netcompany

TCCS

VERSION
1.3

Getting started quickly with Spring Boot

Oslo, august 27, 2020

Agenda

- Motivation
- What is Spring?
- What is Spring Boot?
- Workshop

Motivation

- You have an idea you want to show
- Or a customer wants a PoC
- You need something quickly
  - Spring Boot is one possibility

Spring Theory

- Not string theory!

What is Spring?

- Framework for modern java-based applications
- Spring takes care of the plumbing
    - So you can focus on business logic

spring

by Pivotal™

What is Spring?

Core technologies: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.

Testing: mock objects, TestContext framework, Spring MVC Test, WebTestClient.

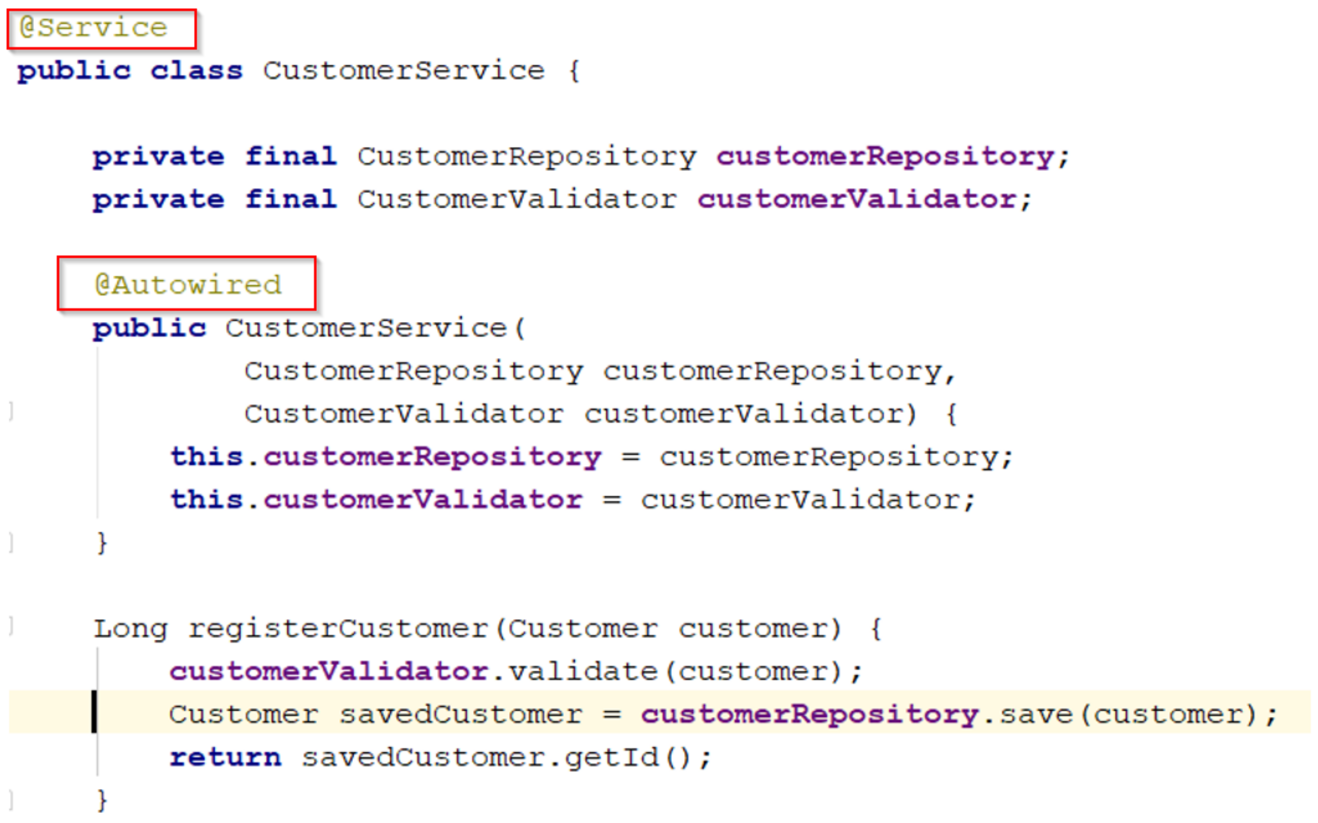
Data Access: transactions, DAO support, JDBC, ORM, Marshalling XML.

Spring MVC and Spring WebFlux web frameworks.

Integration: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.

Languages: Kotlin, Groovy, dynamic languages.

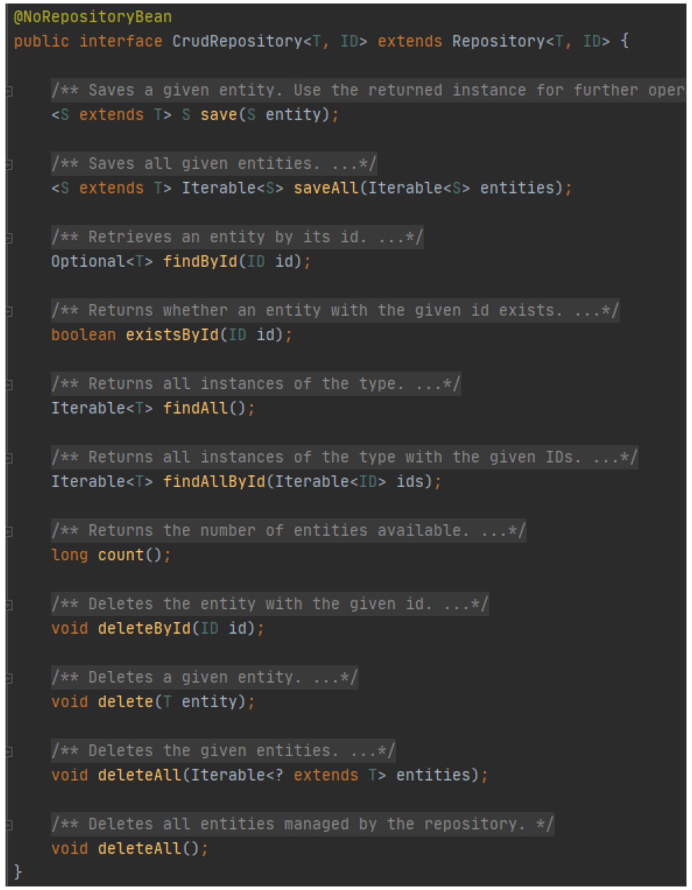
Spring DI



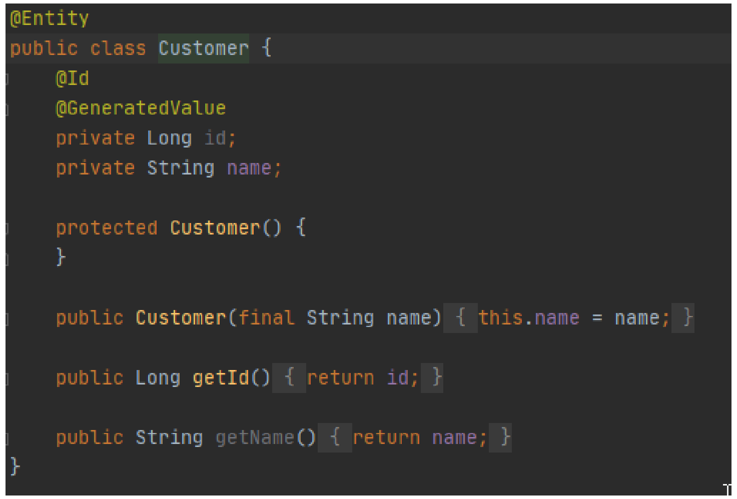
```
@Service
public class CustomerService {

    private final CustomerRepository customerRepository;
    private final CustomerValidator customerValidator;

    @Autowired
    public CustomerService(
            CustomerRepository customerRepository,
            CustomerValidator customerValidator) {
        this.customerRepository = customerRepository;
        this.customerValidator = customerValidator;
    }

    Long registerCustomer(Customer customer) {
        customerValidator.validate(customer);
        Customer savedCustomer = customerRepository.save(customer);
        return savedCustomer.getId();
    }
```

Spring – A bit more details

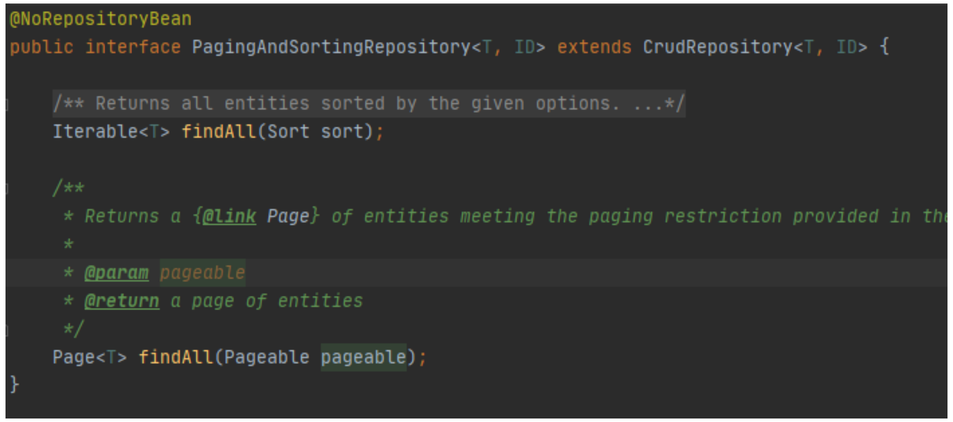
- JPA



```java
@NoRepositoryBean
public interface CrudRepository<T, ID> extends Repository<T, ID> {

    /** Saves a given entity. Use the returned instance for further oper
    <S extends T> S save(S entity);

    /** Saves all given entities. ...*/
    <S extends T> Iterable<S> saveAll(Iterable<S> entities);

    /** Retrieves an entity by its id. ...*/
    Optional<T> findById(ID id);

    /** Returns whether an entity with the given id exists. ...*/
    boolean existsById(ID id);

    /** Returns all instances of the type. ...*/
    Iterable<T> findAll();

    /** Returns all instances of the type with the given IDs. ...*/
    Iterable<T> findAllById(Iterable<ID> ids);

    /** Returns the number of entities available. ...*/
    long count();

    /** Deletes the entity with the given id. ...*/
    void deleteById(ID id);

    /** Deletes a given entity. ...*/
    void delete(T entity);

    /** Deletes the given entities. ...*/
    void deleteAll(Iterable<? extends T> entities);

    /** Deletes all entities managed by the repository. */
    void deleteAll();
}
```



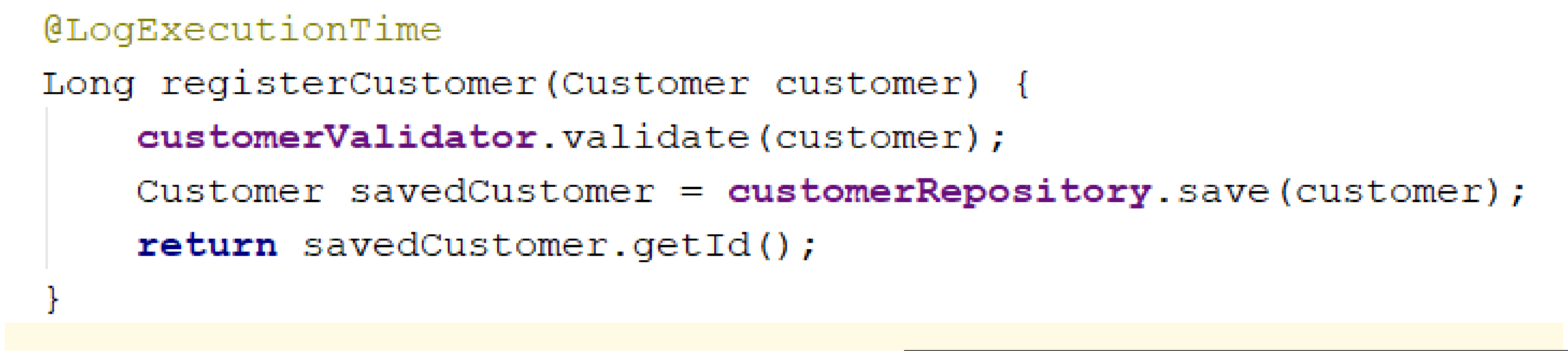
```java
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String name;

    protected Customer() {
    }

    public Customer(final String name) { this.name = name; }

    public Long getId() { return id; }

    public String getName() { return name; }
}
```

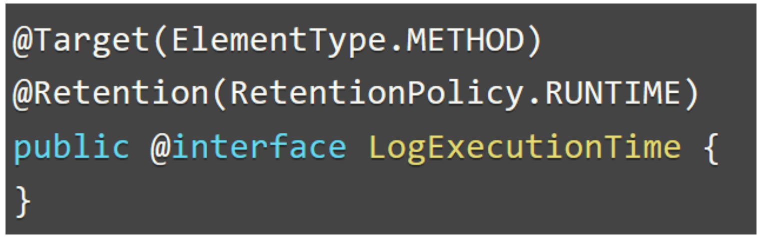


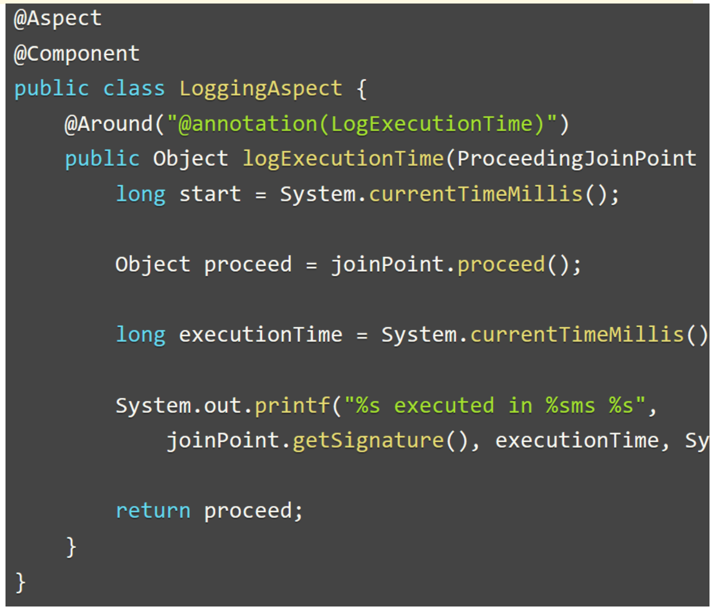
```java
@NoRepositoryBean
public interface PagingAndSortingRepository<T, ID> extends CrudRepository<T, ID> {

    /** Returns all entities sorted by the given options. ...*/
    Iterable<T> findAll(Sort sort);

    /**
     * Returns a {@link Page} of entities meeting the paging restriction provided in the
     *
     * @param pageable
     * @return a page of entities
     */
    Page<T> findAll(Pageable pageable);
}
```

netcompany

Spring AoP

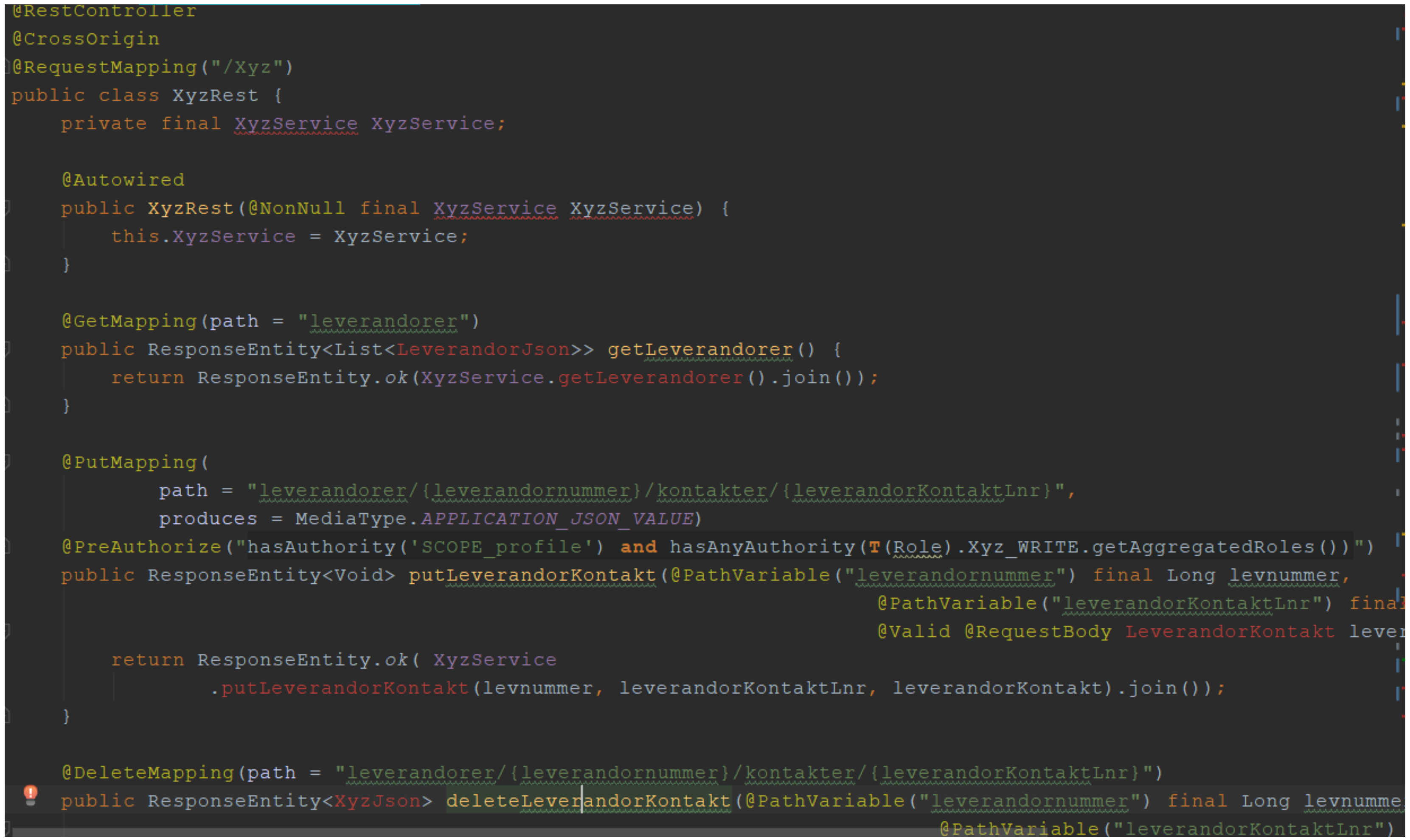

```java
@LogExecutionTime
Long registerCustomer(Customer customer) {
    customerValidator.validate(customer);
    Customer savedCustomer = customerRepository.save(customer);
    return savedCustomer.getId();
}
```


```java
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface LogExecutionTime {
}
```


```java
@Aspect
@Component
public class LoggingAspect {
    @Around("@annotation(LogExecutionTime)")
    public Object logExecutionTime(ProceedingJoinPoint
        long start = System.currentTimeMillis();

        Object proceed = joinPoint.proceed();

        long executionTime = System.currentTimeMillis()

        System.out.printf("%s executed in %sms %s",
            joinPoint.getSignature(), executionTime, Sy

        return proceed;
    }
}
```

netcompany

Spring – A bit more details



```
@RestController
@CrossOrigin
@RequestMapping("/Xyz")
public class XyzRest {
    private final XyzService XyzService;

    @Autowired
    public XyzRest(@NonNull final XyzService XyzService) {
        this.XyzService = XyzService;
    }

    @GetMapping(path = "leverandorer")
    public ResponseEntity<List<LeverandorJson>> getLeverandorer() {
        return ResponseEntity.ok(XyzService.getLeverandorer().join());
    }

    @PutMapping(
            path = "leverandorer/{leverandornummer}/kontakter/{leverandorKontaktLnr}",
            produces = MediaType.APPLICATION_JSON_VALUE)
    @PreAuthorize("hasAuthority('SCOPE_profile') and hasAnyAuthority(T(Role).Xyz_WRITE.getAggregatedRoles())")
    public ResponseEntity<Void> putLeverandorKontakt(@PathVariable("leverandornummer") final Long levnummer,
                                                      @PathVariable("leverandorKontaktLnr") fina
                                                      @Valid @RequestBody LeverandorKontakt lever

        return ResponseEntity.ok( XyzService
                .putLeverandorKontakt(levnummer, leverandorKontaktLnr, leverandorKontakt).join());
    }

    @DeleteMapping(path = "leverandorer/{leverandornummer}/kontakter/{leverandorKontaktLnr}")
    public ResponseEntity<XyzJson> deleteLeverandorKontakt(@PathVariable("leverandornummer") final Long levnumme
                                                            @PathVariable("leverandorKontaktLnr")
```

netcompany

Why use SpringBoot?

Package Application

Choose & Download Webserver

Configure Webserver

Deploy Application & Start Webserver

Package & Run

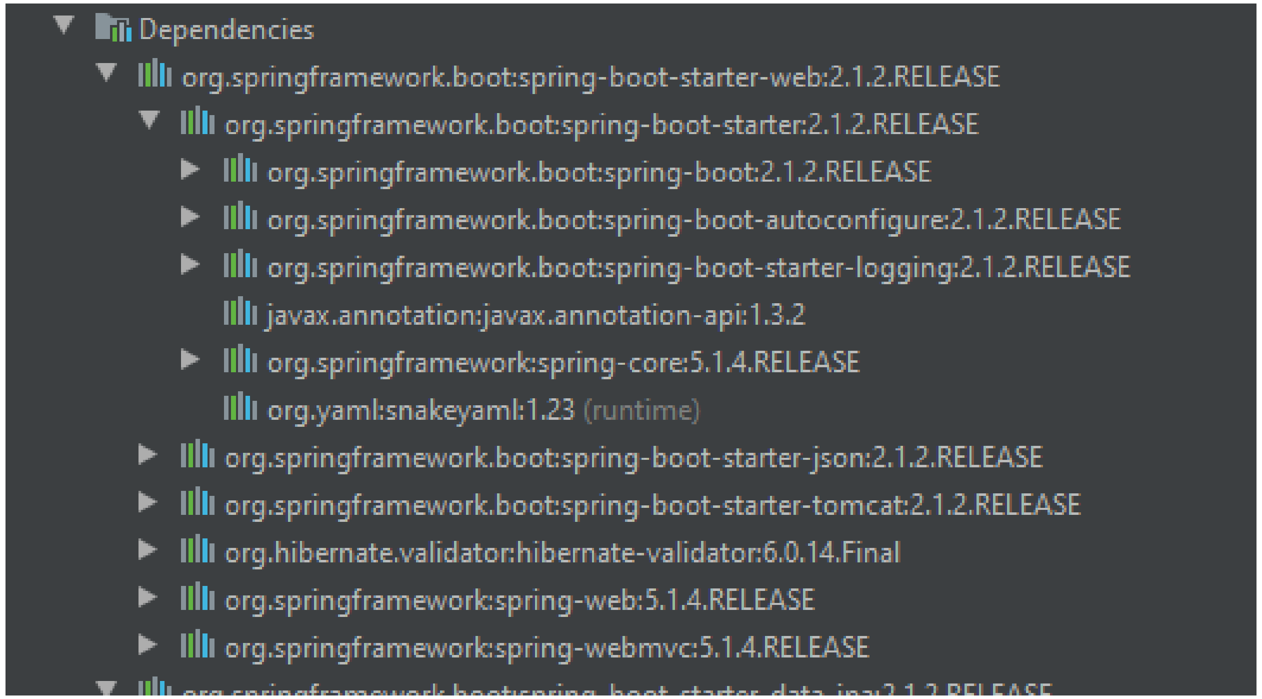
Spring Boot

- Opinionated view of the Spring plattform
- Easy
- Quick
- Configurable if/when you need it
- Standalone

netcompany

Spring Boot

- Starters
  - Opinionated, fixes our dependency tree
  - Includes «all you need»

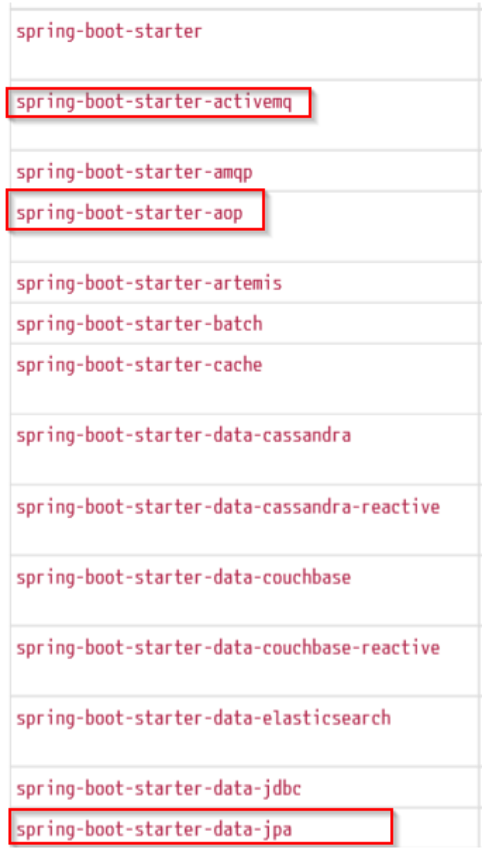


```
▼ Dependencies
    ▼ org.springframework.boot:spring-boot-starter-web:2.1.2.RELEASE
        ▼ org.springframework.boot:spring-boot-starter:2.1.2.RELEASE
            ▶ org.springframework.boot:spring-boot:2.1.2.RELEASE
            ▶ org.springframework.boot:spring-boot-autoconfigure:2.1.2.RELEASE
            ▶ org.springframework.boot:spring-boot-starter-logging:2.1.2.RELEASE
              javax.annotation:javax.annotation-api:1.3.2
            ▶ org.springframework:spring-core:5.1.4.RELEASE
              org.yaml:snakeyaml:1.23 (runtime)
        ▶ org.springframework.boot:spring-boot-starter-json:2.1.2.RELEASE
        ▶ org.springframework.boot:spring-boot-starter-tomcat:2.1.2.RELEASE
        ▶ org.hibernate.validator:hibernate-validator:6.0.14.Final
        ▶ org.springframework:spring-web:5.1.4.RELEASE
        ▶ org.springframework:spring-webmvc:5.1.4.RELEASE
    ▼ org.springframework.boot:spring-boot-starter-data-jpa:2.1.2.RELEASE
```

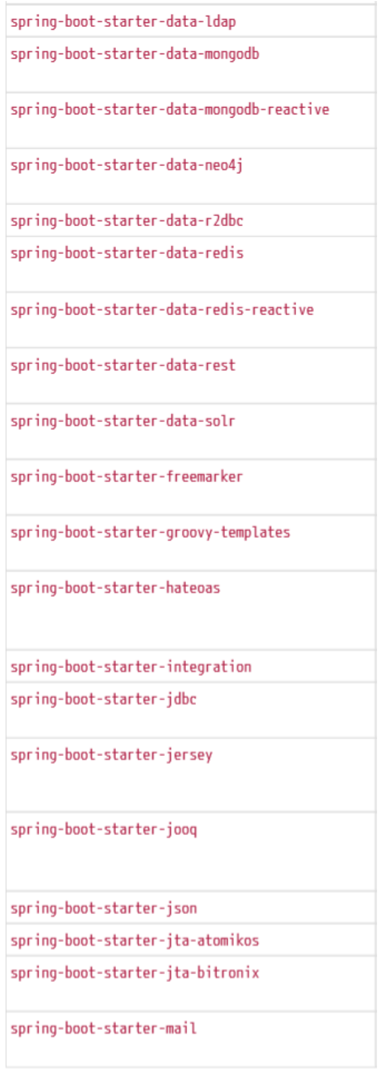
netcompany

Spring Boot

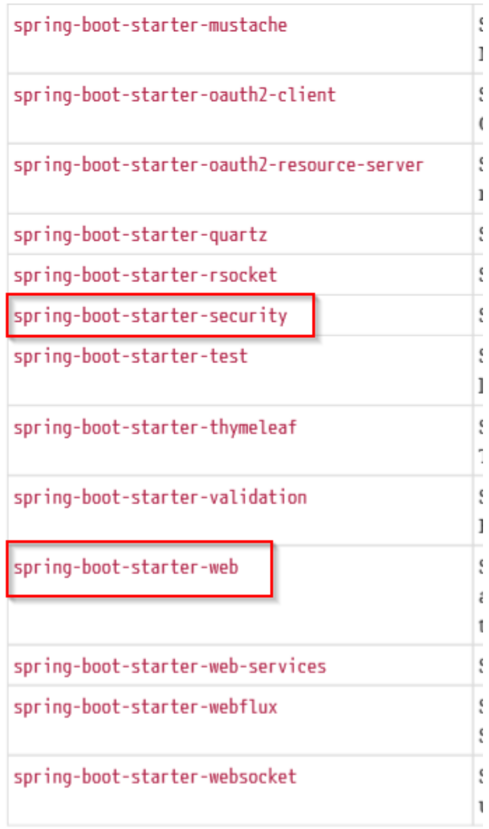
- Starters



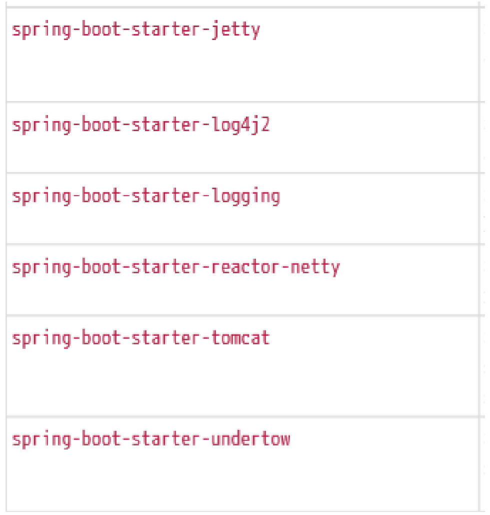
spring-boot-starter
spring-boot-starter-activemq
spring-boot-starter-amqp
spring-boot-starter-aop
spring-boot-starter-artemis
spring-boot-starter-batch
spring-boot-starter-cache
spring-boot-starter-data-cassandra
spring-boot-starter-data-cassandra-reactive
spring-boot-starter-data-couchbase
spring-boot-starter-data-couchbase-reactive
spring-boot-starter-data-elasticsearch
spring-boot-starter-data-jdbc
spring-boot-starter-data-jpa



spring-boot-starter-data-ldap
spring-boot-starter-data-mongodb
spring-boot-starter-data-mongodb-reactive
spring-boot-starter-data-neo4j
spring-boot-starter-data-r2dbc
spring-boot-starter-data-redis
spring-boot-starter-data-redis-reactive
spring-boot-starter-data-rest
spring-boot-starter-data-solr
spring-boot-starter-freemarker
spring-boot-starter-groovy-templates
spring-boot-starter-hateoas
spring-boot-starter-integration
spring-boot-starter-jdbc
spring-boot-starter-jersey
spring-boot-starter-jooq
spring-boot-starter-json
spring-boot-starter-jta-atomikos
spring-boot-starter-jta-bitronix
spring-boot-starter-mail



spring-boot-starter-mustache
spring-boot-starter-oauth2-client
spring-boot-starter-oauth2-resource-server
spring-boot-starter-quartz
spring-boot-starter-rsocket
spring-boot-starter-security
spring-boot-starter-test
spring-boot-starter-thymeleaf
spring-boot-starter-validation
spring-boot-starter-web
spring-boot-starter-web-services
spring-boot-starter-webflux
spring-boot-starter-websocket



spring-boot-starter-jetty
spring-boot-starter-log4j2
spring-boot-starter-logging
spring-boot-starter-reactor-netty
spring-boot-starter-tomcat
spring-boot-starter-undertow

netcompany

Spring Boot – Maven setup

```xml
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.2.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

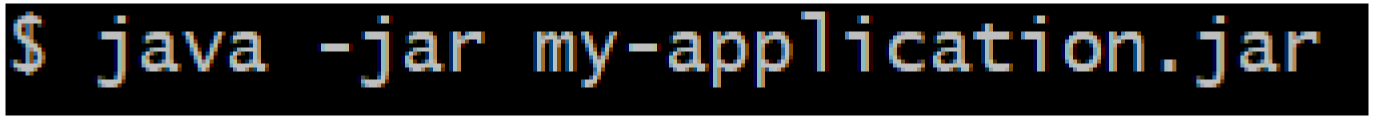
netcompany

Spring Boot

```java
@SpringBootApplication // same as @Configuration @EnableAutoConfiguration @ComponentScan
public class Application {

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

}
```

Spring Boot – Running

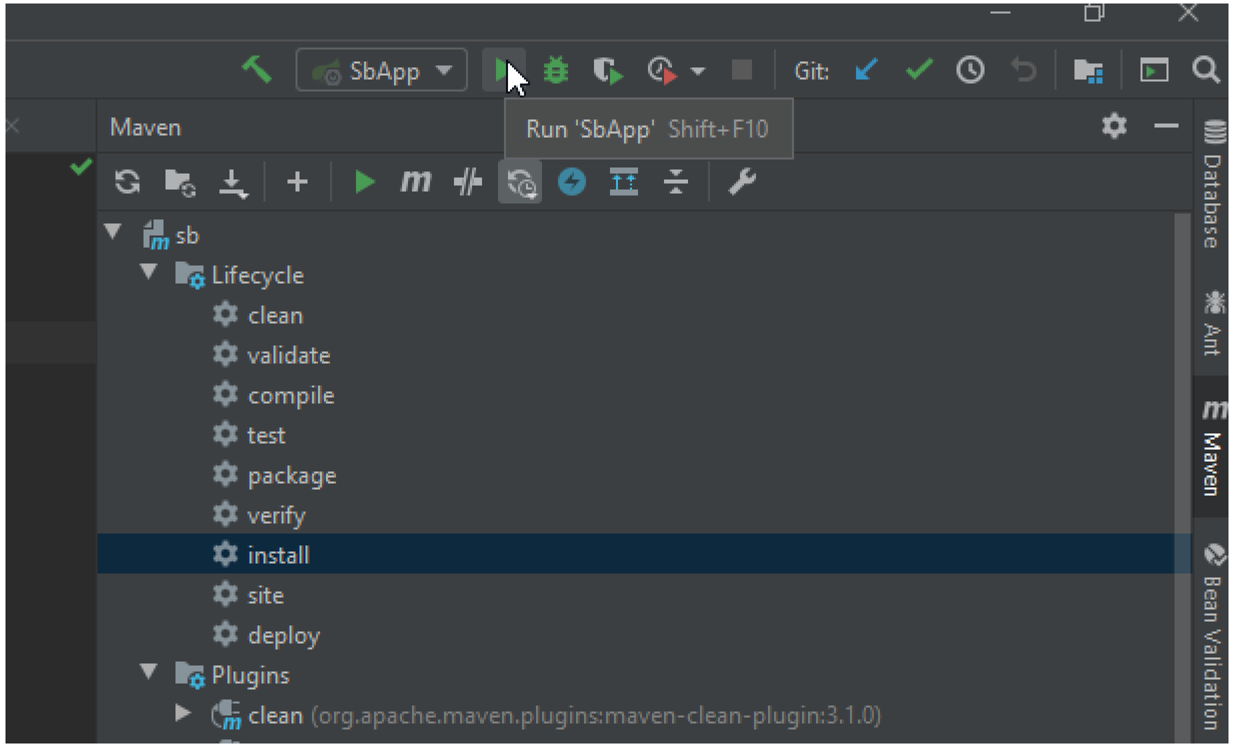
Then we simply run it:



```
$ java -jar my-application.jar
```

netcompany

Spring Boot – Running in IntelliJ



sb
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
clean (org.apache.maven.plugins:maven-clean-plugin:3.1.0)

Testing

- @SpringBootTest
  - Integrationtests
  - Rest endpoint
  - Auth/Roles/access
- Mockito
  - Businesslogic
- Data driven tests
  - Test data in your database
  - SQL/JPA criterion/hql

A bit about PaaS

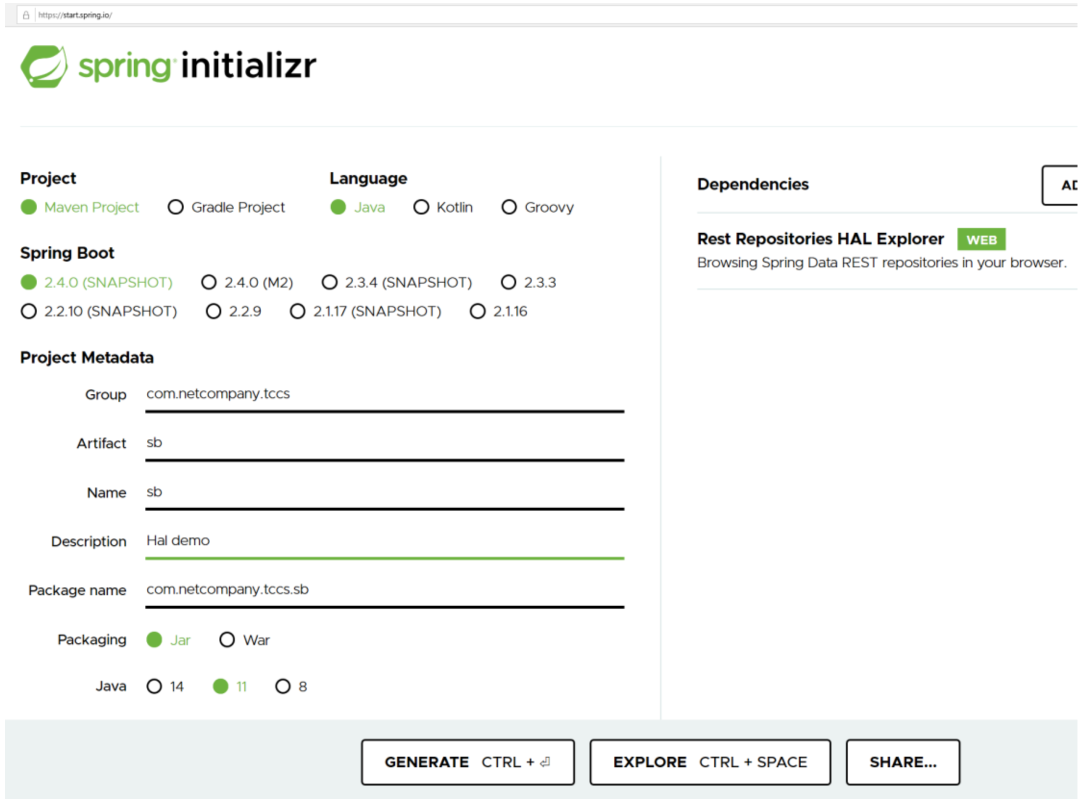
- A Platform we can use to run and manage our app in the cloud
  - Easy and quick setup
  - Initially cheap
- Some vendors
  - Heroku
  - AWS (Elastic Beanstalk)
  - Google App Engine
  - Azure

A bit about Paas

- You can commit you code to the provider
  - Provider examines, builds and runs you app
- You can also point to your own repo
  - Provider fetches, examines, builds and runs your app
- Or you can commit a packaged app
  - You upload a jar
  - Provider runs it

Spring initializr

- Fast and easy way to get relevant dependencie
  - https://start.spring.io/



https://start.spring.io/

spring initializr

Project
[✓ Maven Project] Gradle Project

Language
[✓ Java] Kotlin Groovy

Spring Boot
[✓ 2.4.0 (SNAPSHOT)] 2.4.0 (M2) 2.3.4 (SNAPSHOT) 2.3.3
2.2.10 (SNAPSHOT) 2.2.9 2.1.17 (SNAPSHOT) 2.1.16

Project Metadata

Group	com.netcompany.tccs
Artifact	sb
Name	sb
Description	Hal demo
Package name	com.netcompany.tccs.sb
Packaging	[✓ Jar] War
Java	14 [✓ 11] 8

Dependencies AD

Rest Repositories HAL Explorer WEB
Browsing Spring Data REST repositories in your browser.

Spring Boot – Worth mentioning

- Spring-devtools
  - Restart two classloaders, beware production
- Validation
- Lombok
- Retrofit
- OpenApi(Swagger)
- Flyway
- Templating
  - spring-boot-starter-mustache

Getting started quickly in practice - Workshop

netcompany

Setup

- IntelliJ
- A Heroku account
- Heroku CLI installed
- Postman installed

Tasks - Setting

- We are creating a new app for our client X
- X wants a register of its customers
- X needs something delivered fast!
- Requirements
  - It should have a REST resources
  - It should have a database
  - It should run in the cloud
  - It must be secure

netcompany

Solutions in branches

- Each task ha a solution in a branch. Use this if you are stuck and want to see a running application

Task 1

- Create your own spring boot app from scratch running locally
- It must contain a REST resource
  - Responding to GET '/hello/[name]' with the text: "Hello [name]!", replacing name with the parameter
  - Tip clone : https://github.com/netcompanyno/tccs_springboot.git
  - `>git checkout init`
  - Postman collection is added to init branch as well as this PowerPoint as a PDF

Task 2

- You can remove or hide the hello resource
- Now we need some functionality
- Create a REST resource
  - Which can register a customers, first name, last name, age, date of birth, email address, consent for storing this information
  - NB! Do **not** store the customer, only **receive** it.

Task 3

- We of course need to **store** the customers info for later retrieval
- Store it in a database
  - You can use an in memory DB
  - https://spring.io/guides/gs/accessing-data-jpa/

Task 4

- Now we want to be able to fetch a customer
- Lets assume whoever uses our REST service knows a customers id
- Create a REST resource to fetch a customer by its id

Task 5 (Use no more than 30 min)

- Now we need to move this app to the cloud
- The customer will not allow you to share the source code with Heroku.
- Package the app, and find a way to run it on Heroku
  - Tip (windows users): Use windows cmd for “heroku login”. Then git bash for commands.
  - Tip 2: Connect to you Heroku account using Heroku CLI
  - Tip 3: Use Heroku CLI Deploy plugin or Heroku maven plugin

netcompany

Task 6 (Hard)

- This is all great according to X, but “is our customers data secure?” they ask.
- Oh no, it’s not is it?
- Let’s fix that
  - Tip: Check out WebSecurityConfigurerAdapter
  - Tip 2: Make this simple for now, use basic auth for example

Task 7

- We need to make sure that we access some environment specific properties in our app.
- Use Spring to load properties detailing which environment the app is running in
  - Dev for local
  - Prod for heroku
- Make sure it works by having different values locally and in Heroku
- Find a way to show this while running the app

Task 8

- Add some metrics to your app for monitoring
- Explore what kind of metrics you can get from the spring-boot-starter-actuator artifact
- Make sure you can at least call a health endpoint on your app

Task 9

- AOP
  - Add logging annotation to rest endpoints
- Caching
  - Add caching to a duplicated retrieve endpoint
- Scheduler
  - Create a recurring output from a method in a Spring bean

netcompany

Task 10

- X also wants us to be ready to send and receive JMS messages for later integrations
- Set up your app so that it can send and receive messages on an ActiveMQ queue (or similar JMS queue)
- Make sure to set up your own queue so that you can test that it works

Task 11

- Deploy your app to another of the PaaS suppliers (you decide)
- Some alternatives
  - Amazon AWS
  - Google App Engine