Slutrapport Java

AcivityApp

2017-12-05

YHSIPI17

Tommy Eriksson

Innehållsförteckning

[Sammanfattning 2](#_Toc500679514)

[Inledning 3](#_Toc500679515)

[Genomförande 3](#_Toc500679516)

[Konfiguration 3](#_Toc500679517)

[pom.xml 3](#_Toc500679518)

[Application.properties 3](#_Toc500679519)

[Main 4](#_Toc500679520)

[Models 4](#_Toc500679521)

[Controllers 4](#_Toc500679522)

[Services 5](#_Toc500679523)

[Repository 5](#_Toc500679524)

[Resources 6](#_Toc500679525)

[Static 6](#_Toc500679526)

[Templates 6](#_Toc500679527)

[Avgränsning 7](#_Toc500679528)

[Slutsats 8](#_Toc500679529)

# Sammanfattning

Rapporten beskriver hur en Excel aktivitetslista kan göras som en webbsida med hjälp Java plattformen.

En användare navigerar till en webbsida. På webbsidan visas en lista med olika aktiviteter. Aktiviteterna innehåller information om vem som är ansvarig, skapat datum, önskat slutdatum, verkligt slutdatum, status och även en beskrivande text. Användaren kan skapa nya, redigera och ta bort aktiviteter. Användaren kan också navigera till en enskild aktivitet och lägga till kommentarer, det kan användas som en slags uppföljning av en enskild aktivitet.

Teknologier som används är Spring Boot, JPA, Postgresql, Thymeleaf, Spring-MVC, hibernate-validator

# Inledning

Som ett avslutande arbete i kursen ”Systemutveckling med Java” ska jag utveckla ett mer avancerat projekt i valfri IDE. Applikationen kan vara en miniräknare, ett spel eller ett verktyg av något slag som har en databaskoppling eller fungerar över ett nätverk. Programmet använder klasser och metoder och har i grunden en objektorienterad struktur och uppfyller ett syfte.

Mitt projekt är en Spring-MVC webbapplikation som implementerar funktionalitet hämtat från en Excelfil som hanterar aktiviteter. Syftet med projektet är att jag ska lära mig mer om hur spring-MVC implementeras i en Java applikation och ha en demo på hur actionlistan kan se ut i webbutförande. Tanken är inte att ha ett komplett program utan mer en demo med viss funktionalitet att bygga vidare på.

# Genomförande

## Konfiguration

Som utgångspunkt skapas ett ”Maven med Java och Spring Boot 1.5.8” projekt på <https://start.spring.io/>. Där läggs de dependencies till som behövs för det här projektet:

* **Web** - Full-stack web development with Tomcat and Spring MVC
* **JPA** - Java Persistence API including spring-data-jpa, spring-orm and Hibernate
* **PostgreSQL** - PostgreSQL jdbc driver
* **Thymeleaf** - Thymeleaf templating engine, including integration with Spring

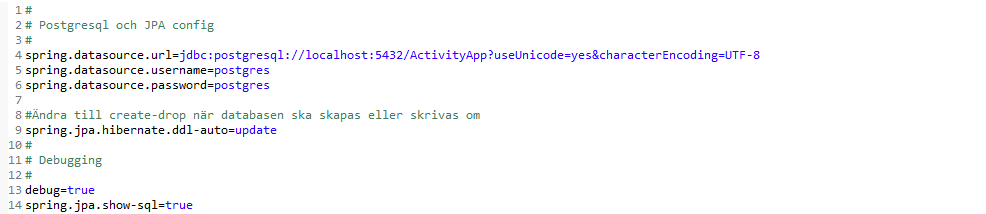
När sedan knappen “Generate Project” väljs så skapas en zip-fil innehållande projektet som laddas ned.

### pom.xml

Eftersom det här är ett Maven projekt så hanteras alla dependencies i den här filen. Behöver vi lägga till något senare i projektet så lägger vi till det i den här filen

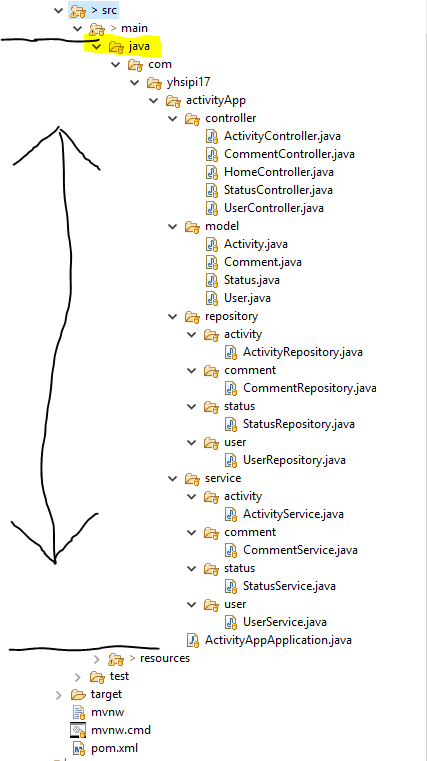
### Application.properties

I den här filen görs inställningar för applikationen. Här anges bland annat var databasen finns samt anändarnamn och lösenord. Vi anger också om databasen ska uppdateras eller skrivas över.



## Main

Backend

Main metoden finns ActivityAppApplication och placeras i root-package och annoteras med @SpringBootApplication, ett enklare sätt att inkludera @Configuration, @EnableConfiguration och @ComponentScan. I och med att Main läggs i root-package så kan @ComponentScan användas utan parametrar.

## Models

Det här är Java klasser som representerar tabeller i databasen. Det är här som tabeller och relationer sätts och även hur klassmedlemmarna ska valideras.

###### Activity.java

Den här klassen representerar själva aktiviteten, den som hela sidan är uppbyggd kring. Den har ett antal annoteringar som talar om hur den ska mappas mot databasen samt hur den ska validera klassmedlemmarna och vilken text som ska generaras om det är fel.

###### User.java

Representerar användarna av applikationen. Även här finns annotering rörande mappning och validering.

###### Comment.java

Varje aktivitet kan ha en lista med kommentarer. Dessa kan ses som en slags uppföljning av aktiviteten. Valideringen är inte implementerad i denna klass. Det bör göras i framtiden.

###### Status.java

Avser aktivitetens status.

## Controllers

De här klasserna tar hand om de http anrop de är mappade för och returnerar en webbsida. De använder sig av en service för att hantera den data som behövs och en view för den html som ska visas.

###### HomeController

Det enda den här controllern gör, är att styra vidare till ActivityController. Detta för att jag vill att användare ska komma rätt om den navigerar till ”/hem” eller ”/”.

###### ActivityController

Den här controllern svarar på anrop som är mappade ”/activity/\* ”.

###### UserController

Den här controllern svarar på anrop som är mappade ”/users/\* ”.

###### CommentController

Den här controllern svarar på anrop som är mappade ”/comment/\* ”.

###### StatusController

Den här controllern svarar på anrop som är mappade ”/status/\* ”.

## Services

Här finns implementationerna av repository. I de här klasserna anropar man de CRUD operationer som ska utföras och returnera den datan.

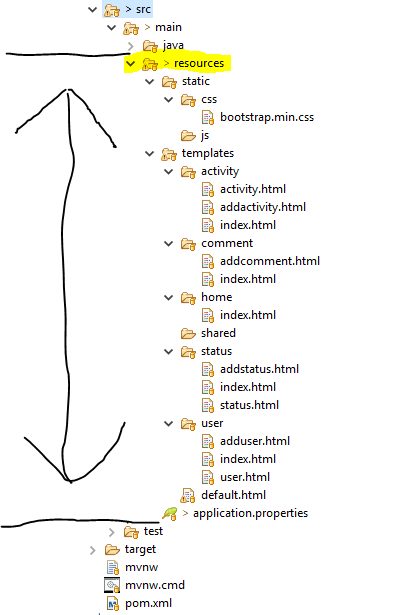
## Repository

De här klasserna är de som pratar med databasen och det görs genom spring-data-JPA. Genom att skapa ett repository och det sedan ärver JpaRepository så får man ett antal standard metoder man kan använda sig av. Man kan även definiera egna metoder genom att använda sig av nyckelord som findBy<field>. ActivityRepository har en sådan definition som returnerar alla aktiviteter sorterade efter datum.

public List<Activity> findAllByOrderByRequestDateAsc();

## Resources

Frontend

Här finns allt som är för frontend.

### Static

Här ligger alla statiska filer som t.ex. CSS, JavaScript mm

### Templates

Här finns alla htmlfiler. Jag använder Thymeleaf som är en template engine. Varje controller har en egen mapp med filer som används av metoderna i klassen.

# Avgränsning

Till varje aktivitet och kommentar är det tänkt att det ska finnas en skapare, det är inte implementerat då jag anser att det behövs ett identifieringssystem av något slag. Det kan vara så att för att utföra ändringar och lägga till nya aktiviteter, kommentarer, statusar så ska man vara inloggad.

När listan växer så kan någon sorts sidvisning och sortering behöva implementeras. Jag har valt att inte göra det nu då det inte tillför något till syftet för projektet.

Någon markering på att en aktivitet är avslutad eller inte finns inte. För nu räcker det att man ser att den är avslutad, i framtiden så kanske den ska markeras avslutad och tas bort från listan. Man kanske även ska kunna återstart avrapporterade aktiviteter.

# Slutsats

Jag har nu en spring-MVC webbapplikation där användarna genom ett webbinterface kan skapa, visa och hantera aktiviteter. Den kan nu användas som en demo och en utgångspunkt för att skapa en actionlista på webben.

Jag har även lärt mig om spring-MVC och hur det implementeras i Java. Vi skulle redogöra djupare för en del av spring framework. Jag tänkte redogöra för spring-MVC men istället för att skriva en massa text tänkte jag låta en bild och hur jag implementerat MVC i mittprojekt stå för redogörelsen.

Jag tänkte visa med en bild hur t.ex. en get request till <http://localhost:8080/activityapp> hanteras av applikationen.

Html

Html

ViewTemplate

Model

Model

Activitys

Activitys

Activitys

Db

Activity Repository

Activity Service

Activity Controller

Activityapp/

FrontController