Intec Brussel Examen 11 : Spring

Handtekening Cursist ***Ik heb het examenpapier met de feedbacks goed ontvangen. Ik accepteer de score.***	**Handtekening Instructeur**

Evaluatie rapport'	**Score**
Analyseren 35 EP + Java en SQL broncode 65EP	Totaal: 100EP (Examenpunten)
Score van de cursist	- - - - / 100

Samenvatting feedback

> ...
>
> ...
>
> ...

Inschrijvingsformulier

Cursist	
Voornaam	
Familienaam	
Klascode	Java Juni 21
Notities	

Algemeen info over het examen:

Het examensdatum	Geadviseerde deadline	De laatste deadline
31/01/2022 Start om 13:00	31/01/2022 23:59	31/01/2022 23:59
Contact info van de instructeur	yilmaz.mustafa@intecbrussel.be	+32 467 71 17 09

Vragen & Antwoorden voor de start

Vraag	Antwoord
Moet ik alle code zelf programmeren zonder een oplossing te zoeken via zoekmachines?	Nee, je mag alle resources gebruiken om te je vragen te antwoorden. Probeer echter niet te veel tijd te besteden aan het zoeken. Soms kost het meer tijd om tijd te besteden aan het vinden van een bestaand antwoord dan het zelf te schrijven.
Mag ik hulp of codestuks vragen aan de andere cursisten?	Nee, het is een volledig individueel examen.

Als ik de eerste deadline mis, mag ik dan doorgaan met het examen?	**Natuurlijk is er hieronder een scoretabel
die de berekening van scores per dag laat zien.**	
Geadviseerde datum	Je krijgt 100% van je score
Na 1 dag	Je krijgt 85% van je score
Na 2 dag	Je krijgt 70% van je score
Na 3 dag	Je krijgt 60% van je score
Na 3dag++	Kreeg 0 (null)

1. Inhoudsopgave

Dit examen leidt u door het proces van het maken van een Spring-toepassing die is verbonden met een MySQL-database.
Het maakt gebruik van Spring Data JPA om toegang te krijgen tot de database, maar dit is slechts een van de vele
mogelijke keuzes. In deze project, kiezen we JPA en REST om onze project te bouwen.

2. Wat je gaat bouwen

U maakt een MySQL-database, bouwt een Spring-toepassing (API) en verbindt deze met de nieuw gemaakte database.

3. Wat je nodig hebt

- [ ] MySQL versie 5.6 of beter. Als u GEEN database-server hebt geïnstalleerd, kan het handig zijn om de een
  package manager zoals chocolatey of scoop te installeren. Dan kan je wel de een van de gerelateerde commando's op
  terminal uitvoeren:
- [ ] IDE (Integrated Development Environment): Het is altijd ideal om een professionele-niveau development tools te
  gebruiken om sneller en veiliger te coderen. Meeste producten van IDE' zijn ingepakt met handig plugins zoals
  database management tools, code generators, syntax highlighting enz. In deze project is het geadviseerd om
  Intellij IDEA Ultieme versie te hebben.

```
choco install mysql
choco install intellijidea-ultimate

of

scoop bucket add JetBrains
scoop install IntelliJ-IDEA-Ultimate
scoop install mysql
```

4. Beginnen met Spring Initializr

> Kies een van deze 4 opties om je project te bouwen.

4.1. Spring Initializr

Deze optie genereert een properder sjabloon voor uw examen.

- [ ] U kunt dit auto-geïnitialiseerde project
  gebruiken en op Genereren klikken om een ZIP-bestand te downloaden. Dit project is geconfigureerd om te passen bij de vereiste jar-afhankelijkheden in dit
  examen.

4.2. Manueel

- [ ] Ga als volgt te werk om het project handmatig te initialiseren:
  - [ ] Navigeer naar https://start.spring.io. Deze service haalt alle afhankelijkheden op die u nodig hebt voor een toepassing en doet het grootste deel van de installatie voor u.
  - [ ] Kies Gradle of Maven en de taal die u wilt gebruiken. In deze handleiding wordt ervan uitgegaan dat u Java hebt gekozen.
  - [ ] Klik op **Afhankelijkheden** en selecteer **Spring Web, Spring Data JPA**en **MySQL Driver**.
  - [ ] Klik op **Genereren**.
  - [ ] Download het resulterende ZIP-bestand, een archief van een webtoepassing die is geconfigureerd met uw keuzes.

4.3. Intellij IDEA

-[ ] Als uw IDE de Spring Initializr-integratie heeft, kunt u dit proces voltooien vanuit uw IDE.

4.4. Git & Github-forken

U kunt het project ook vanuit Github forken en openen in uw IDE of andere editor.

- [ ] Klik hier link om het examen-project van Github te forken

Voeg jar-afhankelijkheden toe aan pom.xml

- [ ] Als parent voeg Spring boot starter parent naar je project toe:

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.3</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
```

- [ ] Pas de app met de volgende settings aan het POM:

```
<groupId>be.intec</groupId>
<artifactId>exam11</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>exam11</name>
<description>Spring Boot project</description>

<properties>
    <java.version>11</java.version>
</properties>
```

De app vereist de volgende dependencies:

- [ ] Lombok

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
  <scope>provided</scope>
</dependency>
```

- [ ] Spring Boot Data voor 'JPA & Hibernate'

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

```
<!--        springdoc-openapi v1.6.5 -->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>1.6.5</version>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
```

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
    <scope>provided</scope>
</dependency>
```

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

- [ ] Eigenschappen van pom.xml zijn hieronder gedeeld:

```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <!-- add property to enable or disable code coverage by default id false for
Jacoco -->
    <jacoco.skip>false</jacoco.skip>
</properties>
```

- [ ] Build-configuraties zijn hieronder gedeeld:

```
<build>
  <finalName>exam11</finalName>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

5. De database maken

Open een terminal (powershell in Microsoft Windows) en open een MySQL-client als een gebruiker die nieuwe gebruikers
kan maken. Als u een nieuwe database wilt maken, voert u de volgende opdrachten uit bij de prompt: mysql

```
create database exam11_db;
create user 'springuser'@'%' identified by 'ThePassword';
grant all on exam11_db.* to 'springuser'@'%';
```

6. Het aanmaken van het bestand 'application.yml'

Spring Boot geeft je standaardinstellingen op alle dingen. De standaarddatabase is bijvoorbeeld. Wanneer u een andere database wilt gebruiken, moet u daarom de verbindingskenmerken in het bestand definiëren.
Maak een bronbestand met de naam, zoals in de volgende lijst wordt weergegeven: `src/main/resources/application.yml`

Zie de documentatie bij Hibernate voor meer informatie.

- [ ] none: De standaardwaarde voor MySQL. Er wordt geen wijziging aangebracht in de databasestructuur.
- [ ] update: Hibernate wijzigt de database volgens de gegeven entiteit-structuren.
- [ ] create: Hiermee wordt de database elke keer gemaakt, maar wordt deze niet op sluiten gezet.
- [x] create-drop: Hiermee maakt u de database en zet deze neer wanneer deze wordt gesloten. (SessionFactory)

> Het optie 'create-drop' past het best voor het examen omdat de hele database contenten op elk app-uitvoeren gaat
> verwijdert worden.

Gebruik hiervoor de volgende configuratiebestand om database-verbindingen te configureren:

```
spring:
  jpa:
    hibernate.ddl-auto: create-drop
    show-sql: true

  datasource:
    url: jdbc:mysql://${MYSQL_HOST:localhost}:3306/exam11_db
    username: springuser
    password: ThePassword
    driver-class-name: com.mysql.jdbc.Driver

springdoc:
  swagger-ui:
    path: /api-page.html
  api-docs:
    path: /api-docs
```

7. Het @Entity-model maken

U moet het entiteit-modellen aanmaken, zoals de volgende vermelding (in) laat zien:
`src/main/java/be/intec/exam11/models/User.java` en
`src/main/java/be/intec/exam11/models/Message.java`

Met de **Hibernate** worden de entiteiten automatisch vertaald in een tabellen.

> JE MOET DE ENTITEITEN_RELATIES ZELF ONTWIKKELEN

8. Het repository layer te creëren

U moet de Repository-klassen (interface) maken die gebruikersrecords gaan beheren, zoals de volgende vermelding (in)
laat zien:
`src/main/java/be/intec/exam11/repositories/UserRepository.java`
en
`src/main/java/be/intec/exam11/repositories/MessageRepository.java`

Spring implementeert deze repository-interface automatisch in een BEAN met dezelfde naam (met een wijziging in de behuizing - het wordt genoemd userRepository en messageRepository).

9. Een controller maken

U moet een controller maken om HTTP-verzoeken (requests) voor uw toepassing af te handelen, zoals in de volgende
lijst (in) wordt weergegeven:
`src/main/java/be/intec/exam11/api/UserControllerImpl.java`
en
`src/main/java/be/intec/exam11/api/MessageControllerImpl.java`

- [ ] Implementeer de volgende interfaces om vereiste methoden sneller te ontwikkelen:
    - [ ] UserController interface

```
package be.intec.exam11.api;


import be.intec.exam11.models.User;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

public interface UserController {

    // @ResponseBody means the returned String is the response, not a view name
    // @RequestParam means it is a parameter from the GET or POST request
    @PostMapping ( path = "/params" )
    // Map ONLY POST Requests
    String addNewUserUsingRequestParams( @RequestParam String name,
@RequestParam String email );

    @PostMapping ( path = "/json" )
        // Map ONLY POST Requests
    String addNewUserUsingJSONData( @RequestBody User user );

    @GetMapping ( path = "/all" )
    List< User > getAllUsers();

    @GetMapping ( path = "/{user_id}" )
    Optional< User > getById( @PathVariable ( "user_id" ) Integer userId );

}
```

- [ ] MessageController interface

```
package be.intec.exam11.api;


import be.intec.exam11.models.Message;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
import java.util.Optional;
import java.util.Set;

public interface MessageController {

    @PostMapping ( path = "/params" )
    // Map ONLY POST Requests
    String addNewMessageUsingRequestParams( @RequestParam Integer senderId,
@RequestParam Set< Integer > recipientIdSet, @RequestParam String subject,
@RequestParam String content );

    @PostMapping ( path = "/json" )
        // Map ONLY POST Requests
    String addNewMessageUsingJSONData( @RequestBody Message message );

    @GetMapping ( path = "/all" )
    List< Message > getAllMessages();

    @GetMapping ( path = "/{message_id}" )
    Optional< Message > getMessageById( @PathVariable ( "message_id" ) Integer
messageId );

}
```

10. Een toepassingsklasse maken

Spring Initializr maakt een eenvoudige klasse voor de toepassing. In de volgende lijst wordt de klasse weergegeven die Initializr voor dit voorbeeld heeft gemaakt (in): `src/main/java/be/intec/exam11/AppStarter.java`

Als je het project zelf geïnstantieerd, Voeg de volgende code naar de-app starter klas toe:

```
package be.intec.exam11;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AppStarter {

    public static void main(String[] args) {
        SpringApplication.run( AppStarter.class, args);
    }
}
```

11. Test de applicatie met behulp van OpenAPI Swagger

Voor de integratie tussen spring-boot en swagger-ui voegt u de bibliotheek toe aan de lijst met projectafhankelijkheden (er is geen extra configuratie nodig)

```
<dependency>
   <groupId>org.springdoc</groupId>
   <artifactId>springdoc-openapi-ui</artifactId>
   <version>1.6.5</version>
</dependency>
```

Hiermee wordt swagger-ui automatisch geïmplementeerd in een spring-boot-toepassing:

- Documentatie zal beschikbaar zijn in HTML-formaat, met behulp van de officiële swagger-ui
- De Swagger UI-pagina is dan beschikbaar op en de OpenAPI-beschrijving is beschikbaar op de volgende url voor json-indeling: `http://server:port/context-path/swagger-ui.html``http://server:port/context-path/v3/api-docs`
  - server: de servernaam of het IP-adres
  - port: De serverpoort
  - context-path: het context-pad van de toepassing
- Documentatie kan ook beschikbaar zijn in yaml-formaat, op het volgende pad: /v3/api-docs.yaml

> `Voor een aangepast pad van de swagger-documentatie in HTML-indeling voegt u een aangepaste springdoc-eigenschap`
>
> toe in uw spring-boot configuratiebestand (application.yml):
> Voor het aangepaste pad van de OpenAPI-documentatie in Json-indeling voegt u een aangepaste eigenschap springdoc
> toe in uw configuratiebestand voor spring-boot (application.yml):

```
springdoc:
  swagger-ui:
    path: /api-page.html
  api-docs:
    path: /api-docs
```

In een bepaalde REST-controllers hebben we enkel belangrijkste API-methoden. Voel je vrij om meer methoden toe te voegen als dat nodig is.

12. Code-share

- [ ] Uploader je code van het examen naar je privémap van SharePoint de instructeur met je vroeger gedeeld heb.