort `8080` in de container.

Je doet met Postman een `GET` request naar `localhost:8082/albums/1`.
Als alles goed werkt, krijg je een response met JSON data over album `1`.

1.10 Sportwinkel

Je voert het script `sportwinkel.sql` uit. Dit script maakt een database `sportwinkel`:

Je maakt een website.

Hij verwerkt POST request met volgende JSON data in de request body:

```
{"naam": "racefiets", "aankoopprijs": 3000, "verkoopprijs": 4000 }
```

Je voegt met deze data een record toe aan de table `artikels`.

Je maakt in RabbitMQ een fanout exchange met de naam `sportartikels`.
Je maakt een queue met de naam `sportartikels`.
Je verbindt die met de exchange `sportartikels`.

Je stuurt ook een message naar een RabbitMQ exchange met de naam `sportartikels`.
De message bevat volgende JSON data (id is de id van het nieuwe record in de table `artikels`):

```
{"id": 1, "naam": "racefiets" }
```

1.11 Sportmailing

Je voert het script `sportmailing.sql` uit. Dit script maakt een database `sportmailing`:

Je voegt met de MySQL Workbench één of meerdere van je email adressen toe.

Je maakt een wesite.

Hij verwerkt boodschappen uit de RabbitMQ queue `sportartikels`.
Je stuurt per bericht een mail naar elke sporter uit de table `sporters`.
De mail bevat de tekst `Er is een nieuw artikel: racefiets`.

1.12 Outbox

Je past het outbox pattern toe op de applicatie sportwinkel.

2 VOORBEELDOPLOSSINGEN

2.1 Weekend

2.1.1 messages.properties

```
weekend=Weekend, tijd voor kwaliteit
werkdag=Werkdag, centjes verzamelen
```

2.1.2 messages_en.properties

```
weekend=Weekend, quality time
werkdag=Working day, make some money
```

2.1.3 IndexController

```
package be.vdab.weekend.controllers;
// enkele imports
@Controller
@RequestMapping("/")
class IndexController {
  @GetMapping
  public ModelAndView index() {
    var modelAndView = new ModelAndView("index");
    var dagVanWeek = LocalDate.now().getDayOfWeek();
    var weekend = dagVanWeek == DayOfWeek.SATURDAY || dagVanWeek == DayOfWeek.SUNDAY;
    modelAndView.addObject("weekend", weekend);
    return modelAndView;
  }
}
```

2.1.4 index.html

```
<!doctype html>
<html lang="nl" xmlns:th="http://www.thymeleaf.org">
  <head>
    <title th:text="Weekend"></title>
  </head>
  <body>
    <h1 th:if="${weekend}" th:text="#{weekend}"></h1>
    <h1 th:if="not ${weekend}" th:text="#{werkdag}"></h1>
  </body>
</html>
```

2.2 Firma

2.2.1 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/firma
spring.datasource.username=cursist
spring.datasource.password=cursist
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
```

2.2.2 IndexController

```
package be.vdab.firma.controllers;
// enkele imports
@Controller
@RequestMapping("/")
class IndexController {
  @GetMapping
  public String index() {
    return "index";
  }
}
```

2.2.3 index.html

```
<!doctype html>
<html lang="nl">
  <head>
    <title>Welkom</title>
  </head>
  <body>
    <h1>Welkom</h1>
  </body>
</html>
```

2.2.4 GeluksController

```
package be.vdab.firma.controllers;
// enkele imports
@Controller
@RequestMapping("geluk")
class GeluksController {
  @GetMapping
  public String geluk() {
    return "geluk";
  }
}
```

2.2.5 geluk.html

```
<!doctype html>
<html lang="nl">
  <head>
    <title>Geluk</title>
  </head>
  <body>
    <h1>7 is het geluk</h1>
  </body>
</html>
```

2.2.6 SecurityConfig

```
package be.vdab.firma.security;
// enkele imports
@EnableWebSecurity
class SecurityConfig extends WebSecurityConfigurerAdapter {
  private final DataSource datasource;
  SecurityConfig(DataSource datasource) {
    this.datasource = datasource;
  }
  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http.formLogin();
    http.authorizeRequests(requests -> requests
      .mvcMatchers("/geluk").hasAuthority("gebruiker"));
  }
  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication().dataSource(datasource)
     .usersByUsernameQuery(
        "select emailAdres as username, paswoord as password, true as enabled" +
        " from werknemers where emailAdres = ?")
     .authoritiesByUsernameQuery("select ?, 'gebruiker'");
  }
}
```

2.3 Artikels en artikelgroepen

2.3.1 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/artikelsenartikelgroepen
spring.datasource.username=cursist
spring.datasource.password=cursist
spring.test.database.replace=none
logging.level.org.hibernate.SQL=DEBUG
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
```

2.3.2 ArtikelGroep

```
package be.vdab.artikelsenartikelgroepen.domain;
// enkele imports
@Entity
@Table(name="artikelgroepen")
public class ArtikelGroep {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String naam;
  // getters
}
```

2.3.3 Artikel

```
package be.vdab.artikelsenartikelgroepen.domain;
// enkele imports
@Entity
@Table(name="artikels")
public class Artikel {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String naam;
  private BigDecimal prijs;
  @ManyToOne(fetch = FetchType.LAZY, optional = false)
  @JoinColumn(name = "artikelgroepid")
  private ArtikelGroep artikelGroep;
  // getters
}
```

2.3.4 ArtikelRepository

```
package be.vdab.artikelsenartikelgroepen.repositories;
// enkele imports
public interface ArtikelRepository extends JpaRepository<Artikel, Long> {
  List<Artikel> findByPrijsBetween(BigDecimal van, BigDecimal tot);
  @Query("select max(a.prijs) from Artikel a")
  BigDecimal findHoogstePrijs();
  List<Artikel> findByArtikelGroepNaam(String naam);
}
```

2.3.5 insertArtikelGroepen.sql

```
insert into artikelgroepen(naam) values('groep1');
insert into artikelgroepen(naam) values('groep2');
```

2.3.6 insertArtikels.sql

```
insert into artikels(naam, prijs, artikelGroepId)
values ('artikel1', 10, (select id from artikelgroepen where naam = 'groep1'));
insert into artikels(naam, prijs, artikelGroepId)
values ('artikel2', 20, (select id from artikelgroepen where naam = 'groep1'));
insert into artikels(naam, prijs, artikelGroepId)
values ('artikel3', 30, (select id from artikelgroepen where naam = 'groep2'));
```

2.3.7 spring.properties

```
spring.test.constructor.autowire.mode=all
```

2.3.8 ArtikelRepositoryTest

```
package be.vdab.artikelsenartikelgroepen.repositories;
// enkele imports
@DataJpaTest
@Sql("/insertArtikelGroepen.sql")
@Sql("/insertArtikels.sql")
class ArtikelRepositoryTest {
  private final ArtikelRepository repository;
  // constructor met parameter
  @Test
  void findByPrijsBetween() {
    var van = BigDecimal.TEN;
    var tot = BigDecimal.valueOf(20);
    assertThat(repository.findByPrijsBetween(van, tot))
      .hasSize(2)
      .extracting(artikel -> assertThat(artikel.getPrijs()).isBetween(van,tot));
  }
  @Test
  void findHoogstePrijs() {
    assertThat(repository.findHoogstePrijs()).isEqualByComparingTo("30");
  }
  @Test
  void findByArtikelGroepNaam() {
    var groep = "groep2";
    assertThat(repository.findByArtikelGroepNaam(groep))
      .hasSize(1)
      .extracting(artikel ->
        assertThat(artikel.getArtikelGroep().getNaam()).isEqualTo(groep));
  }
}
```

2.4 Muziek

2.4.1 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/muziek
spring.datasource.username=cursist
spring.datasource.password=cursist
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.naming.physical-strategy=\
org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
```

2.4.2 Artiest

```
package be.vdab.muziek.domain;
// enkele imports ...
@Entity
@Table(name = "artiesten")
public class Artiest {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String naam;
  // getters
}
```

2.4.3 Album

```
package be.vdab.muziek.domain;
// enkele imports ...
@Entity
@Table(name = "albums")
@NamedAttributeNode("artiest"))
public class Album {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String naam;
  @ManyToOne(optional = false, fetch = FetchType.LAZY)
  @JoinColumn(name = "artiestid")
  private Artiest artiest;
  @ElementCollection
  @CollectionTable(name = "tracks", joinColumns = @JoinColumn(name = "albumid"))
  private Set<Track> tracks = new LinkedHashSet<>();
  // getters voor id, naam en artiest
  public Set<Track> getTracks() {
    return Collections.unmodifiableSet(tracks);
  }
}
```

2.4.4 Track

```
package be.vdab.muziek.domain;
// enkele imports ...
@Embeddable
@Access(AccessType.FIELD)
public class Track {
  private String naam;
  private LocalTime tijd;
  // getters
  @Override
  public int hashCode() {
    return naam.toUpperCase().hashCode();
  }
  @Override
  public boolean equals(Object object) {
    if ((object instanceof Track)) {
      var that = (Track) object;
      return naam.equalsIgnoreCase(that.naam);
    }
    return false;
  }
}
```

2.4.5 AlbumRepository

```
package be.vdab.muziek.repositories;
// enkele imports ...
public interface AlbumRepository extends JpaRepository<Album, Long> {
}
```

2.4.6 AlbumService

```
package be.vdab.muziek.services;
// enkele imports ...
public interface AlbumService {
  Optional<Album> findById(long id);
}
```

2.4.7 DefaultAlbumService

```
package be.vdab.muziek.services;
// enkele imports ...
@Service
@Transactional
class DefaultAlbumService implements AlbumService {
  private final AlbumRepository albumRepository;
  // constructor met parameter
  @Override
  @Transactional(readOnly = true)
  public Optional<Album> findById(long id) {
    return albumRepository.findById(id);
  }
}
```

2.4.8 AlbumArtiest

```
package be.vdab.muziek.dto;
public class AlbumArtiest {
  private final String album;
  private final String artiest;
  public AlbumArtiest(Album album) {
    this.album = album.getNaam();
    this.artiest = album.getArtiest().getNaam();
  }
  // getters
}
```

2.4.9 AlbumNietGevondenException

```
package be.vdab.muziek.exceptions;
public class AlbumNietGevondenException extends RuntimeException {
  private static final long serialVersionUID = 1L;
}
```

2.4.10 AlbumController

```
package be.vdab.muziek.restcontrollers;
// enkele imports ...
@RestController
@RequestMapping("albums")
@ExposesResourceFor(Album.class)
class AlbumController {
  private final AlbumService albumService;
  private final EntityLinks entityLinks;
  // constructor met parameters
  @GetMapping("{id}")
  EntityModel<AlbumArtiest> getAlbum(@PathVariable long id) {
    var optionalAlbum = albumService.findById(id);
    if (optionalAlbum.isPresent()) {
      var album = optionalAlbum.get();
      var albumArtiest = new AlbumArtiest(album);
      var model = new EntityModel<>(albumArtiest);
      model.add(entityLinks.linkToItemResource(Album.class, id));
      model.add(entityLinks.linkForItemResource(Album.class, id)
        .slash("tracks").withRel("tracks"));
      return model;
    }
    throw new AlbumNietGevondenException();
  }
```

```
  @GetMapping("{id}/tracks")
  Set<Track> getTracks(@PathVariable long id) {
    var optionalAlbum = albumService.findById(id);
    if (optionalAlbum.isPresent()) {
      return optionalAlbum.get().getTracks();
    }
    throw new AlbumNietGevondenException();
  }
  @ExceptionHandler(AlbumNietGevondenException.class)
  @ResponseStatus(HttpStatus.NOT_FOUND)
  void albumNietGevonden() {
  }
}
```

2.5 Muziek test

2.5.1 spring.properties

```
spring.test.constructor.autowire.mode=all
```

2.5.2 insertArtiest.sql

```
insert into artiesten(naam) values ('test');
```

2.5.3 insertAlbum.sql

```
insert into albums(naam, artiestid)
values ('test', (select id from artiesten where naam='test'));
```

2.5.4 AlbumControllerTest

```
package be.vdab.muziek.restcontrollers;
// enkele imports
@SpringBootTest
@AutoConfigureMockMvc
@Sql("/insertArtiest.sql")
@Sql("/insertAlbum.sql")
class AlbumControllerTest extends AbstractTransactionalJUnit4SpringContextTests
{
  private final MockMvc mvc;
  // constructor met parameter
  private long idVanTestAlbum() {
    return super.jdbcTemplate.queryForObject(
      "select id from albums where naam='test'", Long.class);
  }
  @Test
  void onbestaandAlbumLezen() throws Exception {
     mvc.perform(get("/albums/{id}", -1))
      .andExpect(status().isNotFound());
  }
  @Test
  void albumLezen() throws Exception {
    var id = idVanTestAlbum();
    mvc.perform(get("/albums/{id}", id))
      .andExpect(status().isOk())
      .andExpect(jsonPath("album").value("test"));
  }
}
```

2.6 Temperatuur

2.6.1 application.properties

```
openweathermapapi=\
https://api.openweathermap.org/data/2.5/weather?q={gemeente}&units=metric&APPID=yyy
```

2.6.2 Main

```
package be.vdab.temperatuur.dto;
import java.math.BigDecimal;
public class Main {
  private BigDecimal temp;
  // getter
}
```

2.6.3 OpenWeatherMap

```
package be.vdab.temperatuur.dto;
public class OpenWeatherMap {
  private Main main;
  // getter
}
```

2.6.4 WeatherClient

```
package be.vdab.temperatuur.restclients;
// enkele imports
public interface WeatherClient {
  Optional<BigDecimal> getTemperatuur(String gemeente);
}
```

2.6.5 OpenWeatherMapClient

```
package be.vdab.temperatuur.restclients;
// enkele imports
@Component
class OpenWeatherMapClient implements WeatherClient {
  private final WebClient client;
  private final String uri;
  OpenWeatherMapClient(WebClient.Builder builder,
    @Value("${openweathermapapi}") String uri) {
    client = builder.build();
    this.uri = uri;
  }
  @Override
  public Optional<BigDecimal> getTemperatuur(String gemeente) {
    try {
      return Optional.of(client.get()
        .uri(uri, uriBuilder->uriBuilder.build(gemeente))
       .retrieve().bodyToMono(OpenWeatherMap.class).block()
       .getMain().getTemp());
    } catch (WebClientResponseException.NotFound ex) {
      return Optional.empty();
    }
  }
}
```

2.6.6 spring.properties

```
spring.test.constructor.autowire.mode=all
```

2.6.7 OpenWeatherMapClientTest

```
package be.vdab.temperatuur.restclients;
// enkele imports
@SpringBootTest
class OpenWeatherMapClientTest {
  private final OpenWeatherMapClient client;
  // constructor met parameter
```

```
  @Test
  void temperatuurInBrussel() {
    assertThat(client.getTemperatuur("Brussel").get())
      .isBetween(BigDecimal.valueOf(-60), BigDecimal.valueOf(60));
  }
  @Test
  void onbestaandeGemeente() {
    assertThat(client.getTemperatuur("xxx")).isEmpty();
  }
}
```

2.7 Muziek client

2.7.1 AlbumController (project Muziek)

Extra regel voor class:

```
@CrossOrigin
```

2.7.2 muziek.css

```
body {
  font-family: sans-serif;
}
.fout {
  color:red;
  display:none;
}
```

2.7.3 index.html

```
<!doctype html>
<html lang="nl">
  <head>
    <script src="index.js" defer></script>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="muziek.css">
    <title>Album</title>
  </head>
  <body>
    <label>Nummer album:
      <input id="nummer" autofocus required type="number" min="1">
    </label>
    <button id="zoeken">Zoeken</button>
    <h1 id="album"></h1>
    <ul id="tracks">
    </ul>
    <div id="technischeFout" class="fout">Technische fout.</div>
    <div id="albumNietGevondenFout" class="fout">Album niet gevonden.</div>
  </body>
</html>
```

2.7.4 index.js

```
"use strict";
document.getElementById("zoeken").onclick = zoekAlbum;
async function zoekAlbum() {
  verbergFouten();
  const albumResponse = await fetch("http://localhost:8080/albums/" +
    document.getElementById("nummer").value);
  if (albumResponse.ok) {
    const album = await albumResponse.json();
    document.getElementById("album").innerText =
      `${album.album} - ${album.artiest}`;
    const ul = document.getElementById("tracks");
    while (ul.lastChild !== null) {
      ul.lastChild.remove();
    }
    const tracksResponse = await fetch(album._links.tracks.href);
    if (tracksResponse.ok) {
      const tracks = await tracksResponse.json();
      for (const track of tracks) {
        const li = document.createElement("li");
          li.innerText = `${track.naam} ${track.tijd}`;
          ul.appendChild(li);
        }
    } else {
      document.getElementById("technischeFout").style.display = "block";
    }
  } else {
    if (albumResponse.status === 404) {
      document.getElementById("albumNietGevondenFout").style.display = "block";
    } else {
      document.getElementById("technischeFout").style.display = "block";
    }
  }
}
function verbergFouten() {
  for (const div of document.querySelectorAll(".fout")) {
    div.style.display = "none";
  }
}
```

2.8 Ondernemingsnummer

2.8.1 Firma

Zie opgave

2.8.2 OndernemingsNummer

```
package be.vdab.ondernemingsnummer.constraints;
// enkele imports
@Target({METHOD, FIELD, ANNOTATION_TYPE})
@Retention(RUNTIME)
@Constraint(validatedBy = OndernemingsNummerValidator.class)
public @interface OndernemingsNummer {
  String message() default "{be.vdab.OndernemingsNummer.message}";
  Class<?>[] groups() default {};
  Class<? extends Payload>[] payload() default {};
}
```

2.8.3 OndernemingsNummerValidator

```
package be.vdab.ondernemingsnummer.constraints;
// enkele imports
public class OndernemingsNummerValidator
  implements ConstraintValidator<OndernemingsNummer, Long> {
  @Override
  public void initialize(OndernemingsNummer constraintAnnotation) {
  }
  @Override
  public boolean isValid(Long value, ConstraintValidatorContext context) {
    if (value == null) {
      return true;
    }
    var laatste2Cijfers = value % 100;
    var overigeCijfers = value / 100;
    return laatste2Cijfers == 97 - overigeCijfers % 97;
  }
}
```

2.8.4 FirmaTest

```
package be.vdab.ondernemingsnummer.domain;
// enkele imports
class FirmaTest {
  private Validator validator;
  private Firma firma;
  @BeforeEach
  void beforeEach() {
    ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
    validator = factory.getValidator();
    firma = new Firma();
  }
  @Test void correctOndernemingsNummer() {
    firma.setOndernemingsNummer(426388541);
    assertThat(validator.validate(firma)).isEmpty();
  }
  @Test void verkeerdOndernemingsNummer() {
    firma.setOndernemingsNummer(426388540);
    assertThat(validator.validate(firma)).isNotEmpty();
  }
}
```

2.9 Docker

2.9.1 Netwerk

```
docker network create muzieknet
```

2.9.2 Container met database

```
docker run -d -p 3308:3306 -e MYSQL_ROOT_PASSWORD=vdab --rm --name muziekdb
 --mount source=muziekdb,target=/var/lib/mysql --network muzieknet mysql
```

Je verbindt de MySQL Workbench met deze MySQL. Je voert het script `muziek.sql` uit.

2.9.3 pom.xml

Extra regels na `<artifactId>spring-boot-maven-plugin</artifactId>`:

```
<configuration>
  <image>
    <name>muziek</name>                                                    ❶
  </image>
</configuration>
```

2.9.4 Container met je applicatie

```
docker run -d -p 8082:8080 --rm --name muziek --network muzieknet
 -e spring.datasource.url=jdbc:mysql://muziekdb/muziek muziek
```

2.10 Sportwinkel

2.10.1 application.properties

```
spring.datasource.url=jdbc:mysql://localhost/sportwinkel
spring.datasource.username=cursist
spring.datasource.password=cursist
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
```

2.10.2 Artikel

```
package be.vdab.sportwinkel.domain;
// enkele imports
@Entity
@Table(name = "artikels")
public class Artikel {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String naam;
  private BigDecimal aankoopprijs;
  private BigDecimal verkoopprijs;
    // getters
}
```

2.10.3 ArtikelRepository

```
package be.vdab.sportwinkel.repositories;
// enkele imports
public interface ArtikelRepository extends JpaRepository<Artikel, Long> {
}
```

2.10.4 ArtikelGemaakt

```
package be.vdab.sportwinkel.events;
import be.vdab.sportwinkel.domain.Artikel;
public class ArtikelGemaakt {
  private final long id;
  private final String naam;
  public ArtikelGemaakt(Artikel artikel) {
    id = artikel.getId();
    naam = artikel.getNaam();
  }
  // getters
}
```

2.10.5 ArtikelService

```
package be.vdab.sportwinkel.services;
import be.vdab.sportwinkel.domain.Artikel;
public interface ArtikelService {
  void create(Artikel artikel);
}
```

2.10.6 DefaultArtikelService

```
package be.vdab.sportwinkel.services;
// enkele imports
@Service
@Transactional
class DefaultArtikelService implements ArtikelService {
  private final ArtikelRepository repository;
  private final AmqpTemplate template;
  //constructor met parameters
  @Override
  public void create(Artikel artikel) {
    repository.save(artikel);
    template.convertAndSend("sportartikels", null, new ArtikelGemaakt(artikel));
  }
}
```

2.10.7 ArtikelController

```
package be.vdab.sportwinkel.restcontrollers;
// enkele imports
@RestController
@RequestMapping("/artikels")
class ArtikelController {
  private final ArtikelService artikelService;
  //constructor met parameter
  @PostMapping
  @ResponseStatus(HttpStatus.CREATED)
  void post(@RequestBody Artikel artikel) {
    artikelService.create(artikel);
  }
}
```

2.10.8 SportwinkelApplication

Extra code:

```
@Bean
Jackson2JsonMessageConverter converter() {
  return new Jackson2JsonMessageConverter();
}
```

2.11 Sportmailing

2.11.1 application.properties

```
server.port=8081
spring.mail.host=smtp.gmail.com
spring.mail.port=465
spring.mail.protocol=smtps
spring.mail.username=desmethans@gmail.com
spring.mail.password=blondeeltje2006
spring.datasource.url=jdbc:mysql://localhost/sportmailing
spring.datasource.username=cursist
spring.datasource.password=cursist
spring.datasource.hikari.transaction-isolation=TRANSACTION_READ_COMMITTED
spring.jpa.hibernate.naming.physical-strategy=\
org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

2.11.2 Sporter

```
package be.vdab.sportmailing.domain;
// enkele imports
@Entity
@Table(name="sporters")
public class Sporter {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private long id;
  private String emailAdres;
  // getters
}
```

2.11.3 SporterRepository

```
package be.vdab.sportmailing.repositories;
// enkele imports
public interface SporterRepository extends JpaRepository<Sporter, Long> {
}
```

2.11.4 SporterService

```
package be.vdab.sportmailing.services;
// enkele imports
public interface SporterService {
  List<Sporter> findAll();
}
```

2.11.5 DefaultSporterService

```
package be.vdab.sportmailing.services;
// enkele imports
@Component
@Transactional
class DefaultSporterService implements  SporterService {
  private final SporterRepository repository;
  // constructor met parameter
  @Override
  @Transactional(readOnly = true)
  public List<Sporter> findAll() {
    return repository.findAll();
  }
}
```

2.11.6 ArtikelGemaakt

```
package be.vdab.sportmailing.events;
public class ArtikelGemaakt {
  private long id;
  private String naam;
  // getters en setters
}
```

2.11.7 SporterMailing

```
package be.vdab.sportmailing.mailing;
// enkele imports
public interface SporterMailing {
  void stuurMailMetNieuwArtikel(Sporter sporter, ArtikelGemaakt gemaakt);
}
```

2.11.8 DefaultSporterMailing

```
package be.vdab.sportmailing.mailing;
// enkele imports
@Component
class DefaultSporterMailing implements SporterMailing {
  private final JavaMailSender sender;
  // constructor met parameter
  @Override
  public void stuurMailMetNieuwArtikel(Sporter sporter, ArtikelGemaakt gemaakt){
    try {
      var message = new SimpleMailMessage();
      message.setTo(sporter.getEmailAdres());
      message.setSubject("Nieuw artikel");
      message.setText("Er is een nieuw artikel:" + gemaakt.getNaam());
      sender.send(message);
    } catch (MailException ex) {
      throw new KanMailNietZendenException(ex);
    }
  }
}
```

2.11.9 ArtikelListener

```
package be.vdab.sportmailing.messaging;
// enkele imports
@Component
class ArtikelListener {
  private final SporterService service;
  private final SporterMailing mailing;
  private final Logger logger = LoggerFactory.getLogger(this.getClass());
  // constructor met parameters
  @RabbitListener(queues = "sportartikels")
  void verwerkBericht(ArtikelGemaakt gemaakt) {
    for (Sporter sporter : service.findAll()) {
      try {
        mailing.stuurMailMetNieuwArtikel(sporter, gemaakt);
      } catch (KanMailNietZendenException ex) {
        logger.error("Kan mail niet sturen", ex);
      }
    }
  }
}
```

2.11.10 SportmailingApplication

Extra code:

```
@Bean
Jackson2JsonMessageConverter converter() {
  return new Jackson2JsonMessageConverter();
}
```

2.12 Outbox

2.12.1 Database

Nieuwe table `artikelsgemaakt`, met zelfde structuur als in theorie.

2.12.2 ArtikelGemaakt

```
package be.vdab.sportwinkel.events;
// enkele imports
@Entity @Table(name="artikelsgemaakt")
public class ArtikelGemaakt {
  @Id private long id;
  private String naam;
  public ArtikelGemaakt(Artikel artikel) {
    this.id = artikel.getId();
    this.naam = artikel.getNaam();
  }
  // protected default constructor
  // getters
}
```

2.12.3 ArtikelGemaaktRepository

```
package be.vdab.sportwinkel.repositories;
// enkele imports
public interface ArtikelGemaaktRepository
  extends JpaRepository<ArtikelGemaakt, Long> {
}
```

2.12.4 DefaultArtikelService

```
package be.vdab.sportwinkel.services;
// enkele imports
@Service
@Transactional
class DefaultArtikelService implements ArtikelService {
  private final ArtikelRepository artikelRepository;
  private final ArtikelGemaaktRepository artikelGemaaktRepository;
  // constructor met parameters
  @Override
  public void create(Artikel artikel) {
    artikelRepository.save(artikel);
    artikelGemaaktRepository.save(new ArtikelGemaakt(artikel));
  }
}
```

2.12.5 SportwinkelApplication

```
@EnableScheduling
```

2.12.6 MessageSender

```
package be.vdab.sportwinkel.messaging;
// enkele imports
@Component
class MessageSender {
  private final ArtikelGemaaktRepository repository;
  private final AmqpTemplate template;
  // constructor met parameters
  @Scheduled(fixedDelay = 5_000)
  @Transactional
  void verstuurMessages() {
    var artikelsGemaakt = repository.findAll();
    for (ArtikelGemaakt gemaakt : artikelsGemaakt) {
      template.convertAndSend("sportartikels", null, gemaakt);
    }
    repository.deleteAll(artikelsGemaakt);
  }
}
```

3 COLOFON

Domeinexpertisemanager: Jean Smits

Moduleverantwoordelijke: Jean Smits

Medewerkers: Hans Desmet

Versie: 1/7/2020

Nummer dotatielijst: