LU08a - Vereinfachungen

Nachfolgend werden einige Bibliotheken gezeigt, um das Schreiben von Code zu vereinfachen.

Project Lombok

Project Lombok hilft Boilerplate-Code in Datenklassen auf ein Minimum zu reduzieren.

Klassisch	Lombok

Klassisch

```
public class Person {

    private Long id;
    private String firstName;
    private String lastName;
    private int age;

    // Standard-Konstruktor
    public Person() {
    }

    // Konstruktor mit allen Feldern
    public Person(Long id, String
firstName, String lastName, int age)
{
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    // Getter und Setter
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String
firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String
lastName) {
        this.lastName = lastName;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Lombok

```
import lombok.Getter;
import lombok.Setter;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;


@Getter                 // generiert
Getter
@Setter                 // generiert
Setter
@NoArgsConstructor      // generiert
Standard-Konstruktor
@AllArgsConstructor     // generiert
Konstruktor mit allen Feldern
public class Person {
    private Long id;
    private String firstName;
    private String lastName;
    private int age;
}
```

Mit `@Data` kann man sogar Getter, Setter, toString, equals und hashCode generieren lassen.

Mehr dazu auf https://www.cegos-integrata.de/blog/it-blog/programmiersprachen-blog/project-lombok-java-klassen-effizient-gestalten

JpaRepository

JpaRepository erleichtert sehr viel, um Objekte in oder aus der Datenbank zu laden.

Klassisch

```
import jakarta.persistence.*;
import java.util.List;

public class PersonPersistor {

    private EntityManagerFactory emf;

    public PersonPersistor() {
        this.emf =
Persistence.createEntityManagerFactory("localPU");
    }

    public List<Person> findByLastName(String
lastName) {
        EntityManager em =
emf.createEntityManager();
        List<Person> result;

        try {
            em.getTransaction().begin();
            TypedQuery<Person> query =
em.createQuery(
                    "SELECT p FROM Person p WHERE
p.lastName = :lastName", Person.class);
            query.setParameter("lastName",
lastName);
            result = query.getResultList();
            em.getTransaction().commit();
        } catch (Exception e) {
            em.getTransaction().rollback();
            throw e;
        } finally {
            em.close();
        }
        return result;
    }

    public void close() {
        emf.close();
    }
}
```

JpaRepository

```
import
org.springframework.data.jpa.repository.JpaRepository;
import java.util.List;

public interface PersonRepository extends
JpaRepository<Person, Long> { // Long ist der Datentyp
des Primärschlüssels
    List<Person> findByLastName(String lastName);
}
```

Schlüsselwort	Bedeutung
`findBy`	SELECT … WHERE
`existsBy`	prüft, ob mindestens ein Eintrag existiert
`countBy`	zählt Datensätze
`deleteBy`	löscht Datensätze
`And`, `Or`	Verknüpfung mehrerer Bedingungen
`LessThan`, `GreaterThan`, `Between`, `Like`, `Not`, `In`	Vergleichsoperatoren
`OrderBy…Asc/Desc`	Sortierung

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/ffit/3-jahr/java/learningunits/lu08/a?rev=1761609222

Last update: **2025/10/28 00:53**