2024-05-24

Einführung in Spring Data

code of change

Hyand by GOD | MT

Sebastian Lammering

Berufserfahrung

CDI | Part of Hyand
15 Jahre 8 Monate

Principal Consultant
Teilzeit
Jan. 2022–Heute · 2 Jahre 5 Monate
Dortmund, Nordrhein-Westfalen, Deutschland · Hybrid

Ausbildung

Fachhochschule Dortmund
Diplom-Ingenieur, Informatik
2004–2008

Agenda

- Was ist Spring?
- Was ist Spring Boot?
- Welche Module gibt es in Spring?
- Livecoding einer Spring Boot Anwendung

Was ist Spring?

- Dependency Injection
- Aspektorientierte Entwicklung (AOP)
- Templates

Was ist dependency injection?

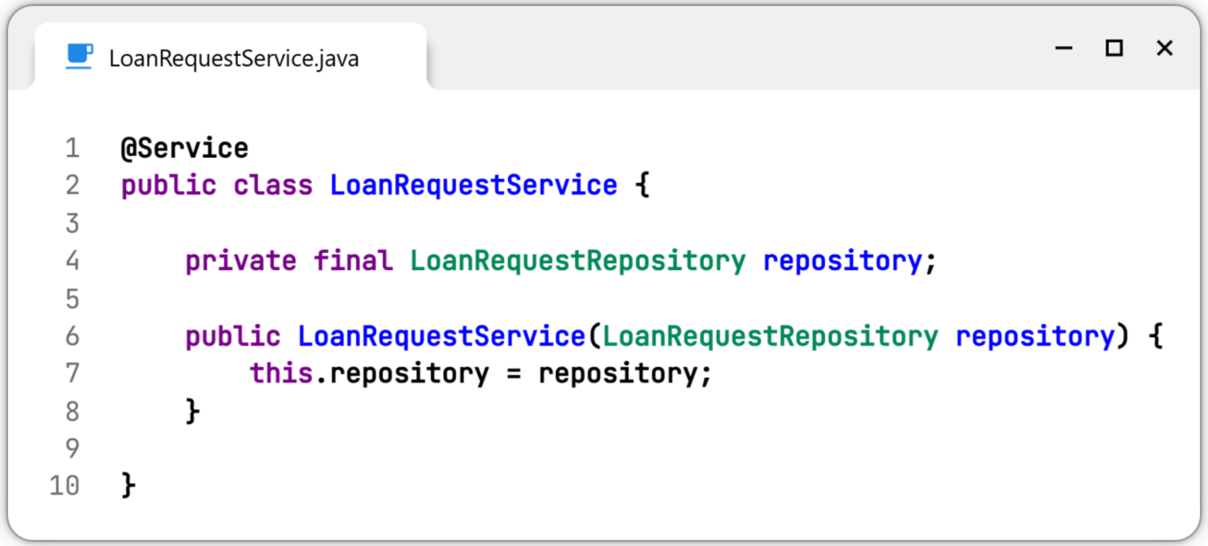
Dependency Injection bedeutet, einem Objekt oder einer Funktion die benötigten Abhängigkeiten von außen zur Verfügung zu stellen, anstatt sie intern zu erstellen.

Anstatt dass jedes Objekt seine eigenen Abhängigkeiten verwaltet, zentralisiert DI diese Verantwortung und macht den Code modularer und wartbarer.

Warum dependency injection verwenden?

- Lose Kopplung
- Vorteile beim Testen
- Flexibilität bei der Konfiguration

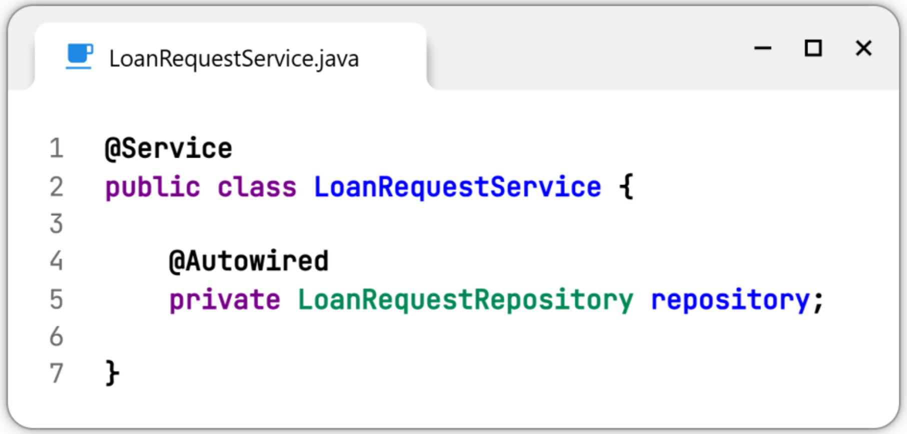
Arten von Dependency Injection



```java
@Service
public class LoanRequestService {

    private final LoanRequestRepository repository;

    public LoanRequestService(LoanRequestRepository repository) {
        this.repository = repository;
    }

}
```

Constructor injection

Field / Setter injection

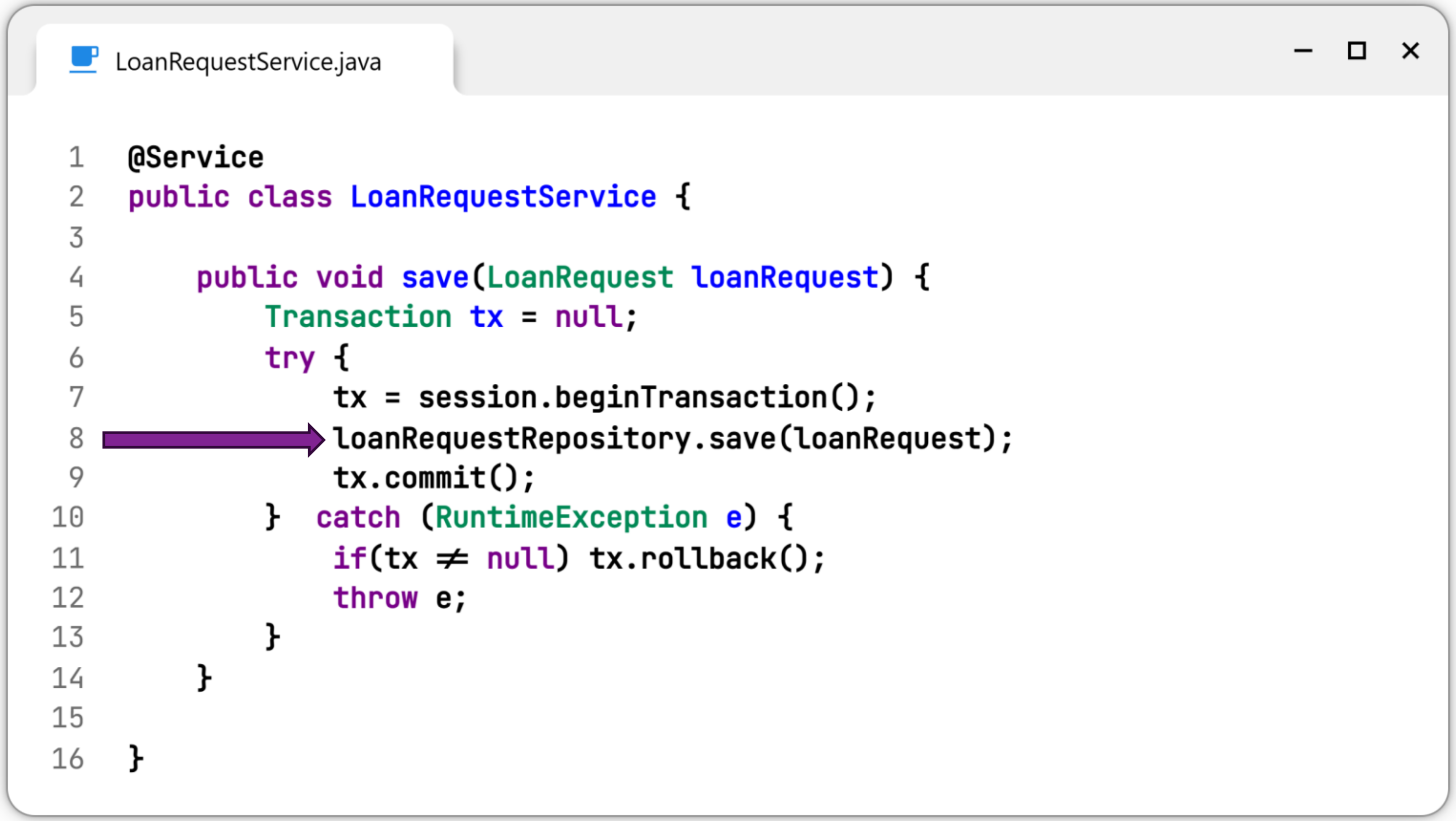


```java
@Service
public class LoanRequestService {

    @Autowired
    private LoanRequestRepository repository;

}
```

Aspektorientierte Entwicklung (AOP)

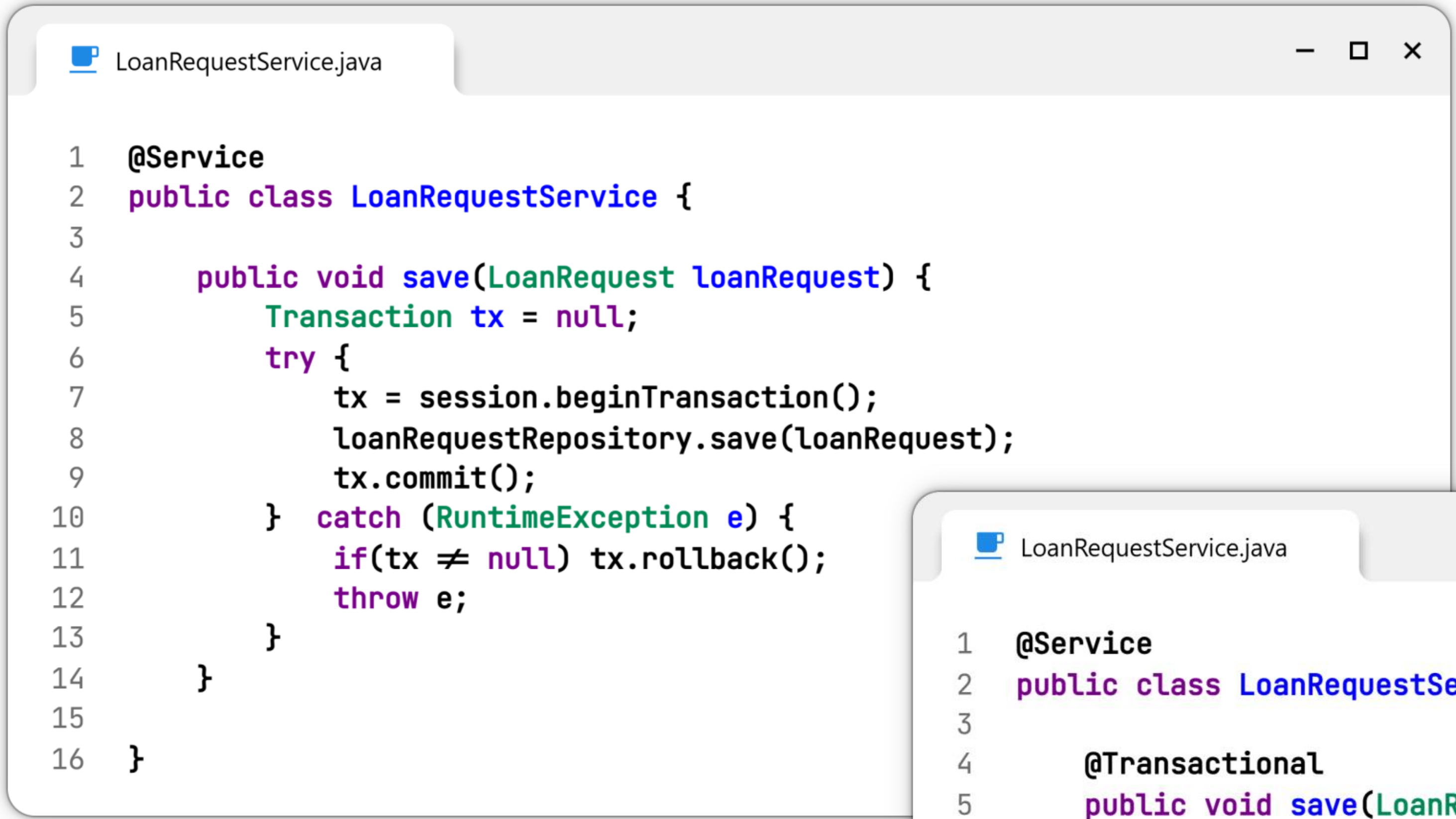
- Einfache Berücksichtigung von generischen Aspekten im Code (Cross-Cutting-Concerns)
- Generische Aspekte einer Softwareanwendung sind zum Beispiel
    - Transaktionssteuerung
    - Zugriffskontrolle
    - Logging
- AOP sorgt für einfacheren und wartbaren Code und hilft dabei den Fokus auf die Fachlichkeit zu legen

Aspektorientierte Entwicklung (AOP)

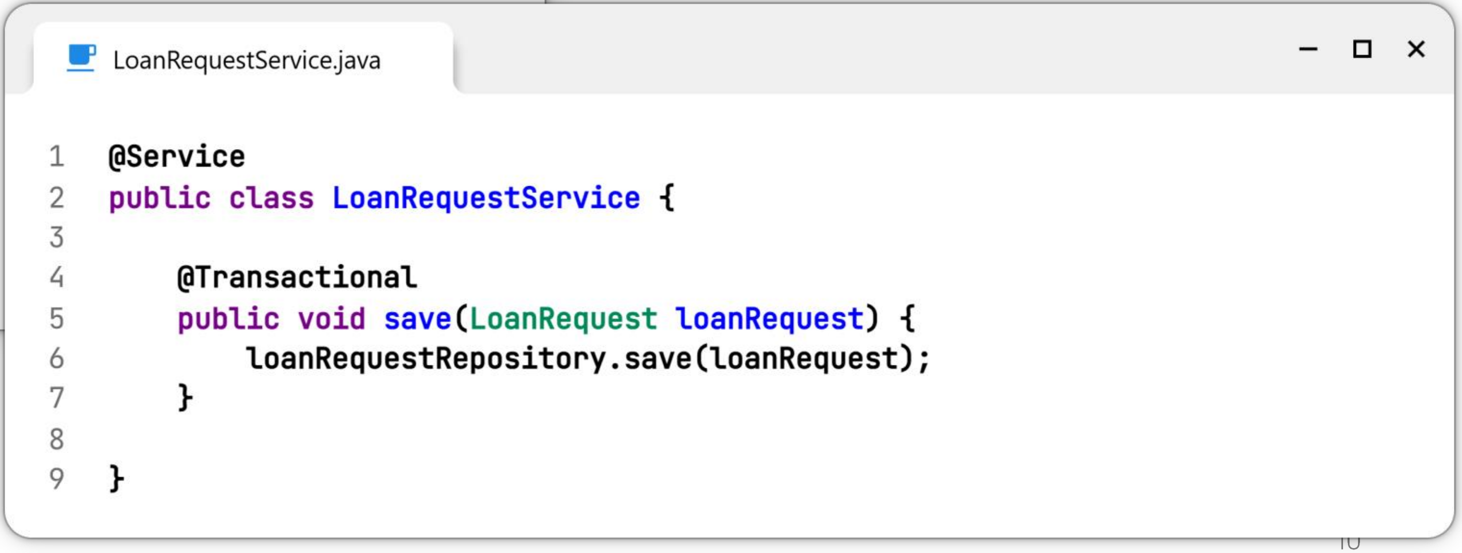


```java
@Service
public class LoanRequestService {

    public void save(LoanRequest loanRequest) {
        Transaction tx = null;
        try {
            tx = session.beginTransaction();
            loanRequestRepository.save(loanRequest);
            tx.commit();
        }  catch (RuntimeException e) {
            if(tx ≠ null) tx.rollback();
            throw e;
        }
    }

}
```

Aspektorientierte Entwicklung (AOP)



```java
@Service
public class LoanRequestService {

    public void save(LoanRequest loanRequest) {
        Transaction tx = null;
        try {
            tx = session.beginTransaction();
            loanRequestRepository.save(loanRequest);
            tx.commit();
        }  catch (RuntimeException e) {
            if(tx ≠ null) tx.rollback();
            throw e;
        }
    }

}
```



```java
@Service
public class LoanRequestService {

    @Transactional
    public void save(LoanRequest loanRequest) {
        loanRequestRepository.save(loanRequest);
    }

}
```

Hyand

Welche Vorteile hat Spring Boot?

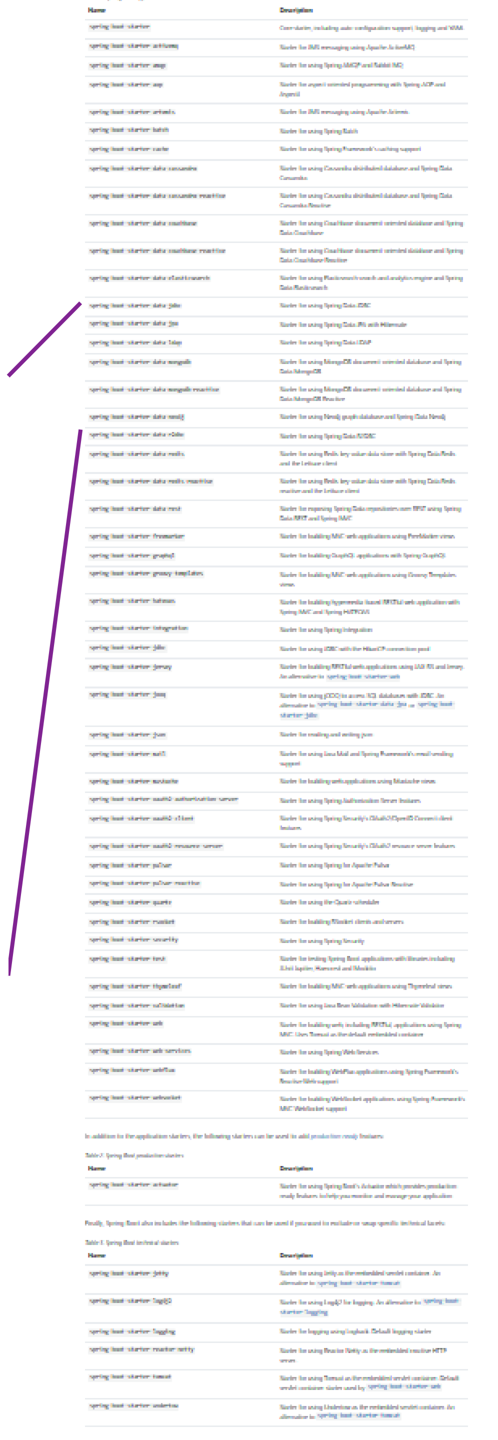
- Spring Boot starters
- Autokonfiguration
- Convention over Configuration (Coc)
- Eigenständige Anwendung

Spring starter Module

`spring-boot-starter-data-jdbc`	Starter for using Spring Data JDBC
`spring-boot-starter-data-jpa`	Starter for using Spring Data JPA with Hibernate
`spring-boot-starter-data-ldap`	Starter for using Spring Data LDAP
`spring-boot-starter-data-mongodb`	Starter for using MongoDB document-oriented database and Spring Data MongoDB
`spring-boot-starter-data-mongodb-reactive`	Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive
`spring-boot-starter-data-neo4j`	Starter for using Neo4j graph database and Spring Data Neo4j

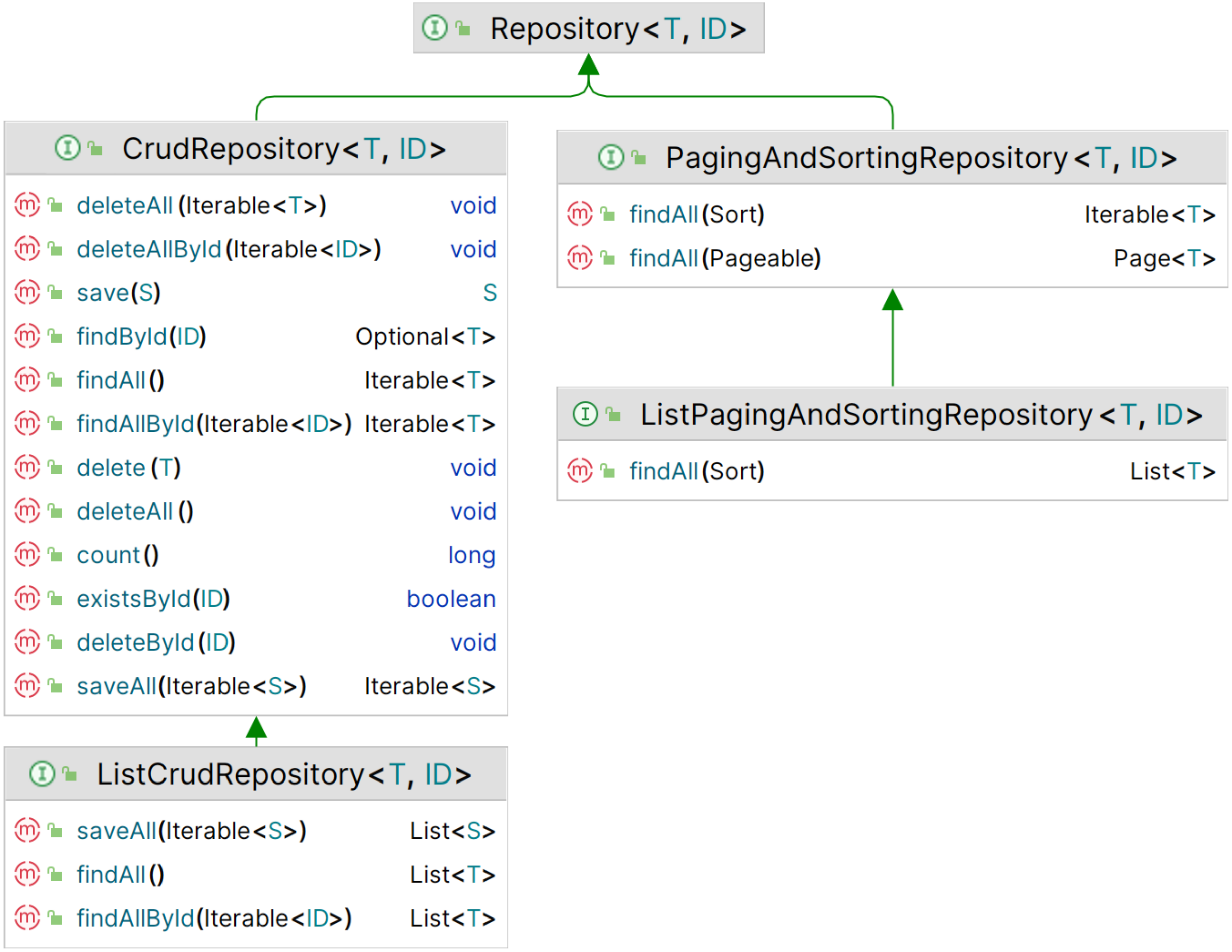
Spring starter Module

`spring-boot-starter-data-jdbc`	Starter for using Spring Data JDBC
`spring-boot-starter-data-jpa`	Starter for using Spring Data JPA with Hibernate
`spring-boot-starter-data-ldap`	Starter for using Spring Data LDAP
`spring-boot-starter-data-mongodb`	Starter for using MongoDB document-oriented database and Spring Data MongoDB
`spring-boot-starter-data-mongodb-reactive`	Starter for using MongoDB document-oriented database and Spring Data MongoDB Reactive
`spring-boot-starter-data-neo4j`	Starter for using Neo4j graph database and Spring Data Neo4j



Hyand

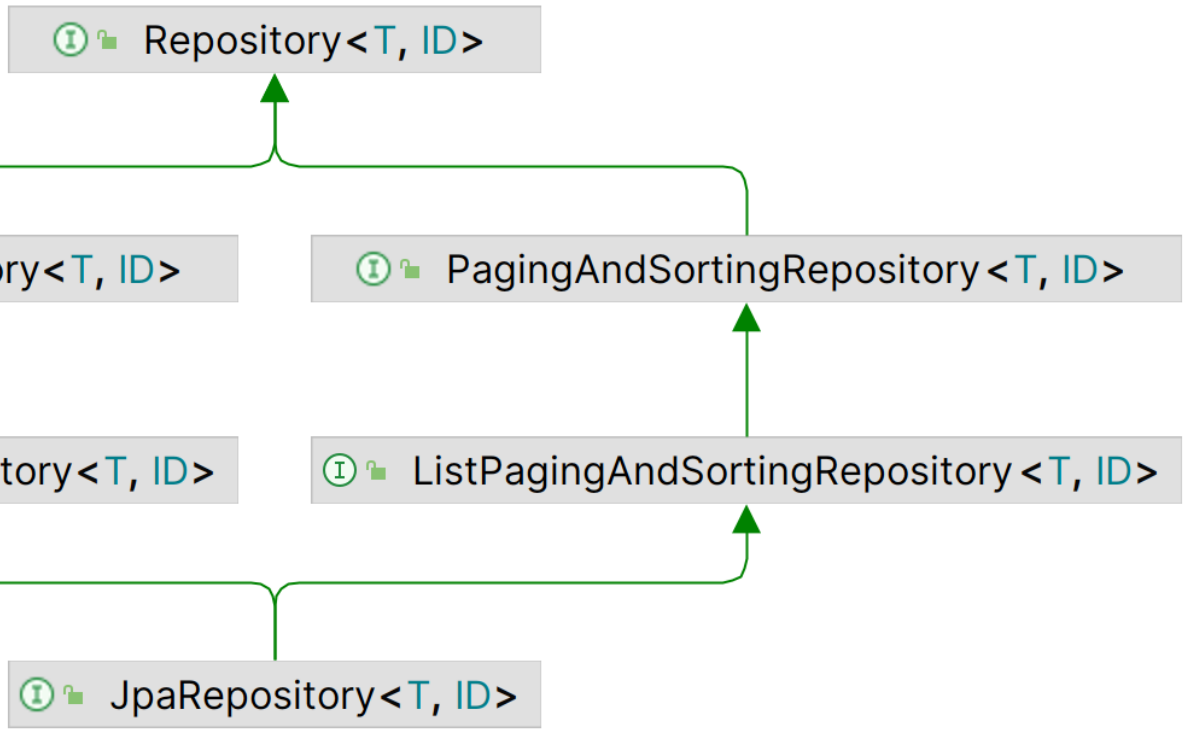
Spring Data Repositories



Repository<T, ID>
CrudRepository<T, ID>
deleteAll(Iterable<T>) void
deleteAllById(Iterable<ID>) void
save(S) S
findById(ID) Optional<T>
findAll() Iterable<T>
findAllById(Iterable<ID>) Iterable<T>
delete(T) void
deleteAll() void
count() long
existsById(ID) boolean
deleteById(ID) void
saveAll(Iterable<S>) Iterable<S>
ListCrudRepository<T, ID>
saveAll(Iterable<S>) List<S>
findAll() List<T>
findAllById(Iterable<ID>) List<T>
PagingAndSortingRepository<T, ID>
findAll(Sort) Iterable<T>
findAll(Pageable) Page<T>
ListPagingAndSortingRepository<T, ID>
findAll(Sort) List<T>

Hyand

Spring Data JPA Repository


Repository<T, ID>
CrudRepository<T, ID>
PagingAndSortingRepository<T, ID>
ListCrudRepository<T, ID>
ListPagingAndSortingRepository<T, ID>
JpaRepository<T, ID>

Codebeispiel

Hyand

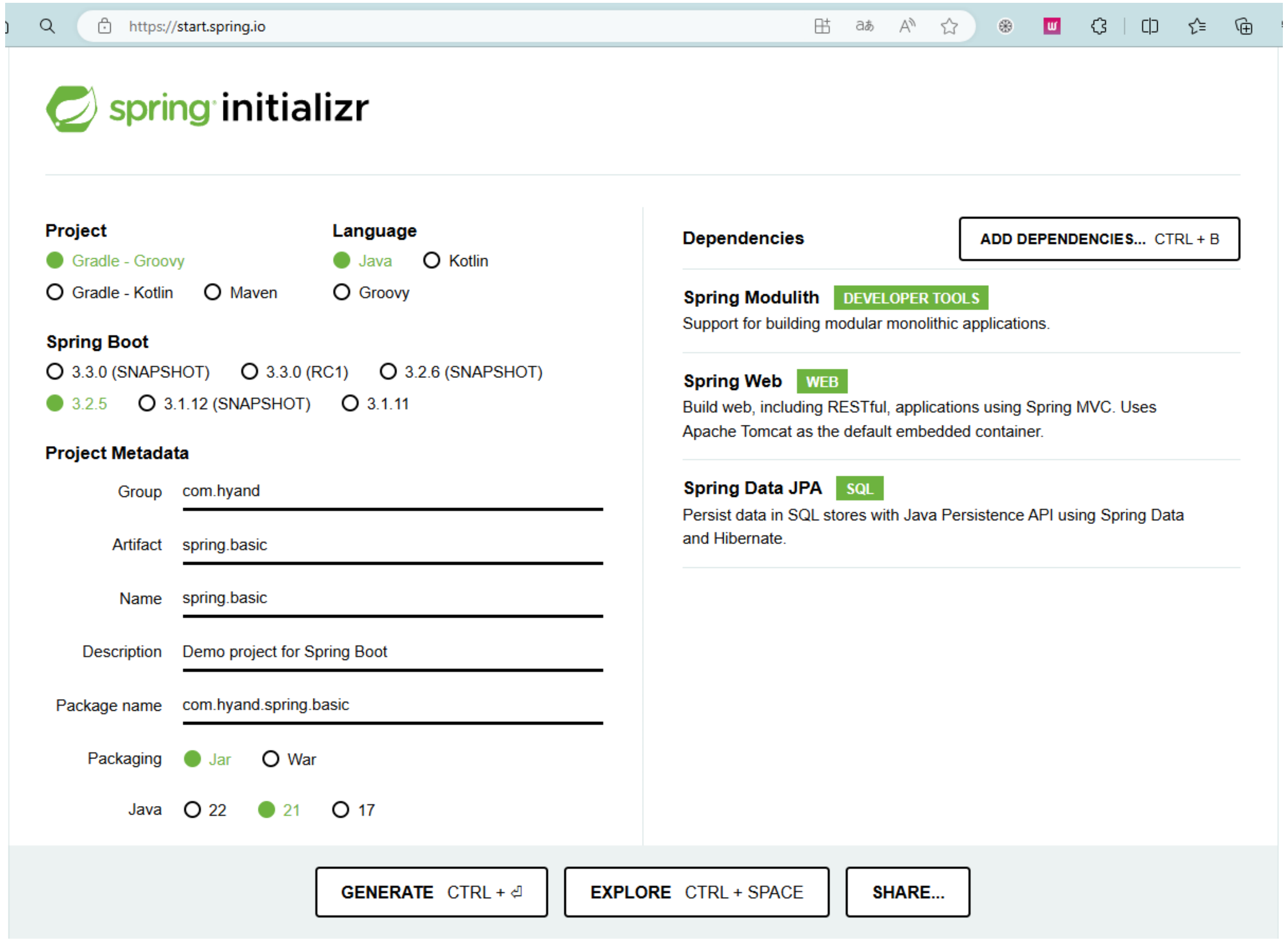
Codebespiel

- Spring initializr
- Entitäten und Repositories definieren
  - Generierung von Spring
- Service
  - Dependency Injection der Repositories
  - Transaktionskontrolle mit Aspektorientierter Entwicklung
- UI Entwicklung
  - Controller
  - Html Seiten

Codebespiel - Frameworks

Thymeleaf

Spring intializr – https://start.spring.io



Project
[✓ Gradle - Groovy] Gradle - Kotlin Maven

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.3.0 (SNAPSHOT) 3.3.0 (RC1) 3.2.6 (SNAPSHOT) [✓ 3.2.5] 3.1.12 (SNAPSHOT) 3.1.11

Project Metadata

Group	com.hyand
Artifact	spring.basic
Name	spring.basic
Description	Demo project for Spring Boot
Package name	com.hyand.spring.basic
Packaging	[✓ Jar] War
Java	22 [✓ 21] 17

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Modulith DEVELOPER TOOLS
Support for building modular monolithic applications.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Live Coding

(ok, mit Fangnetz)

Hyand

Vielen Dank für eure Aufmerksamkeit

Sebastian Lammering

sebastian.lammering@hyand.com
https://www.linkedin.com/in/sebastian-lammering

Codebeispiel
https://github.com/slammering/spring-data-basic

Hyand