**JPA Spring Start Project**

I dette dokument vil vi prøve vores første Spring projekt der bruger JPA.

Der findes masser af stof på nettet om JPA, men det er et godt tegn, hvis links indeholder enten ordet baeldung eller spring.io

<https://www.baeldung.com/the-persistence-layer-with-spring-data-jpa>

<https://spring.io/projects/spring-data-jpa>

Her starter vi med at oprette et nyt Spring projekt i IntelliJ.

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Kig nøje på ovenstående. Det er vigtigt at vælge Maven. Tryk Next.

Et billede, der indeholder tekst, software, skærmbillede, Webside

Automatisk genereret beskrivelse

Spring Boot version ændrer sig jævnligt. Lige nu er default version 3.3.2

Et billede, der indeholder tekst, skærmbillede, software, Webside

Automatisk genereret beskrivelse

Tilvælg Spring Web.

Et billede, der indeholder tekst, skærmbillede, display/skærm/fremvisning, software

Automatisk genereret beskrivelseTilvælg JPA, H2 Database og MySQL Driver.

Tryk Create.

Et billede, der indeholder tekst, skærmbillede, software, display/skærm/fremvisning

Automatisk genereret beskrivelse

Vi får følgende foldere og filer. Bemærk bl.a. at vi får en Maven pom.xml fil.

I pom.xml kan vi bl.a. se de 4 dependencies vi har valgt.

Et billede, der indeholder tekst, Font/skrifttype, linje/række, nummer/tal

Automatisk genereret beskrivelse

Jeg er ikke helt klar over hvad de gule hightligts betyder. IntelliJ prøver at advare os om usikre dependencies.

Højreklik på linjen com.example.jpastudent

Et billede, der indeholder tekst, Font/skrifttype, linje/række, skærmbillede

Automatisk genereret beskrivelse

Og opret en nu package model

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Højreklik på model og opret en ny klasse og kald den Student.

Et billede, der indeholder tekst, Font/skrifttype, linje/række, skærmbillede

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, software, skærmbillede, Computerikon

Automatisk genereret beskrivelse

Tilføj 4 instance variable som ses ovenfor.

Add getter og setter methods.

Et billede, der indeholder tekst, skærmbillede, nummer/tal, software

Automatisk genereret beskrivelse

Nu kommer det spændende, vi skal hook Student op til JPA.

Skriv @Entity over Student.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, Webside

Automatisk genereret beskrivelse

Med det samme burde vi kunne se Entity jakarta.persistence

Et billede, der indeholder tekst, Font/skrifttype, software, Webside

Automatisk genereret beskrivelse

Med det samme vi har tilføjet linjen @Entity, får vi brok fra IntelliJ, Student skal have en primary key.

Vores primary key er id. I JPA gøres det ved at annotere med @Id.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede

Automatisk genereret beskrivelse

Og yderligere en linje @GeneratedValue

Et billede, der indeholder tekst, skærmbillede, software, Font/skrifttype

Automatisk genereret beskrivelse

Vi vælger strategy = IDENTITY

Når vi hooker op til MySql om lidt, kan vi se det betyder vi får sat Autogenerate i MySql kollonne.

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

Til at starte med hooker vi op til H2 databasen.

Gå til application.properties filen under resources og tast spring.h2.console.enabled=true

Et billede, der indeholder tekst, software, Computerikon, Webside

Automatisk genereret beskrivelse

Lad os prøve at køre applicationen.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Hvis vi har tastet rigtigt, får vi noget i den her retning:

Et billede, der indeholder tekst, skærmbillede, nummer/tal, Font/skrifttype

Automatisk genereret beskrivelse

Læg især mærke til linjen:

url=jdbc:h2:mem:564bb0af-be5e-44b0-8d6e-3e2808da6e13 user=SA

Lige efter linjen med

HikariPool-1 - Starting...

Skriv følgende url i en browser:

http://localhost:8080/h2-console/login.jsp

Følgende skærmbillede skal dukke op:

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

I JDBC URL linjen, kopier url ind fra jeres Console, den url linje der er nævnt ovenfor.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, elektronik, skærmbillede, display/skærm/fremvisning

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Vi kan nu tilgå vores H2 database i en browser. H2 databasen er det man kalder for en In Memory database.

Så snart vi stopper vores program, så forsvinder H2 databasen også.

Man kan læse mere om H2 databasen her:

https://www.baeldung.com/spring-boot-h2-database

Nu skal vi prøve at hook op til vores MySql database.

Tag en backup af vores H2 application.properties, vi kan kalde den appproph2.txt

Et billede, der indeholder tekst, skærmbillede, software, Font/skrifttype

Automatisk genereret beskrivelse

I application.properties skal vi nu hook op til vores MySql database.

Vi skal skrive følgende linjer:

spring.datasource.url=jdbc:mysql://127.0.0.1:3306/student01  
spring.datasource.username=jens  
spring.datasource.password=x  
  
spring.jpa.hibernate.ddl-auto=update

Den første linje med url, skriver vi ipadresse på MySql server, efterfulgt af port 3306 så en forwardslash og navn på vores database.

Start MySql workbench.

Et billede, der indeholder tekst, software, Computerikon, Webside

Automatisk genereret beskrivelse

I må meget gerne oprette en ny bruger til lejligheden, med et simpelt password.

Naviger til Administration fanebladet:

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Klik på Users and Privileges

Et billede, der indeholder tekst, skærmbillede, software, Font/skrifttype

Automatisk genereret beskrivelse

Klik på Add Account.

Et billede, der indeholder tekst, skærmbillede, Rektangel, Font/skrifttype

Automatisk genereret beskrivelse

Opret en bruger som er nem at huske med et simpelt password.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, nummer/tal

Automatisk genereret beskrivelse

Klik på fanebladet Administrative Roles

Et billede, der indeholder tekst, elektronik, skærmbillede, display/skærm/fremvisning

Automatisk genereret beskrivelse

Og tilvælg alle roller.

Tryk på Apply knappen.

Et billede, der indeholder tekst, skærmbillede, Rektangel, Font/skrifttype

Automatisk genereret beskrivelse

Vi kan tilføje vores nye bruger til forsiden af MySql workbench.

Ved at trykke på + knappen.

Et billede, der indeholder Font/skrifttype, tekst, logo, Grafik

Automatisk genereret beskrivelse

Det vigtigste lige nu er at vi skal bruge vores nye bruger i vores url, med userid og password.

Vi skal lige oprette en ny database først.

Tryk på + database icon. Nr 4 icon fra venstre.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, nummer/tal

Automatisk genereret beskrivelse

Kald den student og tryk Apply. Jeg skriver alt sql med småt.

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Nu er vi klar til at skrive vores application.properties færdig.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, software

Automatisk genereret beskrivelse

Så altså min application.properties ser præcis sådan her ud:

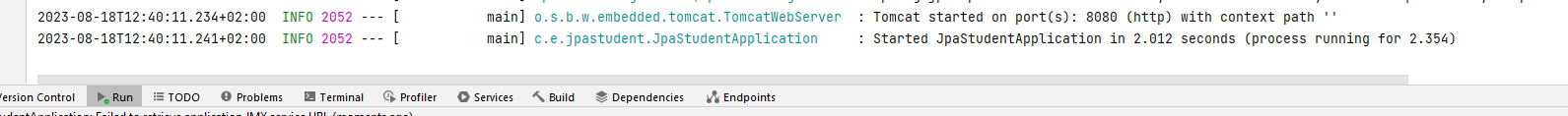
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/student  
spring.datasource.username=jens  
spring.datasource.password=x  
spring.jpa.hibernate.ddl-auto=create-drop

Bemærk at jeg har skrevet create-drop udfor auto=

Man kan skrive create-drop eller update. create-drop betyder at databasen bliver slettet og man starter forfra når programmet køres. Så vi ændrer til update når vi har nogle data i vores database, som vi ikke vil have slettet.

Lad os køre programmet og se hvad der sker.

Hvis I får fejl, så check nøje at databasen er oprettet og I bruger rigtigt navn, samme med user og password.



Der må ikke stå finished nederst i jeres Console.

Ovenstående er korrekt. Nedenstående har jeg skrevet navn på en database der ikke findes.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

Når I har fået programmet til at køre, så til MySql Workbench.

Tryk det lille reload icon.

Et billede, der indeholder tekst, Font/skrifttype, linje/række, skærmbillede

Automatisk genereret beskrivelse

Og nu skulle I kunne se jeres database med en tabel der hedder student.

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Højreklik student tabellen og vælg Alter Table..

Et billede, der indeholder tekst, skærmbillede, nummer/tal, Font/skrifttype

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, software, nummer/tal, Computerikon

Automatisk genereret beskrivelse

Her kan I se at AI er sat, det betyder AutoIncrement.

Så id bliver automatisk talt op af MySql.

Lad os putte nogen data i databasen.

Vi opretter en ny package og kalder den config.

Under config opret en ny klasse og kald den InitData.

Et billede, der indeholder tekst, skærmbillede, software, Font/skrifttype

Automatisk genereret beskrivelse

InitData skal implementere interfacet CommandLineRunner.

Prøv tryk alt+insert og vælg implement methods

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, software

Automatisk genereret beskrivelse

Der er kun 1 metode i CommandLineRunner interface, nemlig run.

Her kan vi oprette nogle Student og gemme dem i vores nye database.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, nummer/tal

Automatisk genereret beskrivelse

Nu har vi en Student, hvordan får vi den i vores database med JPA ?

Designerne af JPA har besluttet at vi gør det via et interface, kaldet JpaRepository.

Opret en ny package og kald den repositories.

Opret et nyt interface og kald det StudentRepository.

Et billede, der indeholder tekst, software, skærmbillede, Font/skrifttype

Automatisk genereret beskrivelse

public interface StudentRepository extends JpaRepository<Student, Integer> {  
}

Vi koder et nyt interface StudentRepository som nedarver fra JpaRepository, som tager 2 typer. Navnet på vores Entity klasse og typen på vores primary key. Så altså Student og Integer.

Det er alt hvad vi behøver i første omgang, nu kan vi gemme studenter i databasen.

Følgende kode vil nu gemme en række i vores student tabel.

Et billede, der indeholder tekst, skærmbillede, software, Webside

Automatisk genereret beskrivelse

Vi bruger Autorwire til at injekt en implementation af vores StudentRepository ind i CommandLineRunner klassen. StudentRepository har 1 save metode vi kan bruge.

Selve InitData klassen skal vi annotere med @Component, for at run metoden bliver udført.

Prøv at gemme en studerende mere i databasen.

Husk den her linje i application.properties

spring.jpa.hibernate.ddl-auto=create-drop

Bestemmer om databasen skal cleares(droppes) hver gang, hvis man skriver update i stedet, så overlever data imellem kørsler.

Hvad sker der hvis vi skriver følgende kode ?

Student s1 = new Student();  
s1.setName("Bruce");  
s1.setBornDate(LocalDate.*of*(2010, 11, 12));  
s1.setBornTime(LocalTime.*of*(10,11,12));  
studentRepository.save(s1);  
studentRepository.save(s1);  
studentRepository.save(s1);

Prøv at forklar.

Det hjælper måske at debug:

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, software

Automatisk genereret beskrivelse

Prøv at tilføje en linje efter linje 26, der ændrer navnet på Bruce til Tim, og se hvad der sker i databasen. Hvad er det for en type sql der bliver udført ?

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Prøv indsæt følgende linje i jeres application.properties

spring.jpa.show-sql=true

Og kør igen og se i jeres Console, med en linje tilføjet i med Tim efter linje 26.

Prøv at få indsat 2 rækker i student tabellen.

**I dette afsnit vil vi vise vores studerende i en browser.**

Det gør vi ved at kode vores første RestApi kald, også kaldet en webservice.

Og her er det OK at spørge ChatGpt, what is a Rest Api.

Opret en ny package og kald den controller

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Opret en ny Klasse og kald den StudentRestController

Et billede, der indeholder tekst, software, Computerikon, skærmbillede

Automatisk genereret beskrivelse

Vi starter med at annotere vores klasse som en RestController.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, design

Automatisk genereret beskrivelse

Vi koder et endpoint der kan kaldes fra en browser og vise alle vores students.

@GetMapping("/students")  
public List<Student> students() {  
   
}

Nu skal vi læse alle vores students fra databasen. Her skal vi bruge vores StudentRepository.

På samme måde som i InitData klassen, vi autowire det ind.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Og vi kan se at der findes en findAll() metode i vores repository som returnere en liste af students.

http://localhost:8080/students

Kan se sådan her ud

Et billede, der indeholder tekst, skærmbillede, software, Webside

Automatisk genereret beskrivelse

Fra browseren kan vi umiddelbart kun kalde Get. Og det er fristende nu at kode en Get metode der indsætter en ny Student i databasen. Kort sagt hvis vi har indtastet dette endpoint i vores browser, så får vi indsat en Student i vores database, hver gang vi trykker refresh i browseren.

@GetMapping("/addstudent")  
public List<Student> addstudent() {  
 Student std = new Student();  
 std.setBornDate(LocalDate.*now*());

lst = studentRepository.findAll();  
 return lst;

I kan i øvrigt kalde jeres Rest Api indefra IntelliJ. Prøv højreklik på controller package.

Et billede, der indeholder tekst, skærmbillede, software, nummer/tal

Automatisk genereret beskrivelse

Vælg New, og så Http Request fra popup menu.

Find på et eller andet navn.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Der oprettes en ny fil i controller package, reststudent.http

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Vi kan nu afprøve vores webservice indefra IntelliJ.

Tryk play knappen til venstre for vores GET linje.

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Vi kan godt eksperimentere lidt mere med JPA i vores RestController, f.eks. kan vi bede JPA om at finde en student med et bestemt navn.

For at kalde vores repository med name som parameter, skal vi lige oprette sådan en metode først.

public interface StudentRepository extends JpaRepository<Student, Integer> {  
 List<Student> findAllByName(String name);  
  
}

@GetMapping("students/{name}")  
public List<Student> getallstudentsbyname(@PathVariable String name) {  
 return studentRepository.findAllByName(name);  
}

Vi kan afprøve det fra en browser eller fra vores nye reststudent.http fil.

Tryk på + knap for at få et nyt web request tilføjet til reststudent.http

Et billede, der indeholder tekst, Font/skrifttype, linje/række, nummer/tal

Automatisk genereret beskrivelse

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Der er mange måder at eksperimenter med JPA via vores RestController, find alle ByBornDate, eller slet en Student med et bestemt Id.

Men det er ikke rigtig meningen at man bruger http verbet Get, til at udføre andet end netop Get.

Vi kommer til at arbejde med Create, Update og Delete http verber senere når vi har installeret Postman.

Så en anden måde at eksperimentere med JPA, er ved at skrive nogle Unit Testmetoder.

Så det sidste afsnit i dette dokument omhandler Unit Test af Jpa metoder.

I en hvilken som helst klasse, f.eks. StudentRepository, tryk Alt+Insert

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, software

Automatisk genereret beskrivelse

Og vælg Test… fra popumenu.

Et billede, der indeholder tekst, skærmbillede, display/skærm/fremvisning, software

Automatisk genereret beskrivelse

Bare tryk OK, uden at vælge noget

Vi får nu genereret en test klasse.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, software

Automatisk genereret beskrivelse

Det kan godt være lidt svært at se hvor den klasse ligger i vores projekt.

Tryk Alt+F1.

Et billede, der indeholder tekst, skærmbillede, software, Webside

Automatisk genereret beskrivelse

Tryk Enter.

Et billede, der indeholder tekst, software, Font/skrifttype, Webside

Automatisk genereret beskrivelse

Vi kan se at klassen ligger under test/java/

Her vil jeg lige nævne en anden vigtig genvejstast i IntelliJ. Tryk Shift tasten 2 gange.

Så kommer et søgevindue frem, hvor man kan søge på alt i sit projekt. Klasser, metoder osv.

Et billede, der indeholder tekst, software, Webside, Website

Automatisk genereret beskrivelse

Vi fortsætter med vores testklasse. Vi skal annotere at det er en test klasse. Det kan gøres med annotationen @SpringBootTest.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Vi vil skrive en testmetode der tester der er en Tim i vores database.

Vi skal igen Autowire vores StudentRepository ind.

Og så skriver vi en simpel testmetode. Og testmetoder skal annoteres med @Test.

Vores første testmetode ser sådan her ud.

Et billede, der indeholder tekst, skærmbillede, software, Computerikon

Automatisk genereret beskrivelse

Herfra er der uanede muligheder for at eksperimentere med vores Student og JPA.

Der er mange måder at køre vores testmetode på.

Tryk f.eks. på play knappen lige til venstre for metoden.

Et billede, der indeholder tekst, Font/skrifttype, skærmbillede, linje/række

Automatisk genereret beskrivelse

Testmetoderne dukker også op i toppen af vores toolbar.

Et billede, der indeholder tekst, skærmbillede, Font/skrifttype, linje/række

Automatisk genereret beskrivelse

Vi kan prøve at skrive en ny testmetode.

Et billede, der indeholder tekst, software, Computerikon, Webside

Automatisk genereret beskrivelse

Man kan køre testmetoder 1 af gangen eller flere af gangen.

F.eks. kan man trykke på play icon til venstre for klassedeklaration.

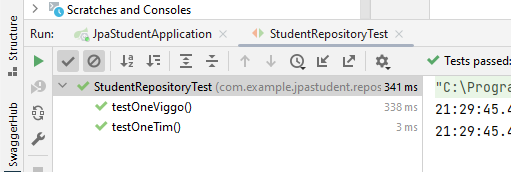
Et billede, der indeholder tekst, skærmbillede, software, Font/skrifttype

Automatisk genereret beskrivelse

Og vælge Run StudentRepositoryTest.

Så bliver alle testmetoder i klassen kørt.

Og resultatet kan ses nederst til venstre.



Prøv at ændre create-drop til update i application.properties. Og kør diverse tests igen.