@[Entity|Document] public class A {  
 @Id [@GeneratedValue(strategy=GenerationType.$)] private long $;  
 @Temporal(TemporalType.$) private Date $  
 @OneToMany(cascade = CascadeType.$, mappedBy=”attribuut\_B”, fetch = FetchType.$)  
 private List<B> $  
 public $ (…){} <- lege constructor VERPLICHT }  
@[Entity|Document] public class B {  
 @ManyToOne(fetch = FetchType.$, optional = [true|false])  
 @JoinColumn(name = “A\_id”, nullable = [true|false])  
 @OnDelete(action = OnDeleteAction.$)  
 A; }  
public interface $ extends [MongoRepository|CrudRepository]<T, V>{  
 @Query(“SELECT x from y x WHERE x.attribuut = :parameter”)  
 [List|Optional]<Klasse> $ (@Param(“parameter”) [int|String|…] parameter);  
 // Bij enum heel het pad vermelden: package.enumklasse.keuze }  
@Component public class $Service {  
 @Autowired private Repository;  
 public $ $ { return repository.$ } }  
[application.properties] server.port=$  
@RestController public class $Controller {  
 @Autowired private $Service;  
 @RequestMapping(value=”/path/{abc}”, method=RequestMethod.$)  
 1) public $ $ () -> /path  
 2) public $ $(@PathVariable(“abc”) $ $) -> /path/{abc} (ook aanpassen in value)  
 3) public $ $ (@RequestBody Object o) -> verwacht JSON of gelijkaardig  
 4) public $ $ (@RequestParam(“$”) $ $) -> form-encoded data  
 { // aanspreken service } }  
[terminal] bin/zookeeper-server-start.sh config/zookeeper.properties  
[terminal] bin/kafka-server-start.sh config/server.properties  
public interface ProducerChannels {  
 String DO\_X = “do\_x”;  
 @Output(DO\_X) MessageChannel doX(); }  
@EnableBinding(ProducerChannels.class) <- IN MAIN APPLICATIE  
[application.properties] spring.cloud.stream.bindings.do\_x.destination=do\_x  
[application.properties] spring.cloud.stream.bindings.do\_x.contentType=application/json  
[application.properties] spring.cloud.stream.bindings.do\_x.group=do\_x\_group <- bij consumers  
@MessagingGateway  
public interface MessageChannelGateway {  
 @Gateway(requestChannel = ProducerChannels.DO\_X) void do\_x(params) }  
[terminal] bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic do\_x   
public interface ConsumerChannels {  
 String X\_HAPPENED = “x\_happened”;  
 @Input(X\_HAPPENED) SubscribableCHannel xHappened();}  
@EnableBinding(ConsumerChannels.class) <- IN MAIN APPLICATIE  
@Component  
public class $CommandHandler {  
 @Autowired private $Service service;  
 @StreamListener(ConsumerChannels.X\_HAPPENED)  
 public … () { }  
}  
public RouteLocator locator(RouteLocatorBuilder b) {  
 return b.routes().route(r -> r.host(“\*”).and().path(“/x/\*\*”).uri(<http://localhost:xxxx>).build()}  
docker ps -a # alle containers tonen  
docker inspect <docker object> # informatie over een image, container, … tonen  
docker run [-d] [--link <container>] [--name <naam>] [-e var1=’x’] [-e var2=’y’] <image> # start image  
docker start [--attach] <container> # STDOUT/STDERR uitprinten van container  
docker container logs <container> # logs van een bepaalde container  
docker rm <container> | docker container prune # [1 | alle] GESTOPTE container(s) verwijderen  
docker pull <image> # bestaande image van de Docker hub halen  
docker exec -i -t <container> /bin/bash # interactie BINNEN een bepaalde container  
FROM <bestaande\_image>  
WORKDIR <directory in container>  
COPY <directory\_op\_systeem> <directory\_in\_container>  
RUN cmd1 && cmd2 && … && cmdn  
CMD <uitvoeren\_jar\_ofzo>  
docker build -t <eigen\_image\_naam> . 🡨 punt niet vergeten op het eindedoc  
[application.properties] spring.jpa.hibernate.ddl-auto=create  
[application.properties] spring.datasource.url=jdbc:mysql://<mysql\_container>:3306/table  
[application.properties] spring.datasource.[username|password]=…  
[application.prop…]spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect   
todo: vanaf 4.3 API gateway Container Image